UNIVERSIDAD PONTIFICIA COMILLAS
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)
MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

APLICACIÓN DE TÉCNICAS DE VISIÓN ARTIFICIAL A LA DETECCIÓN DE DEFECTOS EN CAJAS DE BATERÍAS

Autor: Jon Toledo Bengoechea

Directora: Symone Gomes Soares Alcalá

Co-Director: Álvaro Jesús López López

Madrid

Agosto de 2025

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título APLICACIÓN DE TÉCNICAS DE VISIÓN ARTIFICIAL A LA DETECCIÓN DE DEFECTOS EN CAJAS DE BATERÍAS en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2024/2025 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Jon Toledo Bengoechea Fecha: 27 / 08 / 2025

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Documento assinado digitalmente
gov.br
SYMONE GOMES SOARES ALCALA
Data: 26/08/2025 13:11:43-0300
Verifique em https://validar.iti.gov.br

Fdo.: Symone Gomes Soares Alcalá Fecha: 26/08/2025

APLICACIÓN DE TÉCNICAS DE VISIÓN ARTIFICIAL A LA DETECCIÓN DE DEFECTOS EN CAJAS DE BATERÍA

Autor: Toledo Bengoechea, Jon.
Directora: Gomes Soares Alcalá, Symone
Entidad Colaboradora: Gestamp

RESUMEN DEL PROYECTO

Este trabajo presenta un sistema de visión artificial basado en deep learning para la detección de defectos en agujeros de tapas de batería. Se compararon tres arquitecturas: YOLOv11 para detección y ResNet18 y EfficientNet-B0 para clasificación. Tras la preparación y aumento del conjunto de datos, los modelos alcanzaron precisiones superiores al 97% y elevados valores de recall, mostrando su viabilidad para integrarse en procesos de control de calidad industrial en tiempo real.

Palabras clave: Visión artificial, Deep learning, Inspección automática, Detección de defectos, Procesamiento de imágenes, Industria automotriz

1. Introducción

La introducción de técnicas de visión artificial en la industria automotriz surge como respuesta a las limitaciones de la inspección manual y a la creciente exigencia de procesos de fabricación más seguros, eficientes y consistentes. Diversos estudios han demostrado que la combinación de cámaras industriales con algoritmos de deep learning permite superar los problemas de variabilidad humana y garantizar estándares de calidad cada vez más estrictos [1]. Estas tecnologías permiten implementar inspecciones automáticas en tiempo real, mejorando la repetibilidad y reduciendo significativamente los errores asociados a la fatiga y subjetividad del operario [2].Asimismo, la integración de visión artificial con sistemas inteligentes de producción responde directamente a los principios de la Industria 4.0, donde la automatización avanzada y el análisis de datos en línea son claves para aumentar la competitividad [3]. En este contexto, se plantea la necesidad de investigar soluciones avanzadas que, además de asegurar la detección fiable de defectos, contribuyan a la sostenibilidad operativa de las empresas y a la reducción de costes derivados de rechazos y reprocesos.

2. Definición del proyecto

El proyecto parte del reto planteado por Gestamp, orientado a la inspección de tapas de baterías de automóvil. El problema identificado consiste en la detección automática de fallos de soldadura en las arandelas de los orificios, un aspecto crítico para garantizar la calidad de las piezas. Se requiere un sistema robusto, capaz de operar en un entorno industrial real, frente a variaciones de iluminación, tipos de defectos y diferencias superficiales, y que pueda integrarse en la línea de producción sin afectar a su ritmo.

En este marco, los objetivos del proyecto se centran en diseñar y validar un sistema de visión artificial que clasifique los orificios como correctos o defectuosos, descarte automáticamente las tapas con fallos y registre los defectos detectados para su análisis posterior. Para lograrlo, se compararán distintos modelos de deep learning con el fin de seleccionar la opción más precisa, rápida y robusta para su aplicación práctica en planta.

3. Descripción del sistema

El sistema de visión artificial desarrollado se diseñó para automatizar el proceso de inspección de tapas de baterías, abarcando desde la captura de imágenes hasta el entrenamiento de modelos de *deep learning.* (Ilustración 1*)*. La adquisición de datos se realizó a partir de seis cámaras situadas estratégicamente en la línea de producción, lo que permitió obtener una cobertura completa de cada pieza. Sobre estas imágenes se implementó un algoritmo propio de segmentación automática, encargado de detectar y recortar cada orificio de la tapa de manera individual, generando un conjunto de imágenes homogéneo y adaptado al problema específico de detección de defectos de soldadura. Esta etapa de segmentación resultó clave para garantizar la escalabilidad y repetibilidad del sistema, reduciendo al mínimo la intervención manual.

Debido al desequilibrio existente entre las clases (la mayoría de los orificios se encontraban en estado correcto), fue necesario aplicar técnicas de aumento de datos para ampliar y diversificar el conjunto de imágenes disponibles, asegurando así un entrenamiento más robusto y equilibrado. A continuación, se llevó a cabo un proceso

de etiquetado y anotación: por un lado, se definieron etiquetas binarias ("ok"/"nok") para la clasificación de los orificios, y por otro se generaron anotaciones específicas para tareas de detección de objetos con YOLO. Finalmente, se estableció la división del conjunto de datos en entrenamiento, validación y prueba, y se entrenaron distintos modelos de *deep learning*, incluyendo arquitecturas de clasificación y detección, con configuraciones cuidadosamente ajustadas. Este flujo metodológico constituye la base técnica sobre la que se evaluaron los resultados presentados en los capítulos posteriores.

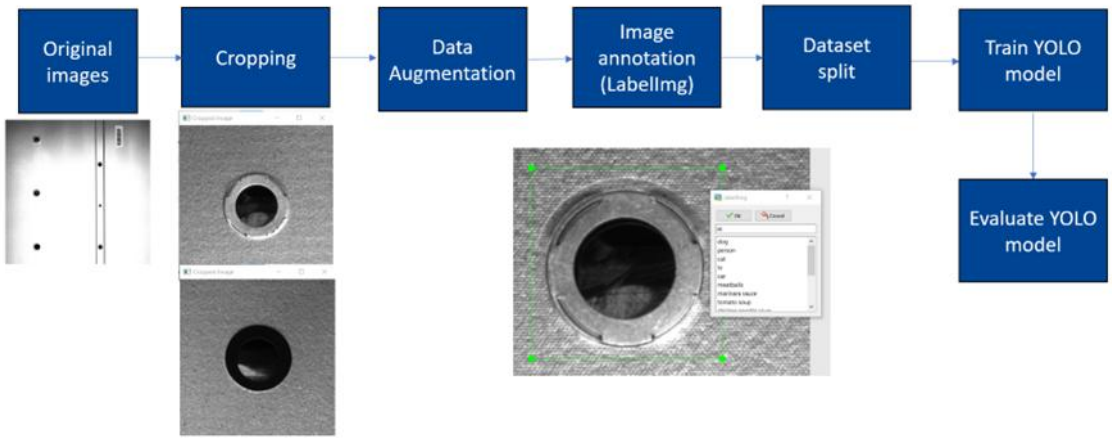

Ilustración 1: Flujo de trabajo del sistema desarrollado

4. Resultados

Los resultados del sistema se evaluaron en dos etapas diferenciadas: el conjunto de validación y el conjunto de prueba. En primer lugar, sobre el conjunto de validación, los tres modelos entrenados respondieron de manera sobresaliente, alcanzando métricas de accuracy, recall y F1-score en torno al 97%. En esta fase no se observaron diferencias relevantes entre las arquitecturas, lo que confirma que todas ellas fueron capaces de generalizar correctamente sobre los datos no vistos durante el entrenamiento.

Por otro lado, en el conjunto de prueba, además de las métricas de rendimiento, se analizó el tiempo de inferencia como factor clave de viabilidad industrial. En este escenario sí se observaron diferencias significativas: los modelos de clasificación basados en redes convolucionales ofrecieron tiempos de predicción reducidos (del orden de 1,5 segundos por tapa), mientras que el modelo YOLO duplicó dicho

tiempo debido a la mayor complejidad de su arquitectura de detección. Sin embargo, a pesar de este incremento en los tiempos, YOLO mostró métricas ligeramente superiores en la identificación de defectos, con precisión del 90%, lo que refuerza su idoneidad para casos donde la precisión del control de calidad prime sobre la velocidad de ejecución.

5. Conclusiones

El proyecto ha cumplido con los objetivos planteados, demostrando la eficacia de un sistema de visión artificial para la detección de fallos de soldadura en tapas de baterías. Entre las principales conclusiones destacan:

- Se analizó el proceso industrial de Gestamp, identificando necesidades reales y criterios de rechazo en planta.
- Se entrenaron y validaron tres modelos (YOLO, EfficientNet y ResNet), todos con alta precisión en validación.
- YOLO mostró mayor robustez en el conjunto de prueba, aunque con tiempos de inferencia superiores.
- El sistema clasifica automáticamente agujeros en OK/NOK, garantizando decisiones claras y consistentes.
- Se diseñó un mecanismo de rechazo total por pieza defectuosa, alineado con los estándares industriales.
- El registro de defectos por posición aporta trazabilidad y facilita análisis de causa raíz en producción.

En conjunto, el sistema constituye una solución práctica y escalable que puede complementar o sustituir la inspección manual, mejorando la eficiencia y la fiabilidad en el control de calidad automotriz.

6. Referencias

[1]. Ameri, R., Hsu, C.-C., & Band, S. S. (2024). A systematic review of deep learning approaches for surface defect detection in industrial applications. Engineering Applications of Artificial Intelligence, 130, 107717. https://doi.org/10.1016/j.engappai.2023.107717

[2]. Dai, W., Li, D., Tang, D., Jiang, Q., Wang, D., Wang, H., & Peng, Y. (2021). Deep learning assisted vision inspection of resistance spot welds. Journal of

Manufacturing Processes, 62, 262-274. https://doi.org/10.1016/j.jmapro.2020.12.015

[3]. Büchi, G., Cugno, M., & Castagnoli, R. (2020). Smart factory performance and Industry 4.0. Technological Forecasting and Social Change, 150, 119790. https://doi.org/10.1016/j.techfore.2019.119790

APPLICATION OF COMPUTER VISION TECHNIQUES FOR DEFECT DETECTION IN BATTERY CASES

Author: Toledo Bengoechea, Jon.
Supervisor: Gomes Soares Alcalá, Symone
Collaborating Entity: Gestamp

ABSTRACT

This work presents a computer vision system based on deep learning for the detection of defects in battery lid holes. Three architectures were compared: YOLOv11 for detection, and ResNet18 and EfficientNet-B0 for classification. After dataset preparation and augmentation, the models achieved accuracies above 97% and high recall values, demonstrating their feasibility for integration into real-time industrial quality control processes.

Keywords: Computer vision, Deep learning, Automated inspection, Defect detection, Image processing, Automotive industry

1. Introduction

The introduction of computer vision techniques in the automotive industry arises as a response to the limitations of manual inspection and the growing demand for safer, more efficient, and consistent manufacturing processes. Several studies have shown that combining industrial cameras with deep learning algorithms can overcome issues related to human variability and ensure increasingly stringent quality standards [1]. These technologies enable the implementation of real-time automated inspections, improving repeatability and significantly reducing errors associated with operator fatigue and subjectivity [2]. Furthermore, the integration of computer vision with intelligent production systems directly aligns with the principles of Industry 4.0, where advanced automation and online data analysis are key to enhancing competitiveness [3]. In this context, the need arises to investigate advanced solutions that not only ensure reliable defect detection but also contribute to the operational sustainability of companies and to the reduction of costs derived from rejections and reprocessing.

2. Project Definition

The project originates from a challenge proposed by Gestamp, focused on the inspection of automotive battery lids. The identified problem consists of the automatic detection of welding defects in the washers of the holes, a critical aspect to ensure the quality of the components. A robust system is required, capable of operating in a real industrial environment, handling variations in lighting, defect types, and surface differences, while being seamlessly integrated into the production line without affecting its throughput.

Within this framework, the objectives of the project are centered on designing and validating a computer vision system that classifies holes as correct or defective, automatically rejects faulty lids, and records detected defects for further analysis. To achieve this, different deep learning models will be compared in order to select the most accurate, fast, and robust option for practical deployment on the production floor.

3. Descripción del sistema

The developed computer vision system was designed to automate the inspection process of battery lids, covering the entire workflow from image acquisition to deep learning model training (Ilustración 2). Data acquisition was carried out using six strategically positioned cameras on the production line, enabling complete coverage of each component. On these images, a proprietary automatic segmentation algorithm was implemented, responsible for detecting and cropping each hole of the lid individually, thereby generating a homogeneous dataset tailored to the specific problem of weld defect detection. This segmentation stage proved crucial to ensure the scalability and repeatability of the system, while minimizing manual intervention.

Due to the imbalance between classes (as most holes were in a correct state), data augmentation techniques were applied to expand and diversify the available dataset, thus ensuring more robust and balanced training. Subsequently, a labeling and annotation process was conducted: on the one hand, binary labels (“ok”/“nok”) were defined for hole classification, and on the other hand, specific annotations were generated for object detection tasks using YOLO. Finally, the dataset was divided into training, validation, and test sets, and various deep learning models were trained,

including both classification and detection architectures, with carefully tuned configurations. This methodological pipeline constitutes the technical foundation upon which the results presented in the subsequent chapters were evaluated

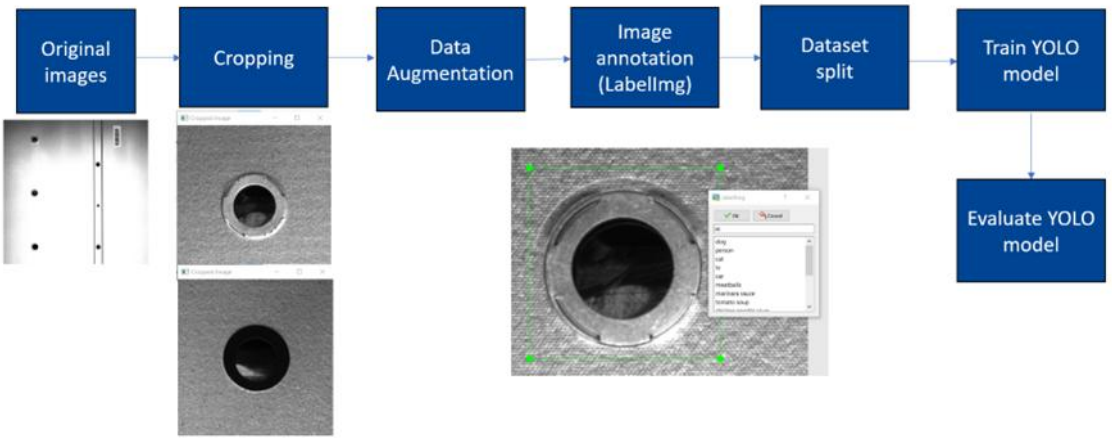

Ilustración 2: Flujo de trabajo del sistema desarrollado

4. Results

The system results were evaluated in two distinct stages: the validation set and the test set. First, on the validation set, the three trained models performed outstandingly, achieving accuracy, recall, and F1-score metrics around 97%. In this phase, no relevant differences were observed between the architectures, confirming that all of them were able to generalize correctly to unseen data during training.

On the other hand, in the test set, in addition to performance metrics, inference time was analyzed as a key factor for industrial feasibility. In this scenario, significant differences were observed: classification models based on convolutional networks offered reduced prediction times (around 1.5 seconds per lid), whereas the YOLO model doubled this time due to the greater complexity of its detection architecture. However, despite this increase in processing time, YOLO showed slightly superior metrics in defect identification, with an accuracy of 90%, reinforcing its suitability for cases where quality control precision takes precedence over execution speed.

5. Conclusions

The project has successfully achieved the proposed objectives, demonstrating the effectiveness of a computer vision system for detecting welding defects in battery lids. The main conclusions can be summarized as follows:

- The industrial process at Gestamp was analyzed, identifying real needs and rejection criteria on the production floor.
- Three models (YOLO, EfficientNet, and ResNet) were trained and validated, all achieving high accuracy on the validation set.
- YOLO proved to be more robust on the test set, although with higher inference times.
- The system automatically classifies holes as OK/NOK, ensuring clear and consistent decisions.
- A full-part rejection mechanism was designed for defective components, aligned with industrial standards.
- Defect logging by position provides traceability and facilitates root cause analysis in production.

Overall, the system represents a practical and scalable solution that can complement or replace manual inspection, enhancing efficiency and reliability in automotive quality control.

6. References

[1]. Ameri, R., Hsu, C.-C., & Band, S. S. (2024). A systematic review of deep learning approaches for surface defect detection in industrial applications. Engineering Applications of Artificial Intelligence, 130, 107717. https://doi.org/10.1016/j.engappai.2023.107717

[2]. Dai, W., Li, D., Tang, D., Jiang, Q., Wang, D., Wang, H., & Peng, Y. (2021). Deep learning assisted vision inspection of resistance spot welds. Journal of Manufacturing Processes, 62, 262-274. https://doi.org/10.1016/j.jmapro.2020.12.015

[3]. Büchi, G., Cugno, M., & Castagnoli, R. (2020). Smart factory performance and Industry 4.0. Technological Forecasting and Social Change, 150, 119790. https://doi.org/10.1016/j.techfore.2019.119790

Índice de la memoria

Índice de figuras

Índice de tablas

Capítulo 1. Introducción y planteamiento del problema

1.1. Contexto

Los procesos de inspección y control de calidad son elementos fundamentales dentro de la cadena de producción en la industria automovilística. Hay que asegurar que los componentes fabricados están libres de defectos no solo es esencial para garantizar la seguridad de los futuros usuarios, sino también para la sostenibilidad económica de las empresas, su eficiencia operativa y la confianza que transmiten a sus clientes y socios industriales.

Tradicionalmente, la inspección de calidad se ha llevado a cabo de forma manual mediante la observación directa por parte de operarios especializados. Este enfoque, aunque extendido y probado durante décadas, presenta importantes limitaciones. Entre los principales inconvenientes se encuentran la fatiga del operario, la variabilidad en los criterios de inspección, la falta de repetibilidad y la dificultad de mantener altos ritmos de producción sin comprometer la fiabilidad del control.

En las últimas décadas, el avance de las tecnologías de automatización ha permitido introducir sistemas de visión artificial como alternativa o complemento a la inspección manual [1]. Estos sistemas utilizan cámaras industriales, técnicas de procesamiento de imágenes y, en muchos casos, algoritmos de inteligencia artificial para detectar de forma automática defectos superficiales, ausencia de elementos, deformaciones o roturas en las piezas fabricadas.

El uso de visión artificial en entornos industriales ha demostrado importantes ventajas en términos de velocidad, precisión y homogeneidad del proceso [2]. Al eliminar los factores humanos más problemáticos, como la subjetividad y el cansancio, se consigue una mayor fiabilidad en los resultados. Además, la posibilidad de registrar digitalmente cada inspección permite una trazabilidad completa y favorece la integración con otros sistemas de calidad y producción.

En este contexto, la aplicación de técnicas de visión artificial no solo responde a una necesidad técnica, sino que también se alinea con las exigencias de la Industria 4.0, que promueve la automatización inteligente, la toma de decisiones basada en datos y la mejora continua de los procesos productivos.

1.2. Motivación

La inspección y control de calidad son procesos esenciales en la industria automovilística, donde cada componente debe cumplir con estándares estrictos para asegurar la seguridad y durabilidad de los vehículos. En este contexto, la inspección visual y los ensayos no destructivos, aunque efectivos, presentan limitaciones importantes en términos de precisión, coste y eficiencia en producciones a gran escala [3].

La precisión y consistencia que exige la industria automotriz —con requerimientos de calidad superiores al 99 %— son difíciles de alcanzar con inspección manual, debido a la variabilidad inherente al factor humano. Además, al operar a altas cadencias, los procesos manuales pueden convertirse rápidamente en cuellos de botella y fuentes de errores que impactan en los costes operativos y en la productividad de la planta.

La integración de sistemas de visión artificial combinados con *deep learning* representa una transformación sustancial [4]. Las redes neuronales profundas son capaces de aprender directamente a partir de ejemplos, reconociendo patrones complejos —incluso defectos nunca vistos previamente— y adaptándose automáticamente a variaciones en iluminación, posición y textura de las piezas. Asimismo, estos sistemas eliminan factores como la fatiga, la subjetividad o la inconsistencia humana, permitiendo un control de calidad continuo, homogéneo y en tiempo real.

Este proyecto no solo representa una mejora en eficiencia, sino también una oportunidad clara de reducción de costes operativos, al disminuir el número de piezas defectuosas, los retrabajos y las interrupciones en la línea. Además, la generación automática de registros y trazabilidad por pieza facilita la auditoría y el análisis posterior, alineándose con las exigencias de la Industria 4.0.

La adopción de tecnologías como el *deep learning* también se inscribe en una tendencia global hacia la industria inteligente [5], donde la automatización, el análisis de datos y la toma de decisiones digitales son factores clave para mejorar la competitividad empresarial.

1.3. Definición del problema

En este contexto se sitúa el reto planteado por Gestamp, una empresa multinacional española especializada en el diseño, desarrollo y fabricación de componentes metálicos para el sector automovilístico [6]. Esta compañía, que colabora con numerosos fabricantes globales, busca garantizar la fiabilidad de sus piezas mediante el uso de tecnologías avanzadas. Uno de los desafíos identificados consiste en la inspección de un componente concreto: las tapas de baterías de automóvil.

El objetivo específico es desarrollar un sistema automatizado que sea capaz de detectar con precisión fallos de soldadura en las arandelas de los orificios de dichas tapas. El sistema debe identificar y descartar automáticamente las tapas que no cumplen con los criterios de calidad establecidos, todo ello sin intervención humana y con la capacidad de integrarse en un entorno de producción real. La solución debe ser robusta frente a variaciones de iluminación, tipo de defectos y diferencias superficiales entre piezas, además de operar con suficiente velocidad para no ralentizar el flujo de trabajo en la línea.

Este problema plantea, por tanto, la necesidad de diseñar e implementar un sistema de visión artificial adaptado a las particularidades del caso de uso, capaz de aplicar algoritmos de procesamiento y análisis que permitan clasificar correctamente las piezas como válidas o defectuosas.

1.4. Objetivos del proyecto

Objetivo general

Desarrollar un sistema de visión artificial capaz de detectar automáticamente fallos de soldadura en las arandelas de los agujeros de tapas de baterías para automóviles, con el fin de descartar piezas defectuosas y mejorar la eficiencia del control de calidad.

Objetivos específicos

1. **Analizar el contexto industrial y comprender el problema planteado por Gestamp**

Antes del diseño del sistema, se realizará un estudio del proceso de fabricación y del sistema de inspección actual, con especial atención al formato y calidad de las imágenes disponibles, los tipos de defectos esperados y los criterios de rechazo utilizados. Esta fase permitirá establecer las bases técnicas y operativas del desarrollo posterior.

2. **Construir y validar un modelo de visión computacional**

Se entrenarán y evaluarán varios modelos de inteligencia artificial ya desarrollados con el objetivo de detectar fallos en las tapas de baterías. La comparación de su rendimiento permitirá seleccionar el modelo más adecuado según criterios de precisión, velocidad y robustez. Esta estrategia responde a enfoques comunes en proyectos industriales de visión artificial, donde se evalúan diferentes arquitecturas antes de tomar decisiones de despliegue [7].

3. **Establecer una clasificación de agujeros en clases 'ok' y 'nok'**

Cada orificio será clasificado automáticamente como 'ok' si no presenta defectos visibles, o 'nok' si no presenta de una arandela correctamente soldada. Este método binario es empleado habitualmente en sistemas de control industrial y asegura una decisión operativa clara.

4. **Descartar automáticamente las piezas defectuosas**

Si al menos un agujero es marcado como 'nok', la tapa será retirada de la línea. Esto responde a la lógica de inspección automatizada, donde un único fallo implica el rechazo del conjunto.

5. **Registrar la localización y patrón de los defectos**

El sistema generará un registro de los agujeros defectuosos para identificar patrones recurrentes y posibles fallos en la línea de montaje, facilitando análisis de causa raíz y acciones correctivas.

1.5. Alineación con los Objetivos de Desarrollo Sostenible (ODS)

ODS 8: Trabajo decente y crecimiento económico

Al mejorar la precisión y eficiencia en los procesos de inspección mediante tecnologías avanzadas, este proyecto contribuye a un crecimiento económico más sostenible y a la competitividad de la industria automovilística. Tal como se destaca, los robots y sistemas automatizados en sectores manufactureros incrementan la productividad, la eficiencia en el uso de recursos y abren la puerta a un entorno laboral más seguro y cualificado. Además, al reducir el trabajo repetitivo y propenso a errores humanos, se mejora el entorno laboral y se liberan recursos humanos para tareas de mayor valor y cualificación.

ODS 9: Industria, innovación e infraestructura

Este proyecto promueve la innovación al integrar tecnologías avanzadas como el deep learning y la visión artificial en la industria automotriz. Fomenta la creación de infraestructura inteligente y sostenible, y respalda una industria resiliente e inclusiva. La aplicación de inteligencia artificial y automatización constituye un motor clave para la transformación digital de la manufactura, alineado con la Industria 4.0.

ODS 12: Producción y consumo responsables

La implementación de un sistema de inspección más preciso y eficiente contribuye directamente a la reducción de residuos y la optimización del uso de materiales, en consonancia con el ODS 12. La competencia en tiempo real del sistema permite identificar piezas defectuosas antes de su ensamblaje, reduciendo la necesidad de reprocesos y el consumo innecesario de energía y recursos.

Figura 1: Iconos de los ODS 8 (Trabajo decente), 9 (Industria e innovación) y 12 (Producción responsable). Fuente: Adaptado de los materiales oficiales de la ONU sobre los Objetivos de Desarrollo Sostenible

Capítulo 2. Descripción de las tecnologías

El objetivo de este capítulo es presentar los fundamentos teóricos de las tecnologías y modelos utilizados en este trabajo, centrado en la aplicación de visión artificial para detectar defectos en cajas de baterías. La metodología se basa en *deep learning*, una rama del aprendizaje automático que ha demostrado un rendimiento sobresaliente en análisis de imágenes, especialmente en entornos industriales y en tareas de inspección visual.

Primero, se explicará el aprendizaje profundo, seguida de una introducción a la visión artificial y al uso de redes neuronales convolucionales (CNN), que constituyen la columna central de las arquitecturas empleadas. A continuación, se describen en detalle los modelos seleccionados (YOLO, ResNet, EfficientNet), explicando su funcionamiento, características y ventajas, en especial su uso para tareas de detección y clasificación de defectos en imágenes industriales.

Este marco teórico es esencial para comprender las decisiones técnicas del proyecto y para analizar correctamente los resultados obtenidos, aportando así rigor a la justificación de la elección de arquitecturas y facilitando la interpretación de las métricas de desempeño en los capítulos posteriores.

2.1. Inteligencia Artificial

2.1.1. De la inteligencia artificial al aprendizaje automático

El desarrollo de la inteligencia artificial (IA) ha seguido una evolución progresiva desde sus orígenes en la década de 1950 hasta las aplicaciones modernas basadas en redes neuronales profundas. Esta evolución puede dividirse en tres grandes etapas: inteligencia artificial simbólica, aprendizaje automático (machine learning) y aprendizaje profundo (deep learning).

Inteligencia Artificial (IA)

La IA surgió como disciplina formal en 1956, durante la Conferencia de Dartmouth [8], donde investigadores como John McCarthy, Marvin Minsky y Allen Newell propusieron que era posible simular la inteligencia humana mediante máquinas. En esta primera etapa,

conocida como IA simbólica o "GOFAI" (Good Old-Fashioned Artificial Intelligence), los sistemas se basaban en reglas lógicas, razonamiento deductivo y conocimiento explícito codificado manualmente. Estos sistemas, también llamados "expertos", resultaban eficaces para problemas bien definidos, pero mostraban importantes limitaciones en contextos abiertos o con gran variabilidad de datos.

Machine Learning

A medida que los datos disponibles aumentaban y las limitaciones de los enfoques simbólicos se hacían evidentes, surgió un nuevo paradigma: el aprendizaje automático (machine learning), orientado a dotar a los sistemas de la capacidad de aprender directamente a partir de los datos. El término fue popularizado por Arthur Samuel en 1959, quien lo definió como "el campo de estudio que da a las computadoras la capacidad de aprender sin ser explícitamente programadas" [9]. Este enfoque introdujo técnicas como perceptrones, SVMs y árboles de decisión, fundamentales para tareas de clasificación y predicción.

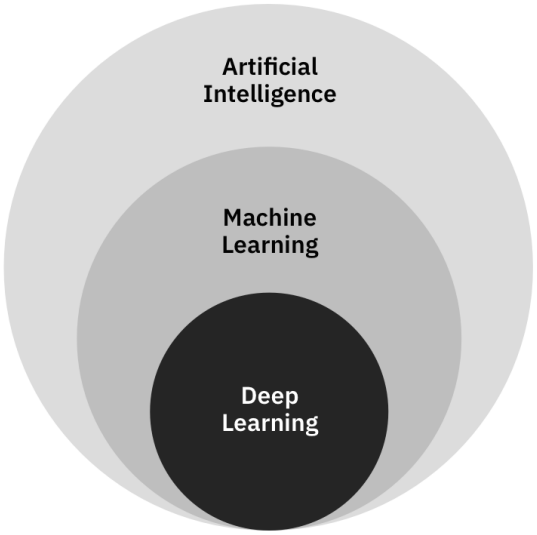

Figura 2: Relación jerárquica entre la inteligencia artificial, el aprendizaje automático y el aprendizaje profundo. Fuente: IBM, Machine learning – basics "Machine Learning" (2022).

2.1.2. Redes neuronales artificiales

Las redes neuronales artificiales (ANN, por sus siglas en inglés) son el componente fundamental sobre el que se construye el aprendizaje profundo [10]. Inspiradas en el funcionamiento del cerebro humano, estas redes están compuestas por unidades llamadas neuronas artificiales, organizadas en capas que procesan información de forma jerárquica.

Neurona artificial: pesos, sesgo y función de activación

Una neurona artificial es la unidad básica de procesamiento en una red neuronal. Su funcionamiento se inspira en las neuronas biológicas, aunque en un nivel mucho más simplificado. Desde un punto de vista computacional, una neurona artificial recibe un conjunto de valores de entrada (también llamados inputs, habitualmente representados como un vector), que pueden corresponder, por ejemplo, a los valores de los píxeles de una imagen o a características extraídas de un conjunto de datos.

Cada una de esas entradas está asociada a un peso, el cual determina la importancia relativa de esa entrada en el proceso de decisión de la neurona. A estas combinaciones ponderadas se les suma un valor adicional llamado sesgo, que permite desplazar la función de activación y mejorar la flexibilidad del modelo. El resultado de esta operación, es decir, la suma de todas las entradas multiplicadas por sus respectivos pesos más el sesgo, se denomina combinación lineal.

Este valor intermedio es procesado posteriormente por una función de activación no lineal. Esta función introduce no linealidad en la salida de la neurona, permitiendo que redes compuestas por muchas de estas unidades sean capaces de aproximar funciones complejas. La salida final de la neurona se transmite entonces a otras neuronas en capas posteriores o se interpreta directamente como el resultado del modelo, según la arquitectura utilizada.

Este mecanismo sencillo, replicado y conectado a gran escala en una red profunda, constituye la base del aprendizaje automático moderno mediante *deep learnin*g.

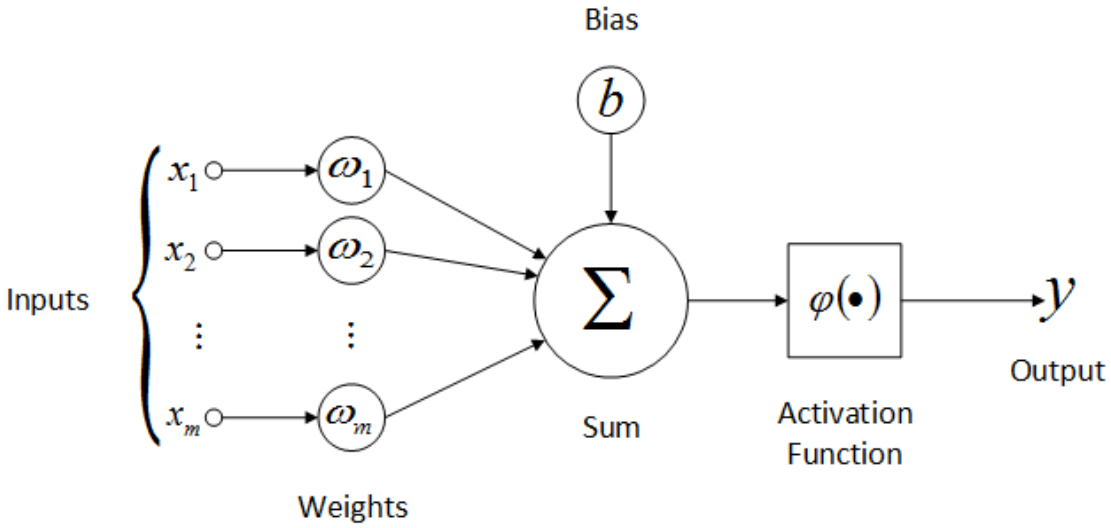

Figura 3: Esquema funcional de una neurona artificial.

La Figura 3 ilustra el funcionamiento interno de una neurona artificial típica en una red neuronal. Cada entrada x_i del conjunto de datos se conecta a la neurona a través de un peso sináptico w_i, que regula la influencia de dicha entrada sobre la salida. Las flechas indican estas conexiones ponderadas, que pueden ser excitadoras (cuando el peso es positivo) o inhibidoras (cuando el peso es negativo).

Todos los productos entre entradas y pesos se suman en el nodo de combinación lineal, representado por el símbolo Σ. A esto normalmente se añade un bias b (sesgo), resultando en un valor:

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

Sobre esa combinación lineal z, se aplica una función de activación φ, que introduce no linealidad y además controla la escala de la salida. Sin esta no linealidad, aunque la red tenga muchas capas, su comportamiento sería equivalente al de una simple transformación lineal, lo que limitaría severamente su capacidad de aprendizaje. Gracias a las funciones de activación, las redes profundas pueden aproximar funciones altamente no lineales y resolver tareas como la clasificación de imágenes, el reconocimiento de voz o la traducción automática.

Las funciones de activación más utilizadas incluyen:

- **Función sigmoide (logística):** Produce una salida en el rango (0, 1), lo que la hace útil en tareas de clasificación binaria.

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

- **Tangente hiperbólica (tanh):** Similar a la sigmoide, pero su salida está centrada en cero y oscila entre -1 y 1.

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

- **ReLU (Rectified Linear Unit):** Es la función más común en redes neuronales profundas modernas. Introduce una activación no lineal simple pero muy eficiente computacionalmente. Su principal inconveniente es que puede hacer que algunas neuronas queden inactivas permanentemente (problema del "neurona muerta").

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

- **Leaky ReLU y otras variantes**: Para mitigar el problema anterior, existen variantes como Leaky ReLU, que permite un pequeño gradiente para valores negativos.

$$LeakyReLU(x) = \max(\alpha x, x)\,, con\ \alpha > 0$$

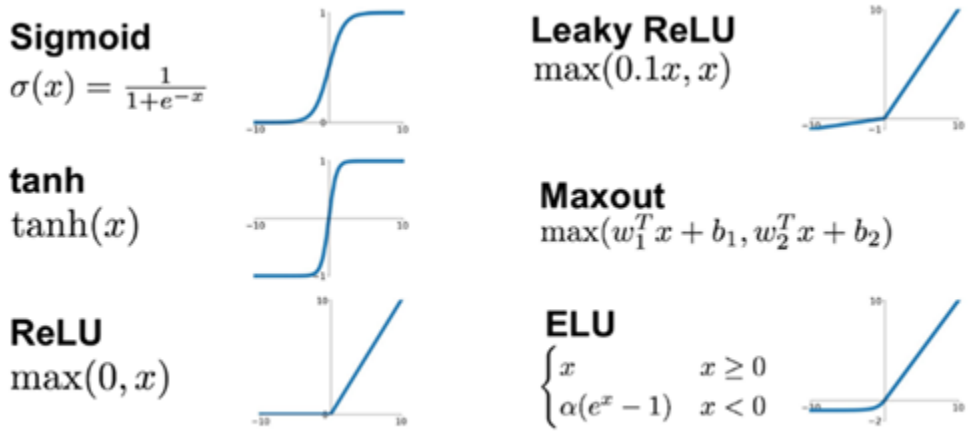

Figura 4: Principales funciones de activación.

Arquitectura de una red neuronal: entrada, capas ocultas, salida.

Una red neuronal artificial está formada por una secuencia ordenada de capas de neuronas interconectadas. Su estructura básica sigue una arquitectura en capas, que se puede dividir en tres tipos principales:

1. **Capa de entrada (input layer):**

Es la capa que recibe los datos originales. Cada neurona de esta capa representa una característica del vector de entrada. Por ejemplo, si el modelo procesa imágenes de 100×100 píxeles en escala de grises, habrá 10 000 entradas, una por cada píxel. Esta capa **no realiza ningún procesamiento**, solo transmite la información hacia adelante.

2. **Capas ocultas (hidden layers):**

Estas son una o más capas intermedias que realizan la transformación de los datos. Cada neurona en una capa oculta realiza lo anteriormente explicado:

- Recibe señales de todas las neuronas de la capa anterior.
- Realiza una combinación lineal (producto de entradas por pesos + sesgo).
- Aplica una función de activación no lineal, como ReLU o sigmoide.

La profundidad de una red (número de capas ocultas) y el número de neuronas por capa son factores clave para su capacidad de aprendizaje. Las redes con muchas capas ocultas se denominan redes profundas (*deep neural networks*), base del deep learning.

3. **Capa de salida (output layer):**

Es la capa final, que produce el resultado del modelo. Su forma depende del tipo de tarea:

- En clasificación binaria, suele consistir en una única neurona con activación sigmoide.
- En clasificación multiclase, tiene tantas neuronas como clases, usando activación softmax.
- En regresión, se emplea una neurona con activación lineal.

Proceso de entrenamiento: propagación hacia adelante y retropropagación.

El entrenamiento de una red neuronal consiste en ajustar sus parámetros internos (pesos y sesgos) para que pueda aprender a predecir salidas correctas a partir de los datos de entrada. Este proceso se basa en dos fases fundamentales: la propagación hacia adelante (forward propagation) y la retropropagación del error (backpropagation), que se repiten durante múltiples iteraciones o épocas.

1. **Propagación hacia adelante (forward propagation)**

Durante esta fase, los datos de entrada se introducen en la red y son transformados capa a capa hasta llegar a la capa de salida [11]. En cada capa:

- Se calcula una combinación lineal de las entradas y los pesos.
- Se aplica una función de activación para obtener la salida de la neurona.
- Esta salida se transmite a la siguiente capa.

El resultado final es una predicción del modelo. Para evaluar su precisión, se compara esta predicción con la etiqueta real utilizando una función de pérdida (*loss function)*, que cuantifica el error. Ejemplos típicos de funciones de pérdida dependiendo del tipo de problema son:

- **Clasificación binaria**:

Se utiliza comúnmente la entropía cruzada binaria (*binary cross-entropy*), que penaliza fuertemente las predicciones incorrectas, especialmente cuando el modelo está demasiado confiado.

- **Regresión**:

Para problemas de predicción de valores continuos, se utiliza habitualmente el error cuadrático medio (MSE):

$$L(y, \bar{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y}_i)^2$$

2. Retropropagación (backpropagation)

Una vez calculado el error, se debe corregir el modelo ya que el objetivo del proceso de optimización es minimizar esa función de pérdida sobre todo el conjunto de entrenamiento, es decir, encontrar los pesos que producen las mejores predicciones posibles. La retropropagación es un procedimiento basado en el cálculo del gradiente de la función de pérdida con respecto a cada uno de los parámetros de la red [12]. Este proceso sigue los siguientes pasos:

- Se aplica la regla de la cadena para derivar el error desde la salida hacia las capas anteriores.
- Se obtienen los gradientes (derivadas parciales) que indican cómo cambiar cada peso para reducir el error.

Una vez calculados los gradientes, se procede a actualizar los pesos y sesgos de la red. Esta actualización se realiza utilizando un algoritmo de optimización, cuyo objetivo es minimizar la función de pérdida en cada iteración del entrenamiento.

El método más básico y ampliamente utilizado es el descenso del gradiente estocástico (SGD) [13]**.** Este algoritmo ajusta cada peso en la dirección opuesta al gradiente del error, de acuerdo con la siguiente regla:

$$w \leftarrow w - \eta \cdot \frac{\delta L}{\delta w}$$

Donde:

$\boldsymbol{\eta}$ es tasa de aprendizaje (*learning rate*), un parámetro clave que controla el tamaño del paso de actualización.

$\frac{\delta L}{\delta w}$ es el gradiente de la función de perdida respecto al peso.

Este cálculo del gradiente de la función de pérdida es un factor que determina el resultado de la red neuronal ya que, dependiendo del problema, surgirán problemas como la sensibilidad a la elección del *learning rate*, o la posibilidad de atascarse en mínimos locales. En la práctica, se utilizan optimizadores más avanzados, algunos ejemplos serían RMSProp, AdaGrad, Adadelta entre otros [14]. Estos optimizadores modifican el paso de aprendizaje adaptándolo al histórico de gradientes, lo que mejora la estabilidad y velocidad de convergencia en muchos problemas reales.

2.1.3. ¿Qué es el *deep learning*?

El *Deep Learning* representa una subcategoría del *machine learning* que emplea redes neuronales con múltiples capas ocultas (de ahí el adjetivo "profundo") para modelar relaciones complejas entre los datos [15]. Aunque las bases matemáticas de las redes neuronales se remontan a los años 80 y 90, su adopción masiva no se consolidó hasta la última década [16], gracias a tres factores clave:

- **Disponibilidad de grandes volúmenes de datos** (Big Data), necesarios para entrenar modelos complejos.
- **Incremento en la capacidad computacional**, especialmente mediante el uso de unidades de procesamiento gráfico (GPU).
- **Avances algorítmicos**, como nuevas funciones de activación (ReLU), normalización por lotes (Batch Normalization), técnicas de regularización (Dropout), y mejores estrategias de inicialización y optimización.

Un hito fundamental fue la victoria del modelo AlexNet en la competición ImageNet de 2012 [17], donde superó con gran diferencia a otros métodos tradicionales de visión por computador. Este modelo, basado en una red convolucional profunda (CNN), demostró el enorme potencial del deep learning para tareas de clasificación de imágenes, marcando el inicio de su expansión en múltiples sectores.

Gracias a su capacidad de generalización y a los avances en computación, el deep learning ha sido adoptado en múltiples sectores, incluyendo:

- **Visión por ordenador:** clasificación de imágenes, detección y segmentación de objetos, reconocimiento facial, inspección industrial, vehículos autónomos.
- **Procesamiento del lenguaje natural (NLP):** traducción automática, generación de texto, asistentes virtuales, análisis de sentimientos.
- **Robótica:** percepción visual, planificación de trayectorias, control adaptativo en entornos dinámicos.
- **Biomedicina y salud:** diagnóstico automatizado por imágenes médicas, predicción de enfermedades, análisis de genómica.
- **Finanzas y predicción de series temporales:** detección de fraude, análisis de riesgo crediticio, previsión de demanda.

Estas aplicaciones comparten la necesidad de modelos capaces de aprender representaciones profundas a partir de grandes volúmenes de datos, donde los enfoques tradicionales de aprendizaje automático suelen ser insuficientes.

2.1.4. Redes neuronales profundas (DNN)

Las redes neuronales profundas (*Deep Neural Networks*, DNN) son una extensión de las redes neuronales artificiales que incluyen múltiples capas ocultas, en lugar de una sola [18]. Esta mayor profundidad permite que el modelo aprenda representaciones jerárquicas cada vez más abstractas de los datos, lo que resulta especialmente útil en tareas como la clasificación de imágenes, el reconocimiento de patrones o la detección de objetos.

Profundidad y capacidad de representación jerárquica.

Una red profunda puede considerarse como una secuencia de funciones compuestas. Cada capa transforma la representación anterior y la entrega a la siguiente. Esto permite a la red:

- En las primeras capas, aprender características locales o simples, como bordes o esquinas.
- En capas intermedias, composiciones más complejas, como formas o texturas.

- En las últimas capas, representaciones semánticas o categorías abstractas (como la presencia o ausencia de defectos de soldadura en una imagen).

Este enfoque jerárquico simula el procesamiento visual del cerebro humano, donde diferentes regiones neuronales analizan distintos niveles de complejidad.

Cuanto más profunda sea una red, mayor será su capacidad para representar funciones complejas. Sin embargo, este aumento de capacidad viene acompañado de nuevos desafíos durante el entrenamiento.

Problemas clásicos: overfitting/underfitting y desvanecimiento del gradiente.

Al aumentar la profundidad, aparecen varios problemas que pueden dificultar o incluso impedir el aprendizaje efectivo [19]:

- **Overfitting (sobreajuste):**

Una red muy compleja puede memorizar los datos de entrenamiento en lugar de aprender patrones generales. Esto da lugar a un excelente rendimiento en el conjunto de entrenamiento, pero un desempeño pobre en datos nuevos (de validación o prueba).

- **Underfitting (subajuste)**

El *underfitting* se presenta cuando un modelo es demasiado simple para capturar la estructura subyacente de los datos. Esto provoca un bajo rendimiento tanto en los datos de entrenamiento como en los de validación, lo que indica que ni siquiera ha aprendido correctamente en el conjunto original.

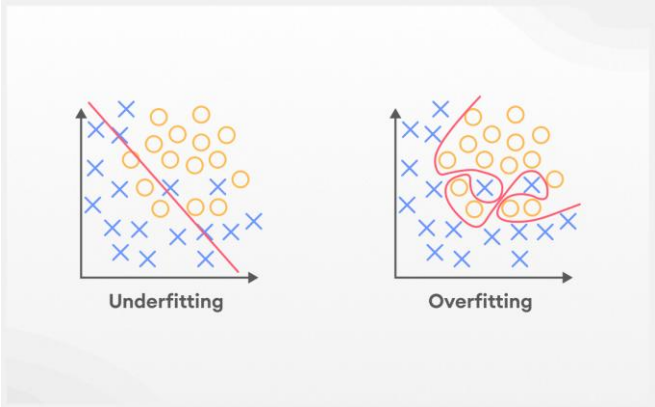

Figura 5: Ejemplo de problemas como overfitting y underfitting

- **Desvanecimiento del gradiente (vanishing gradient):**

Durante la retropropagación del error, los gradientes calculados pueden volverse muy pequeños a medida que se propagan hacia las capas más cercanas a la entrada. Esto hace que los pesos en esas capas apenas se actualicen, bloqueando el aprendizaje. Este fenómeno era particularmente problemático con funciones de activación como la sigmoide o tanh.

Ambos problemas limitaron durante años el uso práctico de redes profundas, hasta la aparición de técnicas que los mitigaron.

Soluciones modernas: normalización, regularización y estrategias de inicialización.

El desarrollo del deep learning ha dado lugar a diversas técnicas que permiten entrenar redes más profundas de manera eficiente y estable [20]:

- **Dropout:**

Técnica de regularización que desconecta aleatoriamente un porcentaje de neuronas durante cada iteración de entrenamiento. Esto fuerza a la red a no depender demasiado de neuronas concretas, lo que mejora la generalización y reduce el overfitting.

- **Batch Normalization:**

Consiste en normalizar las salidas intermedias de cada capa, manteniéndolas con media cero y varianza unitaria. Esto acelera la convergencia y reduce la sensibilidad a la inicialización de pesos.

- **Inicialización adecuada de pesos:**

Métodos como Xavier (Glorot) Initialization o He Initialization ajustan los valores iniciales de los pesos en función del tamaño de las capas. Esto evita que las activaciones exploten o se desvanezcan al inicio del entrenamiento.

- **Optimizadores adaptativos:**

Como se explicó anteriormente, permiten actualizar los parámetros de manera más estable, especialmente en redes profundas.

Estas mejoras, junto con el aumento en la capacidad computacional (GPUs y TPUs), han sido clave para el éxito del deep learning en problemas reales. Arquitecturas como ResNet, EfficientNet o YOLO, empleadas en este trabajo, aprovechan muchas de estas técnicas para permitir redes más profundas, eficientes y precisas.

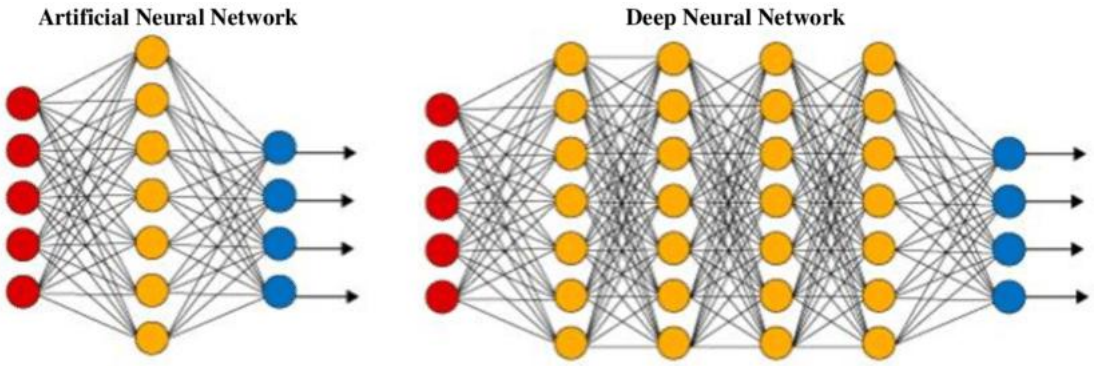

Figura 6: Diferencia entre ANN y DNN

2.1.5. Redes neuronales convolucionales (CNN)

Las redes neuronales convolucionales (*Convolutional Neural Networks*, CNN) son una clase específica de redes diseñadas para procesar datos con estructura espacial, como imágenes o vídeos. Han sido uno de los avances más influyentes en el campo del deep learning, especialmente en tareas de visión artificial, debido a su capacidad para extraer automáticamente características jerárquicas de las imágenes con gran eficiencia y precisión.

Motivación para procesar imágenes.

Las imágenes digitales están compuestas por píxeles organizados en una estructura bidimensional, y los objetos que contienen tienen relaciones espaciales locales (por ejemplo, los bordes, formas y texturas). Las redes neuronales tradicionales (fully connected) no explotan esta estructura, ya que tratan cada píxel como una entrada independiente, lo que resulta ineficiente y propenso al sobreajuste debido al elevado número de parámetros.

Las CNN resuelven este problema introduciendo capas especializadas, que:

- Reducen drásticamente el número de parámetros, aprovechando pesos compartidos.
- Capturan patrones locales mediante filtros conservando la estructura espacial de la imagen.
- Permiten un entrenamiento más rápido y estable en comparación con redes densas profundas sobre imágenes.

Capas principales: convolución, pooling y activación.

Una CNN se compone de distintos tipos de capas, cada una con una función específica:

- **Capa de convolución:**

Es el núcleo de la CNN. Utiliza filtros (también llamados *kernels)* que se deslizan sobre la imagen para detectar patrones locales (bordes, esquinas, texturas). Cada filtro genera un mapa de activación (feature map) que indica la presencia de ese patrón en diferentes zonas de la imagen.

Stride (desplazamiento)

El *stride* indica el número de píxeles que se desplaza el filtro sobre la entrada en cada paso. Un *stride* de 1 significa que el filtro se mueve un píxel a la vez, con solapamiento entre aplicaciones; un *stride* más grande (por ejemplo, 2) reduce la dimensión espacial de la salida, generando mapas de activación más pequeños y disminuyendo la carga computacional.

Padding (relleno de bordes)

El *padding* consiste en añadir píxeles (generalmente de valor cero) alrededor de la entrada antes de aplicar la convolución. Esto ayuda a preservar las dimensiones espaciales tras aplicar filtros (especialmente con *padding* "*same*") y evitar la pérdida de información en los bordes del mapa de activación.

Mapas de activación

Los mapas de activación (también conocidos como *feature maps*) son el resultado directo de aplicar un *kernel* sobre la entrada mediante la operación de convolución. Cada filtro

genera un mapa que representa la detección de un tipo particular de característica en distintas partes de la imagen.

En cada posición del mapa de activación se calcula el producto escalar entre el *kernel* y la región correspondiente de la entrada, seguido por una posible función de activación. Un valor alto en la activación indica que la característica asociada al filtro (por ejemplo, un borde con cierta orientación) está presente en esa región de la imagen.

Básicamente, los mapas de activación permiten a la red CNN identificar qué características se detectan y dónde dentro de la imagen. Además, al ser procesados capa a capa, la red construye representaciones jerárquicas: desde bordes y texturas en capas tempranas hasta formas complejas e incluso objetos en capas profundas.

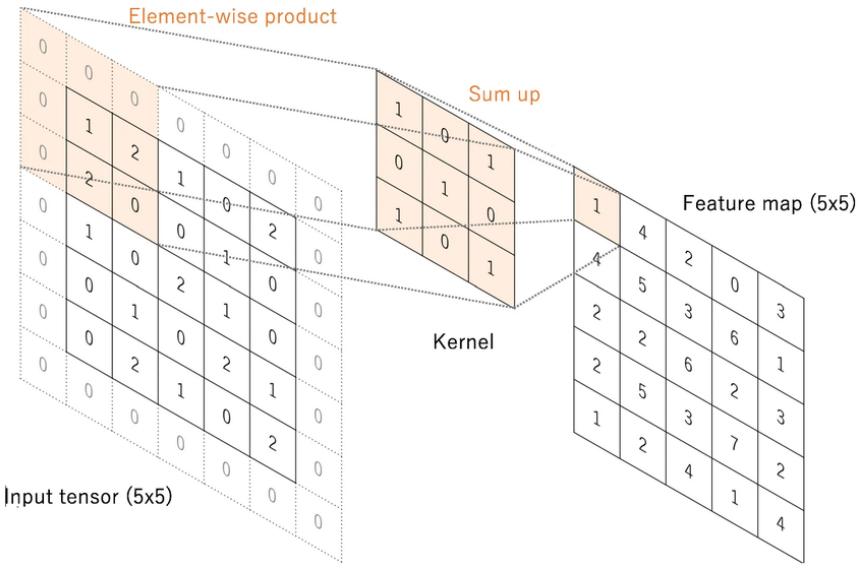

Figura 7: Proceso de convolución para generar un mapa d activación [47]

En la Figura 7 se puede observar el proceso de convolución para generar un mapa de activación a partir de un tensor de entrada usando un *kernel*:

- La entrada es una tensor 2D de 5x5, que podría representar una región de una imagen.
- El *kernel*, es una matriz de 3x3 con valores propios que han sido definidos manualmente o aprendidos.
- El resultado final es un *feature map* de 5x5, típicamente sería de dimensiones reducidas si no se hubiera utilizado *padding* como se observa.

Técnicamente, esta operación de convolución se representa mediante la operación de un *kernel* (3x3) con *stride* = 1 (desplazamiento de un pixel a la derecha) y *padding* = 1 (una capa de píxel en el exterior).

- **Capa de activación:**

Tras la convolución, se aplica una función de activación no lineal, como ReLU, para introducir no linealidad y permitir a la red aprender funciones complejas. Esta activación se aplica de forma elemento a elemento al mapa de activación.

- **Capa de pooling (submuestreo):**

Reduce la dimensionalidad espacial de los mapas de activación mediante operaciones como *max pooling* o *average pooling*. Esto disminuye el coste computacional y proporciona invariancia ante pequeñas translaciones o distorsiones en la imagen.

Las CNN suelen estar compuestas por varias combinaciones de capas convolucionales + activación + pooling, seguidas al final por una o más capas densas (fully connected) que realizan la clasificación o predicción.

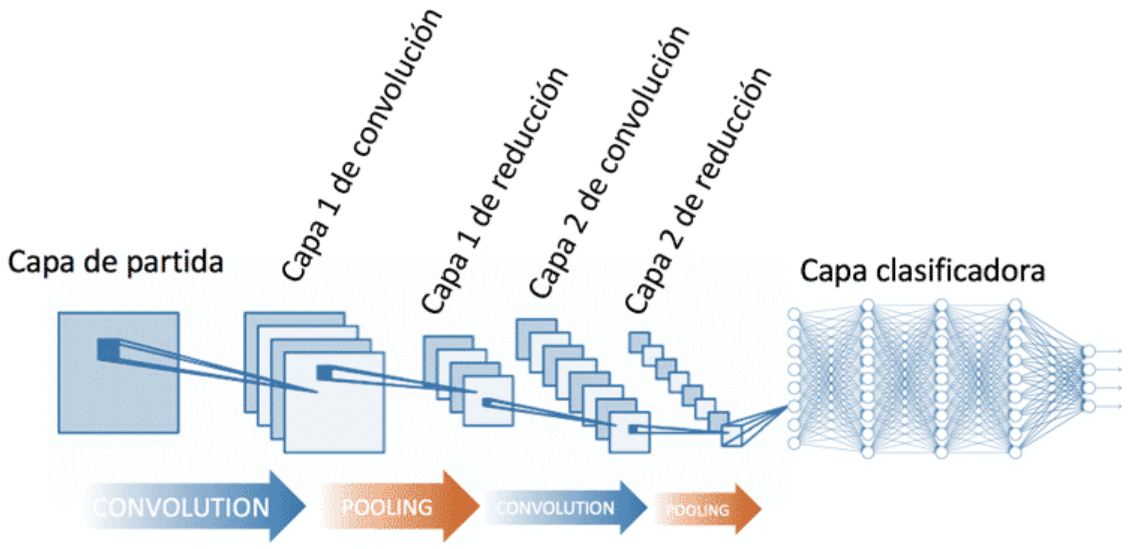

Figura 8: Arquitectura de una red neuronal convolucional CNN

2.2. Visión artificial

La visión artificial constituye uno de los campos de aplicación más relevantes y transformadores del *deep learning* en la actualidad. Al combinar técnicas de redes neuronales profundas con procesamiento de imágenes, se ha logrado automatizar tareas visuales que antes requerían intervención humana, como la inspección de calidad, la clasificación de piezas o la detección de defectos en entornos industriales.

En esta sección se introduce el concepto de visión artificial, sus aplicaciones más destacadas en la industria, las tareas que habitualmente aborda y su importancia específica en procesos de inspección automatizada, como el caso planteado en este proyecto.

2.2.1. Definición y papel dentro del Deep Learning

La visión artificial (*computer vision*) es una rama de la inteligencia artificial cuyo objetivo es dotar a las máquinas de la capacidad de interpretar y extraer información de imágenes o secuencias de vídeo. Esta disciplina abarca muchas tareas, desde la detección de bordes y formas hasta el reconocimiento facial y la segmentación semántica.

Históricamente, la visión artificial se sustentaba en técnicas de extracción manual de características: algoritmos específicamente diseñados para identificar patrones como bordes, texturas o esquinas, sin utilizar aprendizaje. Esto implicaba una fuerte dependencia del conocimiento experto y limitaba la capacidad de adaptación de los sistemas a nuevos escenarios

En contraste, el contexto del *deep learning* ha provocado una auténtica revolución. El uso de redes neuronales profundas, especialmente las CNN, ha permitido que los modelos aprendan automáticamente las representaciones visuales relevantes directamente a partir de los datos, suprimiendo la necesidad de diseñar manualmente los descriptores. Como resultado, la visión artificial se ha transformado en un proceso altamente automatizado, robusto y escalable, capaz de igualar o superar el rendimiento humano en muchas aplicaciones. Especialmente, el rol del *deep learning* en visión artificial trae los siguientes puntos:

- **Aprendizaje de características jerárquicas:**

Las CNN extraen automáticamente de forma progresiva información cada vez más abstracta, desde bordes hasta objetos complejos.

- **Incremento de rendimiento:**

Sistemas basados en *deep learning* han superado a los métodos tradicionales en tareas de clasificación, detección y segmentación.

- **Escalabilidad industrial:**

La combinación de grandes volúmenes de datos, potentes GPUs y técnicas avanzadas de entrenamiento ha habilitado sistemas capaces de trabajar en tiempo real bajo condiciones industriales exigentes, lo que abre nuevas posibilidades en producción automatizada.

2.2.2. Aplicaciones industriales

La visión artificial ha adquirido un rol fundamental en la Industria 4.0, siendo uno de los ejes clave en la automatización y mejora del control de calidad. A continuación, se presentan sus principales aplicaciones industriales:

1. **Inspección de calidad y detección de defectos**

Los sistemas basados en *deep learning* permiten automatizar inspecciones visuales, logrando identificar defectos con mayor precisión, velocidad y consistencia que la inspección manual.

En la industria alimentaria y agrícola, los sistemas de *sorting* óptico identifican y eliminan productos defectuosos (frutas, tubérculos, etc.) a altas velocidades.

2. **Robótica guiada por visión (VGR)**

Robots móviles equipados con cámaras detectan piezas en posiciones variables, lo que permite automatizar el pick-and-place sin necesidad de colocarlas de forma precisa. Esta tecnología reduce costes en utillaje mecánico y mejora la flexibilidad y adaptabilidad de producción.

3. **Inspección en soldadura y monitorización de adhesiones**

Cámaras especiales registran el proceso de soldadura en tiempo real para detectar defectos como falta de fusión, porosidad o salpicaduras.

4. **Mantenimiento predictivo y análisis de maquinaria**

Sistemas de visión observan condiciones de desgaste, corrosión o problemas en piezas de maquinaria, permitiendo intervenciones preventivas antes de fallos críticos.

2.2.3. Tareas comunes en visión artificial

En el marco del *deep learning* aplicado a visión artificial, existen tres tareas primordiales que se repiten en la mayoría de los sistemas [21]:

1. **Clasificación de imágenes (Image Classification)**

Consiste en asignar una etiqueta a una imagen completa. Por ejemplo, determinar si una tapa de batería está "correcta" o "defectuosa". Esta tarea se modela como un problema de clasificación, donde el modelo aprende a identificar patrones globales en la imagen.

2. **Detección de objetos (*Object Detection*)**

Va un paso más allá, ya que requiere localizar y clasificar múltiples instancias de objetos dentro de una imagen, generalmente mediante cajas delimitadoras (*bounding boxes)*. Es fundamental en entornos industriales para identificar defectos puntuales, restos de soldadura o componentes incorrectamente colocados.

3. **Segmentación (*Image Segmentation*)**

Asigna una etiqueta a nivel de píxel, lo que permite obtener un contorno preciso del objeto o defecto. Se utiliza en contextos donde se necesita una detección más detallada, como identificar la forma exacta de un defecto de soldadura. Existen dos variantes principales:

- Segmentación semántica: agrupa píxeles según su clase (por ejemplo: "soldadura" vs "metal").
- Segmentación de instancias: distingue instancias individuales dentro de una misma clase.

Tarea	*Resultado Típico*	*Complejidad*	*Aplicación Industrial*
Clasificación	Etiqueta de clase por imagen	Baja	Filtrado rápido
Detección	*Bounding box* + etiqueta	Media	Identificación y ubicación de defectos
Segmentación	Máscara de píxeles detallada	Alta	Análisis preciso de forma y tamaño

Tabla 1: Comparación de tareas según su complejidad y aplicación

2.3. Modelos empleados para el proyecto

Tras haber establecido los fundamentos del aprendizaje profundo y de las redes convolucionales, en esta sección se presentan las arquitecturas concretas utilizadas en el desarrollo del sistema de detección de defectos: ResNet, EfficientNet y YOLO.

Estas redes han sido seleccionadas por su consolidado rendimiento en tareas de clasificación y detección de objetos en imágenes, y por su capacidad para adaptarse a distintos requisitos de precisión, eficiencia y complejidad computacional. Se describen a continuación sus principales características, ventajas técnicas y el motivo de su elección para abordar el problema planteado.

2.3.1. ResNet

ResNet, o también llamada *Residual Networks* es una arquitectura de red neuronal profunda introducida por He et al. (2015) en su influyente *paper* "Deep Residual Learning for Image Recognition"**Erro! Fonte de referência não encontrada.** [22]**Erro! Fonte de referência não encontrada.**, que presentó la idea innovadora de las conexiones residuales (*skip connections*).

Cuando las redes se vuelven muy profundas, sufren del problema del gradiente desvanecido (*vanishing gradient*), donde añadir más capas empeora la capacidad de entrenamiento. ResNet aborda estas limitaciones mediante funciones residuales:

$$F(x) = H(x) - x$$

En lugar de aprender directamente $H(x)$, las conexiones residuales permiten que una red aprenda la función residual $F(x)$, reformulando la salida de cada bloque como:

$$y(x) = F(x) + x$$

Esto facilita el aprendizaje de funciones cercanas a la identidad y mejora el flujo del gradiente durante el entrenamiento.

Principio de funcionamiento

Los bloques residuales suelen consistir en dos o tres capas convolucionales, seguidas de *batch normalization* y activaciones como ReLU. La conexión de salto suma la entrada x directamente a la salida del bloque, sin incrementar la complejidad computacional. Esta arquitectura permite entrenar modelos con más de 50 o 100 capas (ResNet-50, ResNet-101, ResNet-152) sin sufrir el fenómeno del gradiente desvanecido.

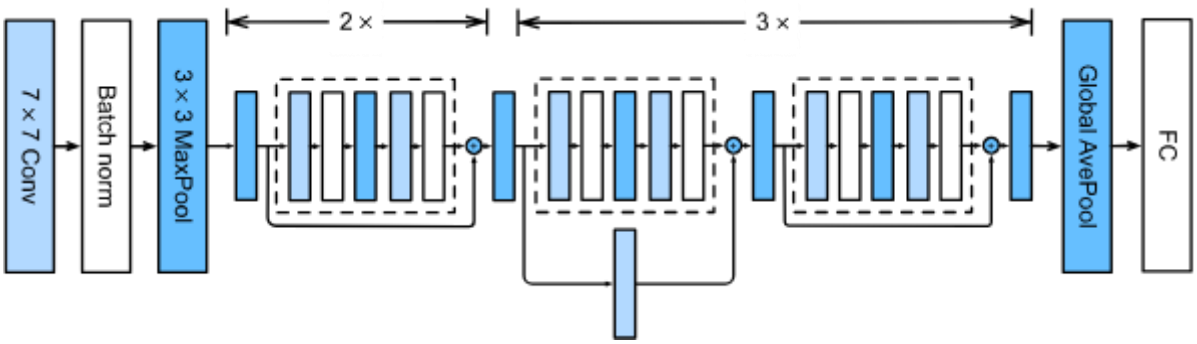

Figura 9: Arquitectura ResNet ***Erro! Fonte de referência não encontrada.***

En la arquitectura de ejemplo, Figura 9, se pueden observar los siguientes componentes clave:

1. **Convolución inicial**

La red comienza con una capa convolucional de tamaño de filtro 7x7, que procesará la imagen de entrada para extraer características de bajo nivel, tales como bordes, texturas y patrones básicos

2. **Submuestreo con Max Pooling 3x3**

A continuación, se aplica una capa de max pooling con ventana 3x3 que actúa reduciendo la dimensión espacial del volumen de activación. El propósito es condensar la información más significativa, reducir la cantidad de parámetros y ganar invariancia frente a desplazamientos menores.

3. **Bloques residuales**

La arquitectura se organiza en etapas con repetición de bloques residuales: por ejemplo, 2 bloques en la primera etapa y 3 en la siguiente, tal como indica el diagrama del 2x y 3x.

4. **Global Average Pooling**

Finalmente, se observa una capa de *Global Average Pooling* al final del bloque convolucional. Esta capa reduce cada mapa de activación completo (por ejemplo, de tamaño 7x7) a un único valor promedio por canal. Su principal ventaja es eliminar capas totalmente conectadas densas, lo que disminuye el número de parámetros y reduce el riesgo de sobreajuste

Ventajas

- Facilidad de entrenamiento: optimizar redes profundas es posible gracias al flujo directo del gradiente.
- Generalización robusta: menos propensas al overfitting, gracias al aprendizaje jerárquico eficiente.
- Capacidad jerárquica: extraen representaciones visuales desde bajo nivel (bordes, texturas) hasta alto nivel (formas complejas), útiles en clasificación detallada de defectos.

2.3.2. EfficientNet

EfficientNet, diseñado por Mingxing Tan y Quoc V. Le (2019) en el trabajo "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks" [23], propone una estrategia novedosa para escalar redes convolucionales de manera eficiente.

Principio de funcionamiento

La innovación clave de EfficientNet es el *compound scaling*, una metodología que escala de forma simultánea y equilibrada tres dimensiones del modelo: profundidad (número de capas), anchura (canales por capa) y resolución de entrada (tamaño de la imagen), usando un único coeficiente φ. Se define de la siguiente manera:

$$depth = \alpha^{\varphi}, \qquad width = \beta^{\varphi}, \qquad resolution = \gamma^{\varphi}$$

Con la restricción:

$$\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$

Este enfoque se determinó mediante búsqueda de cuadrícula para lograr un equilibrio óptimo entre precisión y eficiencia.

Esta estrategia surge de lo siguiente: Cuando aumentas la resolución de una imagen de entrada (por ejemplo, de 224×224 a 300×300), aparece la necesidad de:

- Más capas para capturar patrones y contextos de mayor campo receptivo (visión de objetos más complejos).
- Más canales para detectar detalles más finos presentes en la imagen ampliada

Simplemente aumentar una dimensión (como solo la profundidad) no genera mejoras sostenibles, e incluso puede disminuir el rendimiento. EfficientNet equilibra todos los parámetros para que cada dimensión crezca de forma lógica y proporcional.

Figura 10: Arquitectura EfficientNet [64]

La Figura 10 muestra la estructura básica del modelo EfficientNet-B0[65], compuesto por siete bloques secuenciales que transforman la imagen de entrada hasta generar un *feature map*.

1. **Capa inicial**

Se trata de una capa convolucional 3x3 que reduce la resolución espacial y amplía canales, preparando la entrada para extracción de características.

2. **Bloques MBConv (1 a 7)**

Compuesto por siete bloques secuenciales de MBConv (*Mobile Inverted Bottleneck Convolutions*)[24], diseñados para extraer progresivamente información visual con alta eficiencia y jerarquía, sin inflar excesivamente la carga computacional. Cada bloque sigue el siguiente flujo:

- **Expansion**

Aumenta el número de canales (normalmente x6) para enriquecer la representación.

- **Convolución depthwise (kernel 3×3 o 5×5)**

Aplica filtros independientes por canal, lo que reduce significativamente la carga computacional.

- **Bloque Squeeze-and-Excitation (SE)**

Recalibra la importancia de cada canal según su relevancia mediante pooling global y mecanismos de atención.

En resumen, en los primeros bloques se conservan resoluciones más grandes y se comienza a extraer características simples. Los bloques medios introducen kernels más grandes (5x5). Por último, en los bloques finales, las dimensiones espaciales son reducidas, y se refinan las representaciones de alto nivel.

Ventajas

- Alta precisión con bajo coste computacional.
- Menor riesgo de overfitting: debido al diseño equilibrado y la regularización integrada, es robusto frente a conjuntos de datos limitados

2.3.3. YOLO

YOLO (You Only Look Once), presentado por Joseph Redmon et al. en 2016 [25], es una arquitectura de detección de objetos en una sola pasada ("*single-stage*") que transformó el campo de la visión artificial, permitiendo detectar y clasificar objetos directamente desde la imagen completa en un solo paso de inferencia.

YOLO redefine la detección de objetos como un problema de regresión: en lugar de aplicar clasificadores sobre regiones propuestas, una única CNN predice simultáneamente todas las cajas delimitadoras (*bounding boxes*) y las probabilidades de clase para cada región, optimizando directamente el rendimiento global de detección.

Principio de funcionamiento

1. **División de la imagen en cuadrícula:**

La imagen se divide en una retícula de SxS celdas. Cada celda es responsable de predecir varias cajas de detección que caen dentro de ella, así como las correspondientes probabilidades de clase.

2. **Predicción en una sola pasada:**

A partir de una única evaluación de la red, YOLO genera coordenadas de cajas (centro x, y, ancho, alto), un *score* de confianza por cada caja y las probabilidades de clase. Todo se

predice en una sola salida tensorial fija, lo que facilita el uso de GPU y la eficiencia computacional.

3. **Pérdida unificada:**

El modelo minimiza una función de pérdida global que combina errores de localización (regresión de cuadros), clasificación y puntuación de confianza, entrenando el sistema de forma integrada y exigente.

4. **Postprocesado:**

Tras la inferencia, se una serie de técnicas para eliminar cajas redundantes y mantener las detecciones más fiables.

Ventajas

- Velocidad en tiempo real: la versión original de YOLO procesa imágenes a aproximadamente 45 fps, y la variante más ligera (Fast YOLO) puede alcanzar hasta 155 fps, manteniendo un rendimiento competitivo.
- Modelo unificado y end-to-end: entrena como una única red, lo que simplifica la optimización y reduce errores.
- Eficiencia en GPUs: su arquitectura totalmente convolucional es altamente paralelizable y optimiza el uso de recursos para inferencias rápidas.

Capítulo 3. Estado de la cuestión

Antes de desarrollar cualquier sistema de visión artificial, especialmente uno destinado a tareas críticas como el control de calidad en la industria automotriz, resulta imprescindible analizar el contexto tecnológico y científico existente. Este capítulo tiene como finalidad revisar las soluciones previas, tanto académicas como comerciales, que abordan problemas similares al planteado en este proyecto: la detección automática de defectos de soldadura en componentes metálicos mediante técnicas de *deep learning*.

En particular, se explorarán trabajos relevantes relacionados con la inspección visual automatizada en entornos industriales, haciendo especial énfasis en aquellas aplicaciones que utilizan redes neuronales convolucionales (CNN), arquitecturas como YOLO o ResNet, y técnicas de clasificación de defectos superficiales en piezas metálicas.

Además, se analizarán herramientas comerciales que ofrecen soluciones de visión industrial integradas y se identificarán los aspectos en los que el presente proyecto propone una mejora o contribución original. Esta revisión sentará las bases para justificar la necesidad y pertinencia del desarrollo realizado.

3.1. Detección de defectos adhesivos y soldadura

Uno de los ámbitos donde se ha explorado con mayor intensidad el *deep learning* es en la inspección de adhesivos y puntos de soldadura. Wang et al. [26] desarrollaron un sistema de detección de defectos adhesivos utilizando YOLOv8 como arquitectura base. El modelo fue mejorado mediante la incorporación de un mecanismo de atención denominado *skip squeeze* and *excitation*, que permite resaltar características relevantes en regiones críticas de la imagen. Además, introdujeron una función de pérdida modificada (WIoU), especialmente diseñada para mejorar la localización de defectos pequeños y difíciles de detectar. Sus resultados muestran que la combinación de estas modificaciones no solo mejora la precisión de detección, sino que también mantiene una alta velocidad de inferencia, lo que lo hace apto para su uso en líneas de producción en tiempo real.

En la misma línea, Dai et al. [27] propusieron una versión modificada de YOLOv3 para inspeccionar puntos de soldadura en baterías. El objetivo era identificar soldaduras pequeñas y determinar si eran correctas o defectuosas. Los autores compararon el rendimiento de YOLO con el de Faster R-CNN, observando que, aunque este último alcanzó una ligeramente mayor precisión, el modelo YOLO resultó significativamente más rápido, reduciendo los tiempos de inferencia y haciéndolo más adecuado para aplicaciones industriales donde la eficiencia es prioritaria. Estos estudios ponen de manifiesto que las arquitecturas basadas en YOLO, en sus distintas variantes, ofrecen un equilibrio sólido entre precisión y velocidad, lo cual es un factor clave en la inspección automática de procesos de soldadura.

3.2. Inspección de piezas y componentes

Otro conjunto de trabajos relevantes se centra en la detección de defectos o ausencia de componentes en diferentes piezas automotrices. Hachem et al. [28] plantearon el uso de ResNet-50 para sustituir la inspección visual manual en una planta automotriz. Su sistema estaba orientado a verificar la presencia o ausencia de tornillos en piezas metálicas. Gracias a la capacidad de ResNet para capturar representaciones jerárquicas de los datos visuales, el modelo alcanzó una precisión del 99%, superando la variabilidad y errores propios de la inspección humana. Este resultado valida el uso de redes profundas preentrenadas en tareas de clasificación binaria dentro de entornos industriales.

De forma complementaria, Muresan et al. [29] desarrollaron un modelo basado en LeNet-5, una arquitectura más ligera, con el objetivo de clasificar casquillos en piezas plásticas inyectadas. El sistema identificaba si los casquillos estaban correctamente colocados, mal posicionados o ausentes. A pesar de tratarse de una red relativamente simple, lograron entre 98% y 99% de precisión, incluso bajo condiciones variables de iluminación y cámara. Esto demuestra que, en ciertos escenarios, arquitecturas más compactas pueden ser igualmente efectivas siempre que se ajusten adecuadamente a la naturaleza del problema.

Ambos estudios reflejan cómo distintos niveles de complejidad arquitectónica (desde LeNet hasta ResNet) pueden ser aplicados con éxito dependiendo del balance necesario

entre capacidad de generalización, recursos computacionales y condiciones de adquisición de imágenes.

3.3. Soluciones comerciales en inspección automotriz

Además de los avances académicos, existen numerosas soluciones comerciales consolidadas que utilizan visión artificial para garantizar la calidad en entornos industriales. A continuación, se presentan algunas destacadas:

Ford - AiTriz y MAIVS

Ford ha desplegado dos sistemas internos de visión con IA para apoyar la inspección de calidad en planta: AiTriz y MAIVS. Ambos se diseñan como "asistentes" para el operario (no como sustitutos) y se integran con el sistema de control de producción para verificar que el vehículo que llega al puesto recibe exactamente las piezas y conexiones definidas en su "receta" de fabricación. El objetivo declarado es detectar errores milimétricos de montaje in situ, antes de que deriven en reprocesos o, en el peor de los casos, en campañas de retirada (recalls). Ford ha sido explícita en que estas herramientas se están usando ya a escala en sus plantas norteamericanas [30].

En primer lugar, MAIVS es un sistema basado en imágenes fijas capturadas por teléfonos inteligentes montados en soportes impresos en 3D en puntos de la línea. Con estas tomas, la IA verifica la presencia, posición y orientación correctas de elementos visibles, por ejemplo, emblemas o molduras y la correcta instalación de subconjuntos. Su fortaleza radica en la rapidez de despliegue (coste bajo del hardware, flexibilidad de colocación) y en la homogeneidad de la comprobación visual respecto a la inspección manual. Ford ha comunicado su uso en múltiples estaciones de inspección, con implantación en decenas de puestos en sus plantas norteamericanas; reportes sectoriales añaden que se utiliza de forma amplia en diversos centros de producción.

Por otro lado, AiTriz es un sistema basado en flujo de vídeo que aplica aprendizaje automático para detectar desalineaciones de escala milimétrica, por ejemplo, conectores eléctricos no completamente insertados, incluso en condiciones desafiantes (iluminación variable, oclusiones temporales cuando un operario cruza el campo de visión, o partes de chapa interponiéndose momentáneamente). A diferencia de MAIVS, AiTriz ofrece una

mayor granularidad temporal y precisión gracias a la secuencia de vídeo, lo que le permite confirmar la correcta finalización de operaciones que pueden quedar parcialmente ocultas en una imagen fija. Ford sitúa su despliegue inicial en la planta de Dearborn (EE. UU.) y describe que ya opera en decenas de estaciones dentro de su red industrial.

Cognex - VisionPro

VisionPro es una plataforma de software de visión artificial desarrollada por Cognex [31], diseñada para abordar aplicaciones industriales complejas en entornos de manufactura. Permite la creación mediante programación o interfaz gráfica de flujos de inspección mixtos que incluyen capacidades impulsadas por *deep learning*.

VisionPro está preparado para operar con una amplia gama de cámaras industriales y sistemas de adquisición (visión 2D, line scan y 3D), así como integrarse con sistemas de iluminación y lentes especializados. Su entorno de desarrollo "QuickBuild" permite idear y encadenar bloques de procesamiento visual de forma modular, facilitando el prototipado rápido: herramientas como Caliper (medición), Geometry, OCR, pattern matching, y módulos de *deep learning* como *Blue Locate*, *Red Analyze* y *Green Classify*, se combinan para resolver tareas desde inspección de piezas hasta clasificación y lectura avanzada

En síntesis, los trabajos revisados demuestran que arquitecturas como YOLO son especialmente eficaces para la detección de defectos pequeños en entornos industriales gracias a su equilibrio entre velocidad y precisión, mientras que modelos como ResNet o EfficientNet ofrecen gran rendimiento en clasificación binaria de componentes. A nivel comercial, soluciones como VisionPro o los sistemas AiTriz y MAIVS de Ford muestran cómo la visión artificial se integra ya de forma práctica en líneas de producción, principalmente como apoyo al operario. Este proyecto se alinea con estas tendencias, aportando un estudio comparativo de diferentes arquitecturas aplicadas a un caso industrial concreto, la inspección de tapas de batería, con el fin de identificar el modelo más adecuado en términos de precisión, eficiencia y aplicabilidad real.

Capítulo 4. Sistema desarrollado

Este capítulo describe en detalle el proceso técnico seguido para desarrollar el sistema de visión artificial orientado a la detección automática de defectos de soldadura en tapas de baterías para automóviles. A partir del problema planteado por la empresa Gestamp, se diseñó un flujo de trabajo que abarca desde la adquisición y procesamiento de imágenes hasta el entrenamiento y evaluación de modelos de aprendizaje profundo.

A diferencia de capítulos anteriores, que se centraron en los fundamentos teóricos y la definición formal del sistema, aquí se documentan las decisiones prácticas adoptadas, los algoritmos implementados y la secuencia real de tareas llevadas a cabo para obtener un sistema funcional.

El desarrollo del sistema siguió un enfoque iterativo y modular, lo que permitió depurar y validar cada componente de manera progresiva. A lo largo de este capítulo se detallan las distintas fases que conformaron el flujo de trabajo, desde la gestión inicial de los datos hasta el entrenamiento y evaluación de los modelos. En cada sección se justifican las decisiones técnicas adoptadas, se describen las herramientas y métodos empleados, y se destacan los principales retos y aprendizajes surgidos durante el proceso.

4.1. Alcance del sistema

Tal como se ha descrito en capítulos anteriores, el proyecto aborda un caso concreto de inspección automatizada en el contexto de control de calidad industrial: la detección de fallos de soldadura en las arandelas de los agujeros de tapas de baterías para automóviles.

El sistema desarrollado se plantea como una solución de visión artificial basada en técnicas de *deep learning*, capaz de procesar imágenes capturadas por seis cámaras industriales, identificar y clasificar automáticamente cada agujero como “ok” o “nok”, y tomar decisiones sobre la idoneidad de la tapa en su conjunto.

El alcance funcional del sistema incluye:

- Segmentación automática de agujeros a partir de imágenes de cámara.
- Clasificación binaria de cada agujero.

- Rechazo automático de tapas con al menos un defecto detectado.
- Registro de resultados por tapa para su posterior análisis y trazabilidad.

Todo el flujo, desde la adquisición hasta la evaluación, ha sido diseñado para integrarse potencialmente en un entorno de producción industrial, respetando criterios de precisión, eficiencia y escalabilidad.

4.2. Adquisición y organización de datos

La base del desarrollo del sistema fue la obtención de un conjunto de imágenes reales capturadas directamente en el entorno de producción de Gestamp. Estas imágenes correspondían a tapas de baterías de automóvil y fueron proporcionadas en formato digital, organizadas por lotes y etiquetadas por fecha de captura.

Cada tapa era registrada mediante un sistema compuesto por seis cámaras industriales, cada una enfocada a una región concreta de la pieza. Esta disposición permitía cubrir de manera exhaustiva todas las áreas críticas en las que se encuentran los orificios con arandelas soldadas. Cada cámara capturaba una imagen de alta resolución centrada en un grupo de aproximadamente cinco a siete agujeros.

Estas imágenes iniciales constituyeron el punto de partida del flujo de trabajo. A partir de ellas, fue necesario desarrollar mecanismos para aislar, identificar y estructurar de forma coherente cada agujero como una unidad de análisis independiente. Para ello, se definió una convención de nomenclatura que permitiera vincular cada imagen con su correspondiente tapa, cámara y posición del agujero, lo que facilitó su trazabilidad y posterior clasificación.

En resumen, el proceso de adquisición de datos se caracterizó por:

- El uso de imágenes reales en condiciones industriales representativas.
- Una estructura de datos jerárquica basada en: tapa → cámara → agujero.
- La necesidad de aplicar un preprocesamiento inicial para segmentar y normalizar las regiones de interés (Subsección 4.3)

Este conjunto de imágenes permitió trabajar sobre un caso industrial auténtico y representativo, garantizando así la relevancia práctica del sistema desarrollado y su aplicabilidad directa en entornos reales.

La Figura 11 muestra cómo se distribuyen las 6 diferentes cámaras para capturar la tapa de batería al completo.

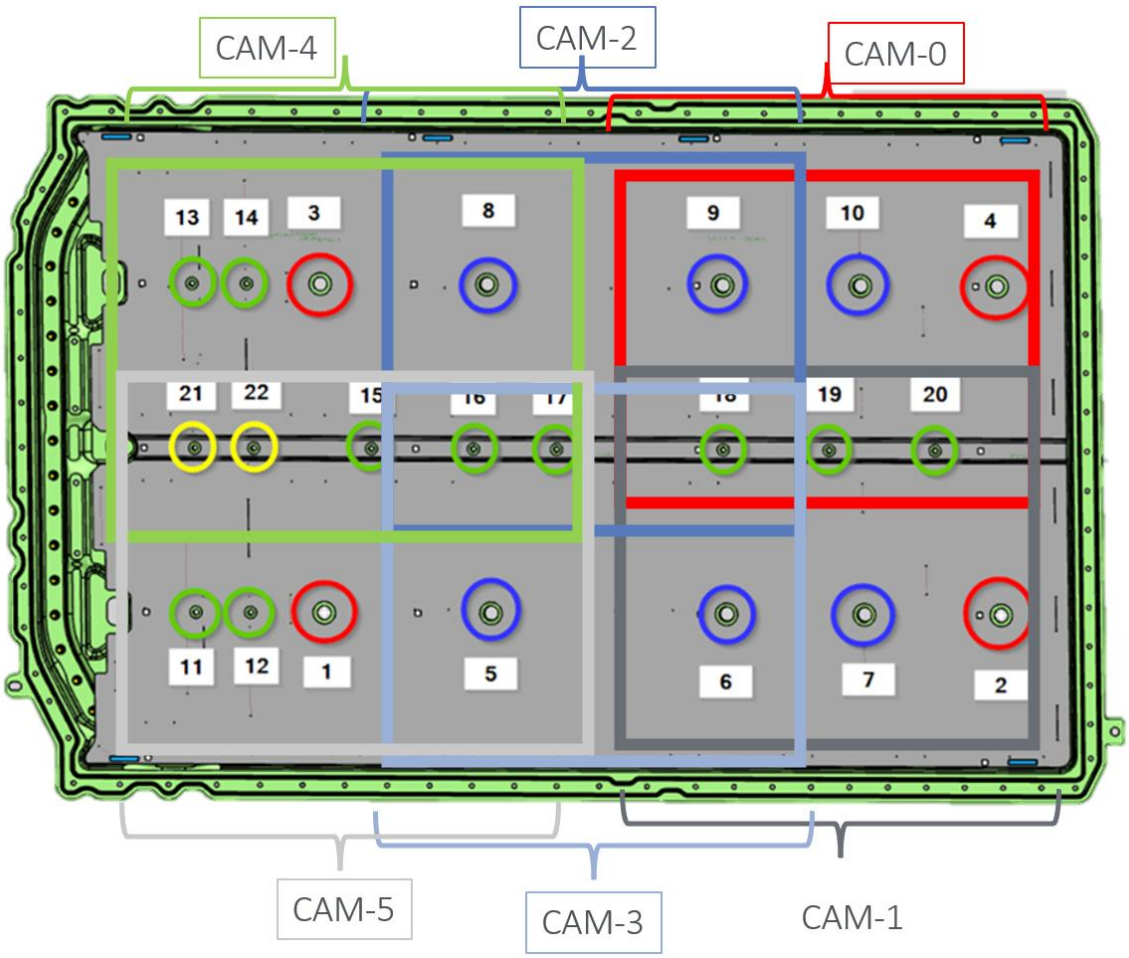

Figura 11: Distribución de las cámaras

4.3. Segmentación y *croppeo* automático de agujeros

Una vez adquiridas las imágenes de las seis cámaras correspondientes a cada tapa, el siguiente paso fue aislar visualmente cada uno de los agujeros presentes en las regiones inspeccionadas. Para ello, se diseñó un algoritmo de segmentación automática, cuyo

objetivo principal era detectar, recortar (*croppear*) y numerar cada agujero individual de manera coherente y reproducible.

Este proceso fue fundamental para transformar cada imagen global en una serie de instancias unitarias (una por agujero), que luego serían utilizadas como entradas en los modelos de clasificación o detección.

4.3.1. Necesidad de automatización

Aunque el número de cámaras y la disposición de los agujeros eran relativamente constantes, el volumen de datos y la necesidad de consistencia hacían inviable realizar esta tarea manualmente. Un algoritmo automatizado permitía:

- Estandarizar el tamaño y la posición de los recortes.
- Garantizar la correcta asociación entre agujeros, cámaras y tapas.
- Reducir errores humanos y mejorar la escalabilidad del sistema.

4.3.2. Algoritmo de detección y recorte

Para obtener imágenes individuales de cada agujero a partir de las imágenes completas capturadas por las cámaras industriales, se desarrolló un algoritmo personalizado de detección y segmentación automática. Este proceso tenía como finalidad generar un conjunto de recortes homogéneos, centrados en cada zona de interés, que sirvieran como entrada a los modelos de clasificación y detección.

El algoritmo parte de imágenes en formato RGB y realiza una serie de transformaciones para detectar las regiones oscuras que corresponden a los agujeros. Primero se aplica una normalización del brillo para corregir desigualdades de iluminación, seguida de una conversión a escala de grises y una umbralización inversa que permite aislar las zonas más oscuras de la imagen.

Una vez segmentadas, se extraen los contornos relevantes y se filtran geométricamente para descartar formas no circulares o ruidos. Los centros de los agujeros se calculan a partir de los momentos de los contornos, y se emplea una técnica de fusión para eliminar duplicidades causadas por detecciones múltiples de un mismo agujero.

Posteriormente, las coordenadas obtenidas se ordenan de forma sistemática (de arriba a abajo y de izquierda a derecha), y se generan recortes centrados en cada agujero, todos de tamaño uniforme. Este procedimiento asegura una consistencia espacial que resulta fundamental tanto para el etiquetado como para el entrenamiento de los modelos.

El algoritmo permitió automatizar un paso que, de hacerse manualmente, habría supuesto una carga de trabajo considerable. La precisión del método fue validada visualmente y los recortes generados demostraron una calidad suficiente para alimentar de forma robusta los siguientes pasos del entramiento de los modelos

4.4. Aumento de datos (Data Augmentation)

Uno de los retos fundamentales a la hora de entrenar modelos de aprendizaje profundo en entornos industriales es la disponibilidad de datos etiquetados y equilibrados. En este proyecto, las imágenes originales presentaban un fuerte desbalance entre las clases: los agujeros etiquetados como “ok” eran significativamente más numerosos que los clasificados como “nok”, lo que podía inducir un sesgo en los modelos y comprometer su capacidad de generalización.

Para mitigar este problema y mejorar la robustez del aprendizaje, se aplicaron técnicas de *data augmentation* sobre el conjunto de imágenes minoritarias. El objetivo de esta etapa era generar artificialmente nuevas muestras mediante transformaciones visuales realistas, preservando la semántica de la clase (es decir, sin alterar la condición de defecto del agujero).

El flujo de proceso de esta aumentación se diseñó y se compuso de las siguientes transformaciones aplicadas de forma aleatoria:

- **Rotaciones suaves** (entre -2° y 2°): para simular ligeras variaciones de orientación.
- **Ajustes de brillo**: mediante multiplicación de intensidad (rango 0.9 a 1.1).
- **Modificaciones de contraste lineal**: con factores de compresión del histograma.

- **Afilado suave**: que incrementa la definición de los bordes sin generar artefactos.
- **Desenfoque gaussiano leve**: para simular pequeñas pérdidas de enfoque.

Cada imagen de la clase minoritaria se sometió a múltiples variaciones controladas, generando así dos nuevas versiones por cada muestra original. Estas imágenes aumentadas se almacenaron con un identificador específico, manteniendo la trazabilidad respecto al archivo original.

Este proceso permitió:

- Equilibrar de forma aproximada el número de ejemplos por clase.
- Introducir variabilidad visual para mejorar la generalización.
- Reducir el riesgo de overfitting, especialmente en arquitecturas profundas.

La técnica empleada es común en tareas de clasificación de imágenes industriales y ha demostrado ser efectiva para mejorar el rendimiento en conjuntos de datos limitados o desbalanceados [32].

La Figura 12 y la Figura 13 muestran, respectivamente, un agujero en su imagen original y el mismo agujero tras aplicar técnicas de aumento de datos, incluyendo rotación, ajuste de brillo y otras transformaciones.

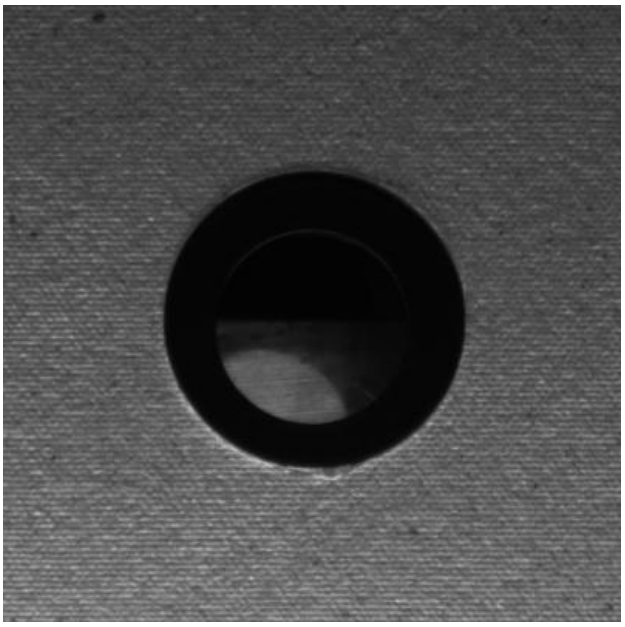

Figura 12: Agujero "nok" original

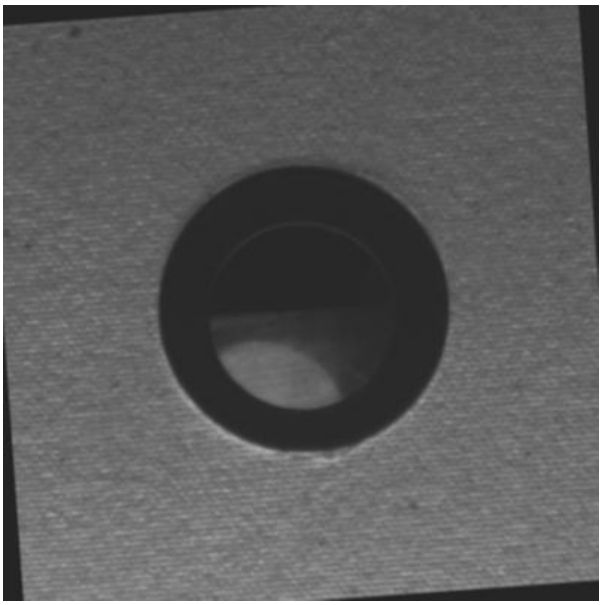

Figura 13: Agujero "nok" aumentado

4.5. Etiquetado y anotación de imágenes

Una vez segmentadas y aumentadas las imágenes individuales correspondientes a cada agujero, se procedió a la fase de etiquetado. Esta etapa fue esencial para supervisar el entrenamiento de los modelos de *deep learning*, ya que proporcionó la información de referencia que guía el aprendizaje del sistema.

En función del tipo de modelo utilizado, se diferenciaron dos enfoques:

4.5.1. Etiquetado para clasificación binaria

Para los modelos basados en clasificación de imágenes (como ResNet o EfficientNet), cada recorte fue asignado a una de dos clases:

- ok: agujeros que presentan una soldadura correcta y arandela en su lugar.
- nok: casos con defectos visibles, como arandela ausente.

El etiquetado se realizó de manera manual tras una inspección visual exhaustiva de las imágenes recortadas. Esta tarea fue llevada a cabo con especial cuidado, dado que la calidad de las etiquetas afecta directamente al rendimiento del sistema. Las etiquetas se mediante la estructura de carpetas, lo que facilitó su integración con los flujos de entrenamiento.

La Figura 14 y la Figura 15 muestran, respectivamente, ejemplos de un agujero correcto y de un agujero defectuoso.

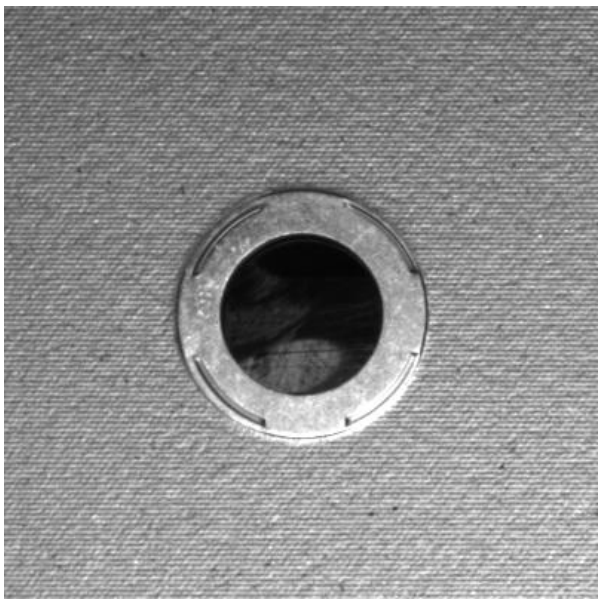

Figura 14: Ejemplo de agujero "ok"

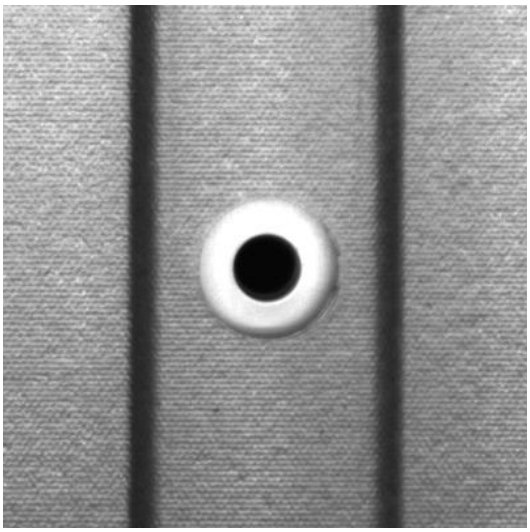

Figura 15: Ejemplo de agujero "nok"

4.5.2. Anotación para detección de objetos (YOLO)

En el caso del modelo YOLO, se requería no solo clasificar la presencia de defectos, sino también delimitar su localización dentro de la imagen mediante bounding boxes. Para ello, se empleó la herramienta *LabelImg* [33], ampliamente utilizada en la anotación de datos para tareas de visión artificial.

Con esta herramienta, se definieron manualmente los cuadros delimitadores de los defectos detectados en cada imagen, generando archivos de texto asociados en formato YOLO. Cada línea de estos archivos seguía la estructura:

$$< class\_id >< x\_center >< y\_center >< width >< height >$$

donde todas las coordenadas están normalizadas entre 0 y 1 respecto al tamaño de la imagen. La clase asignada (0 para "*ok*", 1 para "*nok*") dependía del contenido visual de la región anotada.

El uso de *LabelImg* facilitó el proceso gracias a su interfaz gráfica, su compatibilidad directa con YOLO y su capacidad de exportación en lote. Este etiquetado preciso era imprescindible para que el modelo pudiera aprender a localizar y clasificar defectos con precisión espacial, más allá de una simple decisión global por imagen.

4.6. División del conjunto de datos

Con el conjunto de imágenes preprocesado, aumentado y etiquetado, se realizó la división del conjunto de datos en tres subconjuntos con fines de entrenamiento, validación y prueba. Esta partición es una práctica estándar en el desarrollo de sistemas de aprendizaje automático supervisado, ya que permite evaluar de forma objetiva la capacidad del modelo para generalizar a datos no vistos durante el entrenamiento.

Distribución de clases

El conjunto total estuvo formado por 1832 imágenes individuales de agujeros, distribuidas entre las clases ok y nok, correspondientes a agujeros sin y con defectos, respectivamente. En la siguiente tabla se muestra la distribución de muestras en cada subconjunto:

Conjunto	*OK*	*NOK*	*Total*
Entrenamiento	759	524	1283
Validación	325	224	549
Total	1084	748	1832

Tabla 2: División del conjunto de datos

Se observa una proporción equilibrada entre ambas clases, con una ligera mayoría de imágenes ok, lo que fue compensado en parte mediante técnicas de *data augmentation (*Subsección *4.4)* aplicadas antes de esta división. Tanto el conjunto de entrenamiento como el de validación contienen una distribución representativa, lo cual fue crucial para entrenar y evaluar modelos robustos y generalizables.

El conjunto de prueba fue reservado y utilizado exclusivamente en la etapa de evaluación final (Subsección 5.4), por lo que sus cifras no se incluyen en la tabla anterior.

4.7. Entrenamiento de los modelos

Una vez preparado y dividido el conjunto de datos, se procedió a la fase de entrenamiento de los distintos modelos seleccionados: YOLOv11, ResNet18 y EfficientNet-B0. Esta etapa consistió en ajustar los pesos internos de cada red para que pudieran aprender a reconocer los patrones asociados a defectos en los agujeros de las tapas de batería.

Cada modelo fue entrenado por separado bajo las mismas condiciones experimentales en cuanto al conjunto de datos (división y etiquetado), permitiendo una comparación objetiva de su rendimiento.

4.7.1. YOLOv11

El modelo YOLOv11 (versión 11) es una de las arquitecturas más recientes de la familia YOLO. El entrenamiento del modelo YOLOv11 se llevó a cabo partiendo de la versión nano preentrenada (yolo11n.pt) sobre el conjunto COCO, lo que permitió aprovechar pesos previamente optimizados para tareas generales de detección y adaptarlos al

problema específico mediante *fine-tuning*. El tamaño de entrada de las imágenes se fijó en 640x640 píxeles, utilizando *padding* cuando fue necesario para preservar la relación de aspecto original. El proceso se configuró para ejecutarse durante 100 épocas, evaluando el rendimiento en el conjunto de validación al final de cada una de ellas.

Durante el entrenamiento se utilizaron las configuraciones por defecto de YOLOv11 para el optimizador *AdamW*, con una tasa de aprendizaje inicial de $1x10^{-3}$ y un programador (*scheduler*) de tipo *cosine*, que reduce progresivamente el learning rate a medida que avanza el entrenamiento. Para mejorar la capacidad de generalización del modelo y mitigar el sobreajuste, se aplicaron técnicas de data augmentation integradas en el propio marco de entrenamiento, incluyendo rotaciones leves, variaciones de brillo y contraste, flip horizontal y escalado aleatorio.

La ejecución se realizó en un entorno con NVIDIA GeForce RTX 3060 Laptop GPU, utilizando Python 3.12.7, PyTorch 2.5.1 y CUDA 11.8. El sistema generó automáticamente las métricas de rendimiento más relevantes para la detección, como la precisión, el *recall* y el mAP@0.5, almacenando como resultado final las ponderaciones correspondientes a la mejor época según el valor máximo de mAP obtenido en el conjunto de validación.

4.7.2. Clasificadores (ResNet 18 y EfficientNet-B0)

Para la etapa de clasificación binaria de cada agujero se emplearon las dos arquitecturas mencionadas: ResNet18 y EfficientNet-B0, ambas con pesos preentrenados en ImageNet y posteriormente ajustadas por transferencia de aprendizaje al dominio específico del proyecto. En ambos casos se sustituyó la capa final por un único *logit* (salida de dimensión 1) con el fin de optimizar directamente una pérdida binaria; concretamente se utilizó *BCEWithLogitsLoss*, que integra la activación sigmoide en el propio término de la pérdida y mejora la estabilidad numérica en comparación con aplicar primero una sigmoide y después una *Binary Cross Entropy* convencional.

El input de los modelos se normalizó siguiendo las estadísticas estándar de ImageNet y se redimensionó a 224x224 píxeles, garantizando así compatibilidad con las capas iniciales de ambas arquitecturas y manteniendo un flujo de datos homogéneo. La

organización del *dataset* se realizó mediante la estructura *ImageFolder* (carpetas train/ y val/ con subcarpetas por clase), lo que facilitó la carga con *DataLoaders* y el barajado por lotes durante el entrenamiento.

El procedimiento de optimización fue idéntico para ambas redes con el objetivo de obtener una comparación justa: optimización con Adam con tasa de aprendizaje inicial $1x10^{-3}$, *batch size* de 32 (ajustado según memoria disponible) y un número de épocas en el rango de 10–50 (según convergencia observada en validación). No se congelaron capas de forma explícita, por lo que el ajuste fue de extremo a extremo (fine-tuning completo) sobre pesos preentrenados, permitiendo que tanto los bloques iniciales como la nueva capa final se adaptaran a las características visuales de los agujeros y sus defectos. Durante la validación, las salidas se transformaron con sigmoide y se umbralizaron a 0.5 para obtener la etiqueta predicha; este umbral, si fuera necesario, puede ajustarse posteriormente en función del compromiso precision–recall requerido por la operación en planta.

En cada época se monitorizó la pérdida de entrenamiento y, en validación, se calcularon métricas de clasificación (precisión, recall, F1 y accuracy) a partir de las predicciones binarias. Finalizado el proceso, se guardó el mejor estado de cada modelo (resnet18_binary.pt, efficientnet_b0_binary.pt) para su posterior evaluación en el capítulo de resultados. Todo el entrenamiento se ejecutó sobre GPU, con PyTorch 2.5 y Python 3.12, manteniendo el mismo entorno general utilizado para el entrenamiento de YOLOv11 a fin de asegurar coherencia experimental entre detectores y clasificadores.

4.8. Métricas de evaluación y validación del modelo

Para garantizar una evaluación rigurosa del desempeño de los modelos empleados en este proyecto, se han utilizado métricas estándar derivadas de la matriz de confusión, así como medidas complementarias que permiten valorar la eficacia del sistema desde una perspectiva cuantitativa [34]. Estas métricas son fundamentales para comparar modelos, interpretar sus puntos fuertes y débiles, y seleccionar el más adecuado para su despliegue en un entorno industrial.

4.8.1. Matriz de confusión

La matriz de confusión es una herramienta clave para evaluar sistemas de clasificación binaria, como el caso de estudio presente, donde cada muestra (agujero de la tapa de batería) se clasifica como *ok* (sin defecto) o *nok* (con defecto). Esta matriz incluye cuatro valores:

- **TP (True Positives):** agujeros correctos correctamente identificados como correctos.
- **TN (True Negatives):** agujeros defectuosos correctamente identificados como defectuosos.
- **FP (False Positives):** agujeros defectuosos clasificados erróneamente como correctos.
- **FN (False Negatives):** agujeros correctos no detectados por el modelo (clasificados como defectuosos).

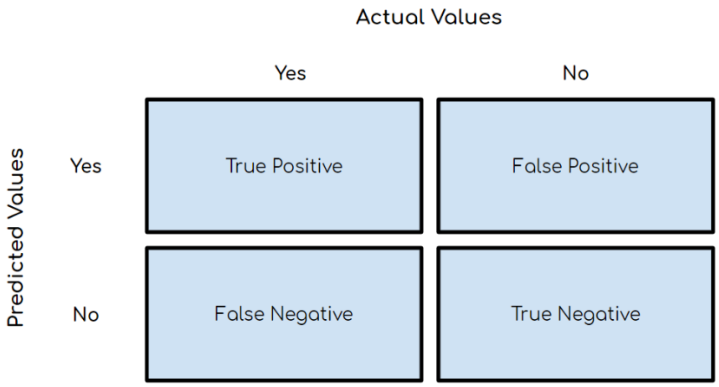

Figura 16: Ejemplo de una matriz de confusión [66]

A partir de estos valores se calculan métricas derivadas, descritas a continuación.

4.8.2. Precisión global *(Accuracy)*

Representa el porcentaje de predicciones correctas sobre el total. Aunque es una métrica ampliamente utilizada, puede resultar engañosa en contextos con clases desbalanceadas (por ejemplo, si los defectos son poco frecuentes), motivo por el cual se complementa con otras métricas.

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

4.8.3. Precisión positiva *(Precision)*

Indica el porcentaje de predicciones positivas (agujeros correctos) que realmente eran correctas. Es especialmente relevante en el contexto industrial, donde un exceso de falsos positivos podría derivar en el rechazo innecesario de piezas válidas.

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

4.8.4. Sensibilidad *(Recall)*

Mide la capacidad del modelo para detectar la totalidad de los casos positivos. Es crítica cuando el coste de no detectar un defecto (falso negativo) es elevado, como ocurre en entornos industriales con estándares de calidad estrictos.

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

4.8.5. F1-Score

Combina precisión y sensibilidad en una única métrica armónica, proporcionando una medida equilibrada del rendimiento del modelo.

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

4.8.6. mAP (*Mean Average Precision*)

El *Mean Average Precision* es una de las métricas más utilizadas para evaluar el rendimiento de modelos de detección de objetos. Se calcula promediando la precisión promedio (*Average Precision*, AP) obtenida para cada clase en el conjunto de validación o prueba. La AP de una clase se deriva de la curva precisión–*recall*, que se construye variando el umbral de confianza en las predicciones del modelo.

En problemas de detección, es habitual indicar el mAP junto a un umbral de IoU (*Intersection over Union*), como en mAP@0.5, que evalúa las predicciones correctas cuando el solapamiento entre el *bounding box* predicho y el de referencia es al menos del 50 %. Valores como mAP@0.5:0.95 indican el promedio sobre diferentes umbrales de

IoU (desde 0.5 hasta 0.95 con incrementos de 0.05), proporcionando una evaluación más estricta del rendimiento.

Esta métrica ofrece una visión equilibrada de la capacidad del modelo para localizar y clasificar correctamente los objetos, y resulta especialmente útil en contextos industriales donde la precisión en la localización es tan relevante como la correcta clasificación del defecto.

4.9. Limitaciones del sistema

A pesar del alcance técnico del sistema, existen una serie de limitaciones derivadas tanto del contexto industrial como de las restricciones tecnológicas y del propio marco del proyecto académico. Las principales limitaciones son las siguientes:

- **Modalidad de imagen:**

El sistema trabaja exclusivamente con imágenes RGB. No se emplean sensores térmicos, cámaras hiperespectrales ni sistemas de escaneo 3D, por lo que defectos internos o no visibles superficialmente no pueden ser detectados.

- **Naturaleza superficial del defecto:**

La detección está limitada a imperfecciones visibles en la soldadura (ausencia, desplazamiento, irregularidades superficiales), pero no incluye pruebas de resistencia, integridad estructural o calidad metalúrgica.

- **Alcance de la generalización:**

El sistema ha sido entrenado y validado exclusivamente con imágenes proporcionadas por Gestamp, bajo condiciones de iluminación y calidad propias de su entorno de producción. Su aplicabilidad a otros entornos requeriría un reentrenamiento y adaptación del pipeline.

- **Número de clases:**

La clasificación de los agujeros es binaria (ok/nok). No se contempla una clasificación multiclase según el tipo exacto de defecto, aunque esta posibilidad se deja abierta para desarrollos futuros.

- **Requisitos computacionales:**

Aunque se ha priorizado la eficiencia, algunos modelos empleados requieren GPU para entrenamiento y despliegue óptimo. La integración en entornos de producción con restricciones de hardware puede requerir técnicas de optimización o modelos más ligeros.

- **Evaluación en entorno real:**

Por tratarse de un proyecto académico, el sistema ha sido evaluado en entorno controlado y no ha sido desplegado ni validado de forma integrada en la línea de producción de Gestamp.

Capítulo 5. Análisis de resultados

En este capítulo se presentan y analizan los resultados obtenidos tras el entrenamiento y validación de los modelos de visión artificial desarrollados en el marco de este proyecto. El objetivo principal es evaluar su rendimiento en la detección y clasificación automática de defectos de soldadura en las arandelas de los agujeros de tapas de baterías, comparando las métricas de precisión, *recall*, F1 y mAP.

El análisis incluye tanto la evaluación individual de cada arquitectura (YOLOv11, ResNet18 y EfficientNet-B0) como una comparativa global, con el fin de identificar el modelo más adecuado para su implementación en un entorno industrial real. Asimismo, se examinan los casos de error más relevantes, las limitaciones detectadas y las implicaciones prácticas de los resultados.

Los resultados presentados en esta sección corresponden al conjunto de validación, utilizado durante la fase de entrenamiento para ajustar los hiperparámetros y evaluar el desempeño de los modelos en datos no vistos previamente. Por su parte, los resultados obtenidos sobre el conjunto de prueba, reservados para la evaluación final del sistema, se presentarán y discutirán en la Subsección 5.4, garantizando así una valoración imparcial y representativa del modelo en condiciones reales de despliegue.

5.1. Resultados de YOLOv11

El entrenamiento del modelo YOLOv11 se evaluó mediante métricas estándar en detección de objetos, incluyendo curvas de Precisión–Confianza, Precisión–Recall, Recall–Confianza y F1–Confianza, así como la evolución de las pérdidas durante las 100 épocas de entrenamiento.

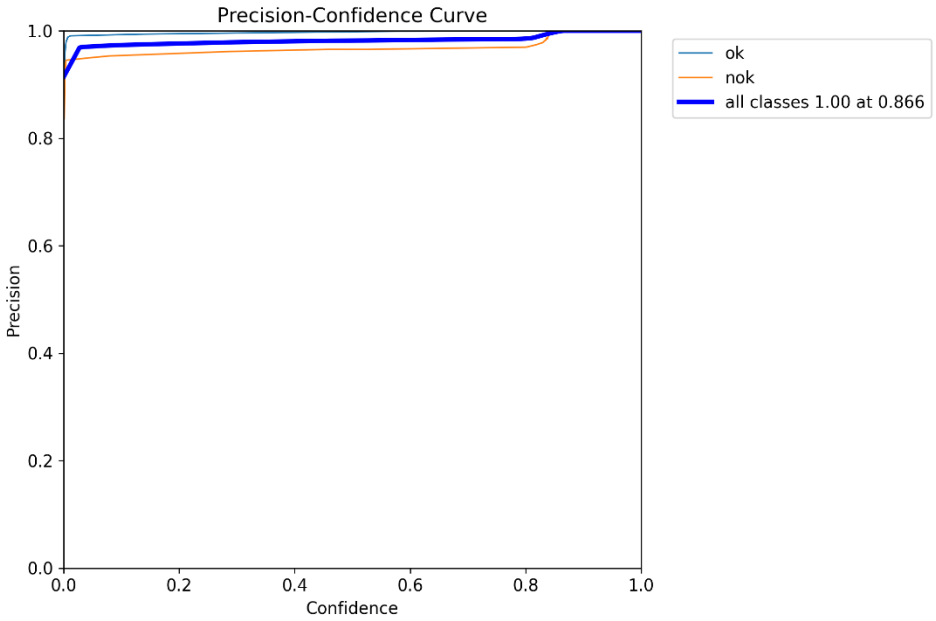

Figura 17: Curva Presicion-Confidence YOLOv11

En la curva Precisión–Confianza (Figura 17), se observa que el modelo alcanza una precisión cercana al 100 % para valores de confianza superiores a 0.85, con un comportamiento muy consistente entre las clases *ok* y *nok*. Esto sugiere que, al aumentar el umbral de confianza, las predicciones positivas tienden a ser correctas, reduciendo al mínimo los falsos positivos.

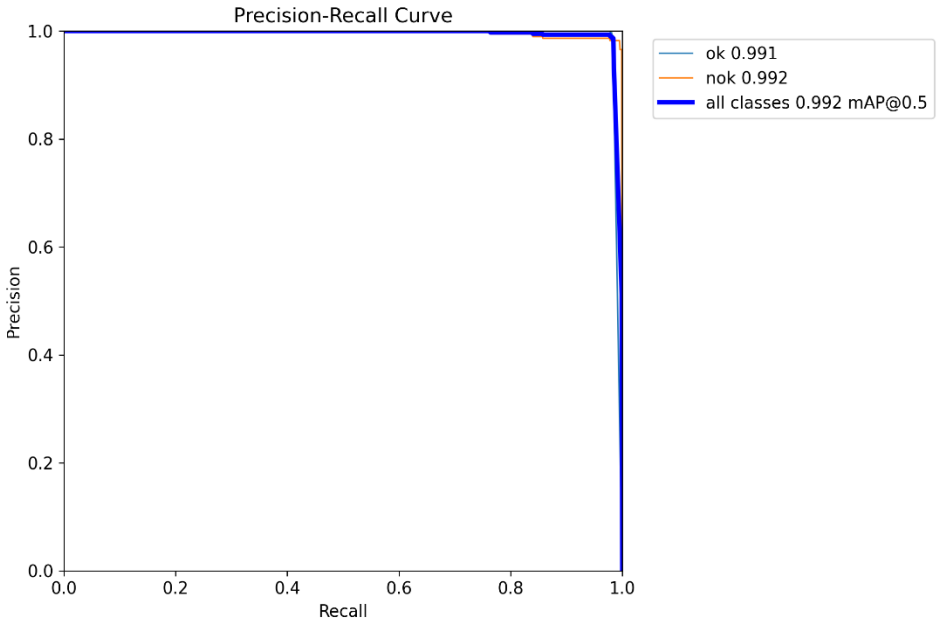

Figura 18: Curva Presicion-Recall YOLOv11

La curva Precisión–Recall (Figura 18) muestra un *mAP@0.5* promedio de 0.992, con valores muy próximos para ambas clases (0.991 para *ok* y 0.992 para *nok*). Este alto rendimiento en todas las clases indica que el modelo es capaz de detectar defectos con gran precisión y sin sacrificar cobertura.

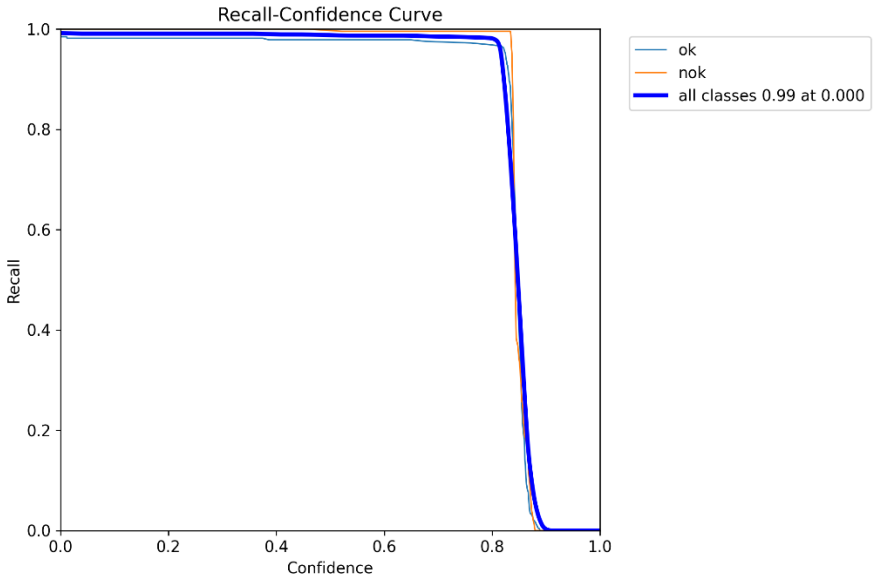

Figura 19: Curva Recall-Confidence YOLOv11

En la curva Recall–Confianza (Figura 19), se aprecia que el *recall* se mantiene cercano a 1.0 para niveles de confianza bajos y medios, disminuyendo únicamente cuando se aplican umbrales muy estrictos (>0.85). Este patrón es común en modelos bien calibrados, donde existe un equilibrio entre mantener alta la cobertura y evitar falsas detecciones.

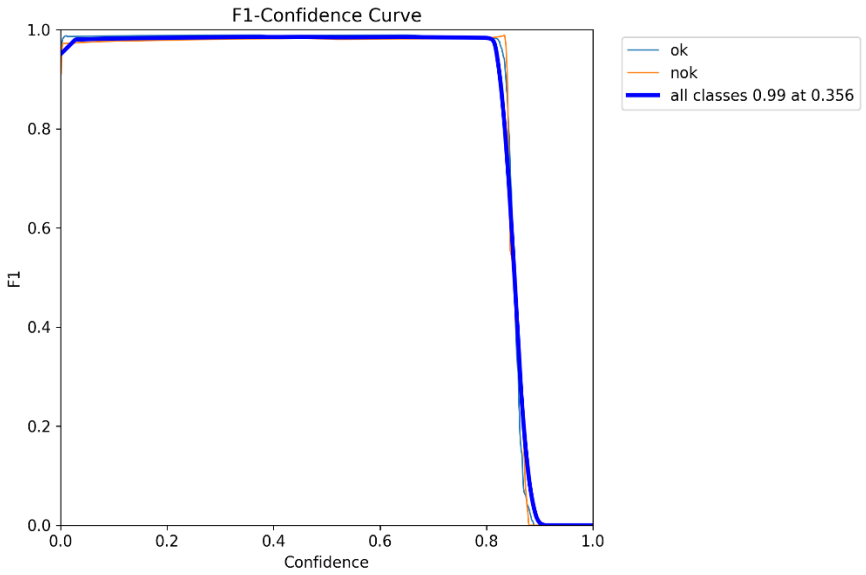

Figura 20: Curva F1-Confidence YOLOv11

La curva F1–Confianza (Figura 20) confirma un comportamiento equilibrado entre precisión y recall, alcanzando valores máximos cercanos a 0.99 para un umbral de confianza óptimo alrededor de 0.35.

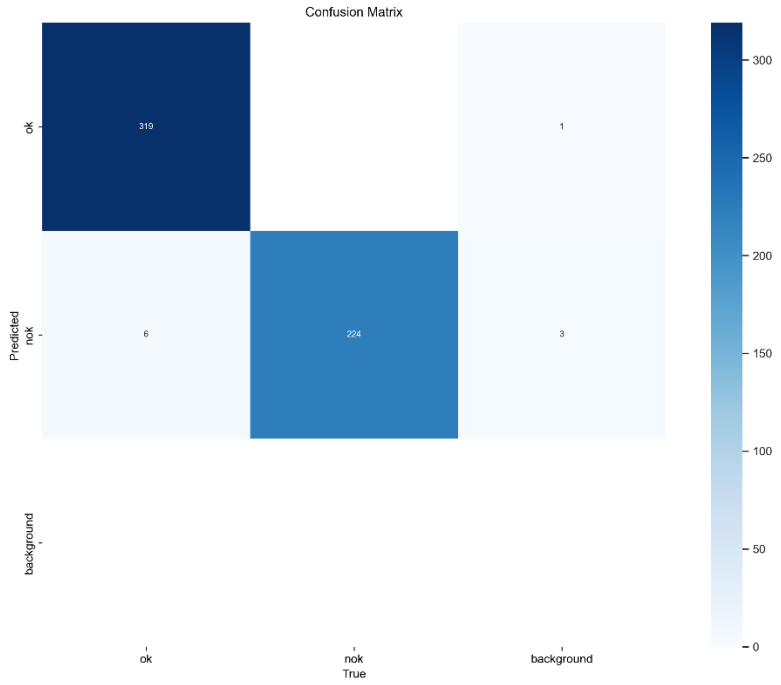

Figura 21: Matriz de confusión para el modelo YOLOv11

Finalmente, la matriz de confusión (Figura 21) evidencia la capacidad del modelo para clasificar correctamente los ejemplos de ambas clases. De un total de 549 instancias evaluadas, se obtuvieron únicamente 10 errores de clasificación, distribuidos en 6 falsos negativos y 4 agujeros que no fueron correctamente detectados, siendo confundidos por el fondo. Este tipo de fallo sugiere que, en algunos casos puntuales, el contraste entre el agujero y el entorno no fue suficiente para una correcta detección, representando un área de mejora para futuros desarrollos.

5.2. Resultados de ResNet18

A modo de resumen y recordatorio, el modelo ResNet18 fue entrenado utilizando el mismo conjunto de datos balanceado y aumentado descrito en el Capítulo 5. Las imágenes fueron redimensionadas a 224x224 píxeles y normalizadas con los valores estándar de *torchvision*. Se empleó un *batch size* de 32, optimizador Adam con una tasa de aprendizaje de 0,001 y función de pérdida *BCEWithLogitsLoss*. El entrenamiento se desarrolló durante 10 épocas, evaluándose posteriormente el modelo sobre el conjunto de validación para medir su capacidad de generalización.

5.2.1. Métricas de evaluación

En la Tabla 3 se recogen las métricas obtenidas sobre el conjunto de validación:

Métrica	*Valor*
Accuracy	0.9763
Precision	1.0000
Recall	0.9600
F1-Score	0.9796

Tabla 3: Métricas de evaluación para el modelo ResNet18

El modelo alcanzó una *accuracy* del 97,63 %, con una *precision* perfecta del 100 %, lo que significa que no se cometieron falsos positivos. El *recall* del 96,00 % indica que un pequeño porcentaje de piezas correctas fue clasificado como defectuoso, mientras que el *F1-Score* de 0,9796 confirma un equilibrio muy alto entre precisión y exhaustividad en la detección.

5.2.2. Matriz de confusión

La matriz de confusión obtenida se presenta en la Tabla 4:

	Real OK	*Real NOK*
Pred. OK	312	0
Pred. NOK	13	224

Tabla 4: Matriz de confusión para el modelo ResNet18

La clasificación fue correcta en 224 casos *nok* y 312 casos *ok*. No se produjeron falsos positivos y se registraron únicamente 13 falsos negativos, lo que representa un rendimiento muy consistente al igual que otros modelos evaluados.

5.3. Resultados de EfficientNet-B0

Del mismo modo, como introducción a los resultados, el modelo EfficientNet-B0 fue entrenado utilizando el conjunto de datos balanceado y aumentado descrito en el Capítulo

4. Sistema desarrollado, compuesto por imágenes recortadas de los agujeros obtenidos a partir de las fotografías originales proporcionadas por Gestamp. Las imágenes fueron redimensionadas a 224x224 píxeles y normalizadas con los valores estándar de la librería *torchvision*. El entrenamiento se realizó con un *batch size* de 32, optimizador Adam con una tasa de aprendizaje inicial de 0,001 y función de pérdida *BCEWithLogitsLoss*, durante 10 épocas. La evaluación se llevó a cabo sobre el conjunto de validación para medir el rendimiento del modelo en datos no vistos.

5.3.1. Métricas de evaluación

En la Tabla 5 se recogen las métricas obtenidas sobre el conjunto de validación:

Métrica	*Valor*
Accuracy	0.9727
Precision	1.0000
Recall	0.9538
F1-Score	0.9764

Tabla 5: Métricas de evaluación para el modelo EfficientNet-B0

El modelo alcanzó una *accuracy* del 97,27 %, lo que indica que clasifica correctamente la gran mayoría de las imágenes evaluadas. La *precision* del 100 % confirma que no se produjeron falsos positivos, es decir, no se etiquetó ninguna pieza defectuosa *nok* como correcta *ok*. El *recall* del 95,38 % evidencia la existencia de algunos falsos negativos, casos en los que piezas correctas fueron clasificadas como defectuosas. El *F1-Score*, que combina ambas métricas, alcanzó un valor de 0,9764, lo que respalda la alta fiabilidad del modelo.

5.3.2. Matriz de confusión

La matriz de confusión obtenida se presenta en la **Erro! Fonte de referência não encontrada.**:

	Real OK	*Real NOK*
Pred. OK	310	0
Pred. NOK	15	224

Tabla 6: Matriz de confusión para el modelo EfficientNet-B0

La matriz muestra que el modelo clasificó correctamente 224 casos *nok* y 310 casos *ok*. No se produjeron falsos positivos, mientras que se detectaron 15 falsos negativos, correspondientes a piezas correctas clasificadas como defectuosas. Este comportamiento

refleja una tendencia conservadora del modelo, priorizando la detección de defectos sobre la reducción de falsos rechazos.

5.4. Evaluación en prueba

Tras la fase de validación, en la que se analizaron las métricas de los modelos a partir de un conjunto de imágenes separadas del entrenamiento, resultaba fundamental comprobar la capacidad de generalización de las arquitecturas propuestas sobre un conjunto de prueba completamente independiente. Esta etapa es esencial para garantizar que los modelos no solamente se ajustan a los datos conocidos, sino que también son capaces de ofrecer un rendimiento consistente cuando se enfrentan a nuevas imágenes nunca vistas durante el proceso de entrenamiento ni en la validación.

El conjunto de prueba estuvo conformado por nueve piezas distintas, cada una de ellas con un número de imágenes comprendido entre 37 y 38 agujeros. Todas las imágenes fueron etiquetadas previamente de manera manual, generando así el *ground truth* (archivo con los agujeros correctamente etiquetados) necesario para la evaluación objetiva. Cada agujero se clasificó en una de las dos categorías mencionadas.

En esta fase se evaluaron los tres modelos objeto de estudio: YOLO, EfficientNet y ResNet. El proceso seguido para cada uno difiere ligeramente en función del formato de salida de las predicciones. En el caso de YOLO, las predicciones se almacenaron en ficheros de texto (.txt), uno por cada agujero, de manera que el primer valor de cada fichero correspondía a la clase predicha (0 para OK y 1 para NOK). No obstante, se detectaron situaciones en las que YOLO producía más de una detección por imagen. En estos casos, y con el fin de reflejar la inestabilidad del modelo, se decidió contabilizar la predicción como errónea. Esta decisión metodológica permite que las métricas calculadas representen de forma más fiel el rendimiento práctico del modelo, penalizando aquellos casos en los que el sistema ofrece múltiples resultados para una única muestra.

En el caso de EfficientNet y ResNet, el formato de salida fue un archivo denominado test_predictions.csv, en el que cada línea recogía el nombre de la imagen, la probabilidad estimada por el modelo y la clase asignada. Estas etiquetas textuales se transformaron en un formato binario (0 para OK y 1 para NOK), de modo que fuese posible unificar el

procedimiento de evaluación con el utilizado para YOLO. Además, en todos los casos se cuidó de mantener una nomenclatura homogénea para los nombres de archivo, lo que permitió cruzar de manera automática las predicciones con las etiquetas del *ground truth* correspondiente.

Junto a las métricas de clasificación, se decidió incorporar en la evaluación el tiempo de inferencia de cada modelo para la pieza entera, almacenado en un archivo independiente (inference_time.txt) dentro de la carpeta correspondiente a cada pieza. Este parámetro es especialmente relevante en aplicaciones industriales en tiempo real, donde no basta con alcanzar altos valores de precisión o recall, sino que también es necesario garantizar que las predicciones se obtienen en un tiempo reducido.

En el caso del conjunto de prueba, se ha decidido no reportar la métrica de *precisión*, dado que ninguno de los modelos evaluados clasificó como defectuosos (*NOK*) agujeros que realmente eran correctos (*OK*). En otras palabras, todos alcanzaron la precisión máxima (1.0). Este resultado, lejos de ser trivial, es especialmente positivo en un contexto industrial, ya que implica que el sistema no generaría falsos rechazos de tapas en buen estado, evitando así costos innecesarios asociados al descarte erróneo de piezas válidas.

De esta manera, las métricas consideradas en esta evaluación fueron:

- **Accuracy**
- **Recall**
- **F1-score**
- **Inference time:** tiempo medio de procesamiento por pieza, expresado en segundos, como indicador de eficiencia computacional.

Los resultados obtenidos se organizaron en dos niveles de análisis. En primer lugar, se elaboró una tabla detallada con las métricas de cada modelo para cada pieza (Tabla 7), lo que permite observar la variabilidad de desempeño entre piezas concretas y analizar casos en los que un modelo puede comportarse de forma más inestable. En segundo lugar, se calcularon los valores promedio de cada métrica por modelo (Tabla 8), proporcionando así una visión global y resumida del rendimiento de cada arquitectura en el conjunto de *test*. Esta doble aproximación ofrece una perspectiva más completa: por un lado, se

estudia la variabilidad pieza a pieza, y por otro se obtiene un resumen que facilita la comparación directa entre YOLO, EfficientNet y ResNet.

pieza	*modelo*	*accuracy*	*recall*	*f1*	*inference_time (s)*
Pieza 1	YOLO	0.921	0.889	0.941	3.011
Pieza 1	EfficientNet	0.921	0.889	0.941	1.574
Pieza 1	ResNet	0.921	0.889	0.941	**1.528**
Pieza 2	YOLO	1.000	1.000	**1.000**	3.285
Pieza 2	EfficientNet	0.973	0.941	0.970	1.431
Pieza 2	ResNet	0.973	0.941	0.970	**1.404**
Pieza 3	YOLO	0.973	0.944	**0.971**	3.416
Pieza 3	EfficientNet	0.842	0.684	0.813	1.753
Pieza 3	ResNet	0.842	0.684	0.813	**1.573**
Pieza 4	YOLO	1.000	1.000	**1.000**	3.595
Pieza 4	EfficientNet	0.921	0.900	0.947	1.610
Pieza 4	ResNet	0.921	0.900	0.947	**1.517**
Pieza 5	YOLO	1.000	1.000	**1.000**	3.417
Pieza 5	EfficientNet	0.921	0.917	0.957	1.722
Pieza 5	ResNet	0.921	0.917	0.957	**1.509**
Pieza 6	YOLO	1.000	0.000	0.000	3.302
Pieza 6	EfficientNet	0.947	0.000	0.000	1.476
Pieza 6	ResNet	0.947	0.000	0.000	**1.536**
Pieza 7	YOLO	1.000	1.000	**1.000**	3.298
Pieza 7	EfficientNet	0.974	0.973	0.986	1.725
Pieza 7	ResNet	0.974	0.973	0.986	**1.505**
Pieza 8	YOLO	1.000	1.000	**1.000**	3.437
Pieza 8	EfficientNet	0.921	0.919	0.958	1.642
Pieza 8	ResNet	0.921	0.919	0.958	**1.492**
Pieza 9	YOLO	1.000	1.000	**1.000**	3.387
Pieza 9	EfficientNet	0.895	0.852	0.920	**1.480**
Pieza 9	ResNet	0.895	0.852	0.920	1.501

Tabla 7: Métricas de cada modelo para cada pieza en la evaluación de prueba

modelo	*accuracy*	*precision*	*recall*	*f1*	*inference_time (s)*
EfficientNet	0.924	**0.889**	0.786	0.832	1.601
ResNet	0.924	**0.889**	0.786	0.832	**1.507**
YOLO	**0.988**	**0.889**	**0.870**	**0.879**	3.350

Tabla 8: Métricas promedio para cada modelo en la evaluación de prueba

5.5. Discusión de resultados

En una primera etapa, correspondiente a la evaluación en validación, se observaron valores de métricas globales altos y bastante similares entre los tres modelos analizados. Tanto YOLO, como EfficientNet y ResNet mostraron un desempeño notable, con tasas de acierto que evidencian una buena capacidad de aprendizaje a partir del conjunto de entrenamiento. En esta fase inicial, YOLO destacó ligeramente sobre las demás arquitecturas, alcanzando valores de exactitud y F1-score marginalmente superiores, aunque las diferencias no fueron lo suficientemente amplias como para establecer una conclusión clara sobre cuál de los modelos ofrecía un mejor rendimiento de manera consistente.

Este resultado es, en cierto sentido, esperado. El conjunto de validación procede de la misma distribución de datos que el de entrenamiento, lo que facilita que los modelos obtengan métricas altas y relativamente estables. La similitud entre los resultados de las tres arquitecturas en validación también indica que la tarea de clasificación entre agujeros OK y NOK puede resolverse de manera satisfactoria con diferentes aproximaciones de *deep learning*, y que la principal diferencia entre modelos podría no estar en su rendimiento sobre validación, sino en su comportamiento frente a datos nuevos, lo que se evalúa posteriormente en el conjunto de prueba.

De este modo, la fase de validación sirvió principalmente como una primera criba para descartar configuraciones poco prometedoras y confirmar que las tres arquitecturas seleccionadas eran viables para la tarea propuesta. Sin embargo, el análisis en validación no resultó concluyente respecto a qué modelo era más adecuado en términos de generalización, lo que motivó la necesidad de una evaluación adicional sobre un conjunto de *test* independiente.

Al pasar a la evaluación en test, los resultados permiten profundizar en el análisis y, sobre todo, identificar las razones detrás de los errores cometidos por cada modelo. Tal y como se refleja en la tabla de métricas por pieza (Tabla 7), la gran mayoría de errores provienen de agujeros previamente marcados por la empresa Gestamp con el objetivo de reforzar estas técnicas de inspección. Es decir, en general, los modelos aprendieron a reconocer adecuadamente la mayor parte de los defectos introducidos, pero se observaron ciertos casos particulares donde el rendimiento no fue el esperado.

Entre los errores más representativos se encuentran dos situaciones recurrentes. En primer lugar, algunos agujeros presentaban una pegatina colocada sobre la superficie, lo que generó detecciones dobles por parte de YOLO y, en ocasiones, una clasificación errónea. En segundo lugar, se identificaron agujeros OK recubiertos con pintura negra, lo que enmascaraba su apariencia y llevó a que varios modelos los predijeran incorrectamente como NOK. Estos casos son especialmente interesantes porque, aunque las imágenes con pegatinas y pintura también estaban presentes en el conjunto de entrenamiento, los modelos no siempre lograron generalizar correctamente en el conjunto de prueba, lo que pone de relieve las limitaciones que aún presentan estas arquitecturas frente a variaciones visuales específicas.

Otro aspecto relevante que merece discusión es el tiempo de inferencia. A diferencia de lo esperado, los modelos basados en clasificación, EfficientNet y ResNet, presentan tiempos muy similares y reducidos, en torno a 1,5–1,6 segundos por pieza. En cambio, YOLO, pese a ser una arquitectura concebida para la detección en tiempo real, en este caso mostró un tiempo de ejecución bastante mayor, alcanzando 3,35 segundos, es decir, aproximadamente el doble que las otras dos arquitecturas. Esta diferencia, aunque pueda parecer pequeña en términos absolutos, cobra especial importancia en un escenario de producción industrial, donde se deben inspeccionar miles de piezas al día y la latencia acumulada impacta directamente en la eficiencia del proceso.

En conjunto, la discusión de los resultados de prueba permite extraer conclusiones más claras que las obtenidas en validación. Si observamos la Tabla 8, si bien EfficientNet y ResNet ofrecen métricas prácticamente idénticas, con un *accuracy* del 92 % y un F1-score de 0,83, es YOLO quien logra un rendimiento superior, alcanzando un *accuracy* cercano al 99 % y un F1-score de 0,88. Esto confirma que YOLO es el modelo con mayor

capacidad de generalización frente a las condiciones adversas observadas en el conjunto de prueba. No obstante, su mayor calidad predictiva conlleva la desventaja de un tiempo de inferencia aproximadamente el doble de alto que el de EfficientNet y ResNet. Por lo tanto, la elección final dependerá del compromiso entre precisión y velocidad que se considere más adecuado para el entorno industrial en el que se vaya a implementar la solución.

Capítulo 6. Conclusiones y trabajos futuros

6.1. Conclusiones

El presente trabajo ha tenido como objetivo principal el desarrollo de un sistema de visión artificial para la detección automática de fallos de soldadura en las arandelas de los agujeros de tapas de baterías para automóviles, con el fin de descartar piezas defectuosas y contribuir a mejorar la eficiencia del control de calidad en un entorno industrial real.

En relación con los objetivos específicos, se pueden extraer las siguientes conclusiones:

1. **Análisis del contexto industrial y del problema planteado por Gestamp**

Se ha llevado a cabo un estudio detallado del proceso de fabricación y del sistema de inspección actual. Este análisis permitió comprender el flujo de producción, las características de las imágenes obtenidas y los tipos de defectos más relevantes, así como los criterios de rechazo que se aplican en la planta. Este paso resultó fundamental para contextualizar el proyecto y asegurar que el sistema propuesto se ajusta a las necesidades reales de la empresa.

2. **Construcción y validación de modelos de visión computacional**

Se entrenaron y evaluaron tres arquitecturas de *deep learning*: YOLO, EfficientNet y ResNet. Los modelos se validaron en un primer conjunto de datos, obteniendo resultados elevados y muy similares en términos de precisión. Posteriormente, se realizó una evaluación sobre un conjunto de prueba independiente que permitió confirmar la capacidad de generalización. De acuerdo con las métricas obtenidas, YOLO se mostró como el modelo más robusto y preciso, aunque con la desventaja de un tiempo de inferencia superior al de EfficientNet y ResNet.

3. **Clasificación automática de agujeros en clases ‘OK’ y ‘NOK’**

Se logró implementar un sistema binario de clasificación capaz de identificar con alta fiabilidad si cada agujero estaba correctamente soldado (*OK*) o presentaba defectos (*NOK*). Esta metodología, alineada con prácticas habituales en control de calidad, garantiza una decisión clara y objetiva para cada muestra analizada.

4. **Descartar automáticamente las piezas defectuosas**

El sistema propuesto fue diseñado para operar bajo la lógica de rechazo total: si un único agujero de una tapa es clasificado como *NOK*, la pieza completa se considera defectuosa. De este modo, se cumple con los criterios de inspección industrial y se minimiza el riesgo de que una batería defectuosa llegue a etapas posteriores de ensamblaje.

5. **Registro de localización y patrón de defectos**

Además de la clasificación binaria, el sistema permite registrar qué agujeros han sido identificados como defectuosos. Este mecanismo posibilita la trazabilidad de los errores y abre la puerta a análisis más profundos de causa raíz, con el objetivo de identificar patrones recurrentes de fallo y plantear acciones correctivas en la línea de producción.

En conjunto, el proyecto ha demostrado que es posible desarrollar un sistema de visión artificial eficaz y adaptado al contexto industrial real, capaz de complementar o incluso sustituir procesos de inspección manual menos eficientes. Se han cumplido los objetivos planteados y se ha generado una solución funcional que constituye una aportación práctica para el sector automotriz.

6.2. Trabajos futuros

Aunque el sistema desarrollado ha demostrado ser eficaz y cumplir con los objetivos planteados, aún quedan diversas líneas de mejora que podrían reforzar su aplicabilidad y robustez en un entorno industrial real. En primer lugar, sería interesante abordar la optimización de los tiempos de inferencia, ya que el modelo YOLO, si bien ha mostrado el mejor rendimiento en términos de exactitud y capacidad de detección, presenta una latencia aproximadamente el doble que las arquitecturas de clasificación. En este sentido, podrían explorarse variantes más ligeras de la misma familia, como versiones reducidas de YOLO.

Otro aspecto clave para el futuro desarrollo del proyecto es la ampliación y diversificación del *dataset*. Aunque las imágenes utilizadas han permitido entrenar y validar correctamente los modelos, se ha observado que ciertas condiciones específicas, como la presencia de pegatinas o la aplicación de pintura negra, dificultan la clasificación y

provocan errores puntuales. La incorporación de más ejemplos de estas situaciones, así como de nuevas condiciones de iluminación o variaciones en el proceso de producción, permitiría entrenar modelos más robustos y preparados para enfrentarse a la variabilidad inherente a un entorno industrial.

En una fase posterior, resultará fundamental integrar y validar el sistema en la propia línea de producción, evaluando no solo su rendimiento técnico, sino también su impacto en la eficiencia global del proceso y en la reducción de errores de inspección. Este despliegue permitiría analizar aspectos adicionales como la interacción con el resto de los sistemas de control de la planta, el tiempo de respuesta en condiciones reales y la aceptación por parte de los operarios.

Finalmente, el registro detallado de defectos generado por el sistema abre la puerta a desarrollar un módulo de retroalimentación y análisis en tiempo real. Este podría convertirse en una herramienta de gran valor para la empresa, ya que permitiría identificar patrones recurrentes de fallo, realizar un seguimiento de las causas raíz y proponer acciones correctivas de manera temprana. Asimismo, no se descarta la posibilidad de explorar enfoques híbridos, combinando modelos de detección y clasificación para aprovechar las ventajas de ambos y lograr un equilibrio aún mayor entre velocidad y precisión.

En conjunto, estas líneas de trabajo futuro no solo refuerzan la viabilidad del sistema desarrollado, sino que también abren nuevas oportunidades para consolidar la visión artificial como una herramienta clave en la mejora de los procesos de control de calidad en la industria automotriz.

Capítulo 7. Bibliografía

[1]. *La revolución de la visión artificial en la industria. (2024, marzo 8). Mottus.* https://www.mottus.es/la-revolucion-de-la-vision-artificial-en-la-industria/

[2]. *Saberironaghi, A., Ren, J., & El-Gindy, M. (2023). Defect Detection Methods for Industrial Products Using Deep Learning Techniques: A Review. Algorithms, 16(2), Article 2.* https://doi.org/10.3390/a16020095

[3]. *(PDF) Human Factors in Visual Quality Control. (s. f.). ResearchGate.* https://doi.org/10.1515/mper-2015-0013

[4]. *Muralidhar, S. (2025). Recent Advancements in Machine Vision Systems for Industrial Defect Detection: A Review. Journal of Engineering Research and Reports, 27(3), 385-392.* https://doi.org/10.9734/jerr/2025/v27i31441

[5]. *Reyes Domínguez, D., Infante Abreu, M. B., & Parv, A. L. (2024). Main Trend Topics on Industry 4.0 in the Manufacturing Sector: A Bibliometric Review. Applied Sciences, 14(15), Article 15.* https://doi.org/10.3390/app14156450

[6]. *Gestamp—Home. (s. f.). Recuperado 19 de julio de 2025, de* https://www.gestamp.com/en/home

[7]. *How to Deploy Deep Learning Neural Networks in Machine Vision | Vision Systems Design. (s. f.). Recuperado 4 de junio de 2025, de* https://www.vision-systems.com/boards-software/article/55286139/how-to-deploy-deep-learning-neural-networks-in-machine-vision

[8]. *The History of Artificial Intelligence: Complete AI Timeline. (s. f.). Search Enterprise AI. Recuperado 22 de junio de 2025, de* https://www.techtarget.com/searchenterpriseai/tip/The-history-of-artificial-intelligence-Complete-AI-timeline

[9]. *This week in The History of AI at AIWS.net – Arthur Samuel popularizes the term "machine learning" | aiws.net. (s. f.). Recuperado 22 de junio de 2025, de* https://aiws.net/the-history-of-ai/this-week-in-the-history-of-ai-at-aiws-net-arthur-samuel-popularizes-the-term-machine-learning/?utm_source=chatgpt.com

[10]. *Varela-Arregoces, E., & Campbells-S, E. (2011). Redes Neuronales Artificiales: Una revisión del estado del arte, aplicaciones y tendencias futuras. Investigación y desarrollo en TIC, 2(1), Article 1.*

[11]. *Propagación hacia delante en redes neuronales: Guía completa. (s. f.). Recuperado 4 de mayo de 2025, de* https://www.datacamp.com/tutorial/forward-propagation-neural-networks

[12]. *¿Qué es la retropropagación? | IBM. (2024, noviembre 25).* https://www.ibm.com/es-es/think/topics/backpropagation

[13]. *¿Qué es el descenso de gradiente? | IBM. (2021, octubre 7).* https://www.ibm.com/es-es/think/topics/gradient-descent

[14]. *CertiDevs. (2024, diciembre 13). Optimización en TensorFlow: Adam, SGD, RMSProp y más. CertiDevs.* https://certidevs.com/tutorial-tensorflow-optimizadores-adam-sgd-rmsprop

[15]. *¿Qué es el aprendizaje profundo? - Explicación de la IA de aprendizaje profundo - AWS. (s. f.). Amazon Web Services, Inc. Recuperado 4 de mayo de 2025, de* https://aws.amazon.com/es/what-is/deep-learning/

[16]. *Historical context and evolution of deep learning | Deep Learning Systems Class Notes. (s. f.). Fiveable. Recuperado 4 de mayo de 2025, de* https://library.fiveable.me/deep-learning-systems/unit-1/historical-context-evolution-deep-learning/study-guide/ALVCX29Pf2dG574E

[17]. *AlexNet and ImageNet: The Birth of Deep Learning | Pinecone. (s. f.). Recuperado 22 de mayo de 2025, de* https://www.pinecone.io/learn/series/image-search/imagenet/

[18]. *¿Qué es una red neuronal profunda? (s. f.). Recuperado 4 de abril de 2025, de* https://botpress.com/es/blog/deep-neural-network

[19]. *Tung, N. X. (2023, abril 17). ¿Cuáles son las ventajas y desventajas de agregar más nodos a DNN? - Academia EITCA. EITCA Academy.* https://es.eitca.org/inteligencia-artificial/eitc-ai-gcml-google-nube-aprendizaje-autom%C3%A1tico/primeros-pasos-en-el-aprendizaje-autom%C3%A1tico/estimadores-y-redes-neuronales-profundas/%C2%BFCu%C3%A1les-son-las-ventajas-y-desventajas-de-agregar-m%C3%A1s-nodos-a-dnn%3F/

[20]. *Diaz, L. (2023, diciembre 21). Técnicas de regularización ⟫ Optimiza tus redes neuronales. Escribe contenidos con IA.* https://blog.dinobrain.ai/tecnicas-de-regularizacion-para-redes-neuronales/

[21]. *¿Qué es la visión artificial? - Explicación de la IA y el aprendizaje automático de imágenes - AWS. (s. f.). Amazon Web Services, Inc. Recuperado 5 de agosto de 2025, de* https://aws.amazon.com/es/what-is/computer-vision/

[22]. *He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition (arXiv:1512.03385). arXiv.* https://doi.org/10.48550/arXiv.1512.03385

[23]. *Tan, M., & Le, Q. V. (2020). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks (arXiv:1905.11946). arXiv.* https://doi.org/10.48550/arXiv.1905.11946

[24]. *MobileNetV2: Inverted Residuals and Linear Bottlenecks | by Paul-Louis Pröve | TDS Archive | Medium. (s. f.). Recuperado 28 de abril de 2025, de* https://medium.com/data-science/mobilenetv2-inverted-residuals-and-linear-bottlenecks-8a4362f4ffd5

[25]. *[1506.02640] You Only Look Once: Unified, Real-Time Object Detection. (s. f.). Recuperado 28 de abril de 2025, de* https://arxiv.org/abs/1506.02640

[26]. *Automotive adhesive defect detection based on improved YOLOv8 | Signal, Image and Video Processing. (s. f.). Recuperado 16 de agosto de 2025, de* https://link.springer.com/article/10.1007/s11760-023-02932-1

[27]. *Dai, W., Li, D., Tang, D., Jiang, Q., Wang, D., Wang, H., & Peng, Y. (2021). Deep learning assisted vision inspection of resistance spot welds. Journal of Manufacturing Processes, 62, 262-274. https://doi.org/10.1016/j.jmapro.2020.12.015*

[28]. *Hachem, C. E., Perrot, G., Painvin, L., & Couturier, R. (2021). Automation of Quality Control in the Automotive Industry Using Deep Learning Algorithms. 2021 International Conference on Computer, Control and Robotics (ICCCR), 123-127. https://doi.org/10.1109/ICCCR49711.2021.9349273*

[29]. *Muresan, M. P., Cireap, D. G., & Giosan, I. (2020). Automatic Vision Inspection Solution for the Manufacturing Process of Automotive Components Through Plastic Injection Molding. 2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP), 423-430. https://doi.org/10.1109/ICCP51029.2020.9266249*

[30]. *Shimkus, B. (s. f.). AI cameras help Ford factory workers spot assembly errors and prevent costly recalls. Business Insider. Recuperado 16 de agosto de 2025, de https://www.businessinsider.com/ford-uses-ai-cameras-in-factories-prevent-recalls-costly-rework-2025-8*

[31]. *VisionPro Software | Cognex. (s. f.). Recuperado 16 de agosto de 2025, de https://www.cognex.com/products/machine-vision/vision-software/visionpro-software?utm_source=chatgpt.com*

[32]. *Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. Journal of Big Data, 6(1), 60. https://doi.org/10.1186/s40537-019-0197-0*

[33]. *labelImg: LabelImg is a graphical image annotation tool and label object bounding boxes in images (Versión 1.8.6). (s. f.). [Python]. Recuperado 4 de mayo de 2025, de https://github.com/tzutalin/labelImg*

[34]. *Métricas De Evaluación De Modelos En El Aprendizaje Automático. (2023, septiembre 25). DataSource.ai. https://www.datasource.ai/es/data-science-articles/view-source:https://www.datasource.ai/es/data-science-articles/metricas-de-evaluacion-de-modelos-en-el-aprendizaje-automatico*

[35]. *Chai, J., Zeng, H., Li, A., & Ngai, E. W. T. (2021). Deep learning in computer vision: A critical review of emerging techniques and application scenarios. Machine Learning with Applications, 6, 100134. https://doi.org/10.1016/j.mlwa.2021.100134*

[36]. *5.3. Forward Propagation, Backward Propagation, and Computational Graphs—Dive into Deep Learning 1.0.3 documentation. (s. f.). Recuperado 22 de mayo de 2025, de https://d2l.ai/chapter_multilayer-perceptrons/backprop.html*

[37]. *8.6. Residual Networks (ResNet) and ResNeXt—Dive into Deep Learning 1.0.3 documentation. (s. f.). Recuperado 28 de mayo de 2025, de https://d2l.ai/chapter_convolutional-modern/resnet.html*

[38]. *Ahmad, H. M., & Rahimi, A. (2022). Deep learning methods for object detection in smart manufacturing: A survey. Journal of Manufacturing Systems, 64, 181-196. https://doi.org/10.1016/j.jmsy.2022.06.011*

[39]. *Alif, M. A. R. (2024). YOLOv11 for Vehicle Detection: Advancements, Performance, and Applications in Intelligent Transportation Systems (arXiv:2410.22898). arXiv. https://doi.org/10.48550/arXiv.2410.22898*

[40]. *Applications of Machine Learning and Computer Vision in Industry 4.0. (s. f.). Recuperado 24 de julio de 2025, de https://www.mdpi.com/2076-3417/14/6/2431?utm_source=chatgpt.com*

[41]. *Artificial Intelligence (AI) vs. Machine Learning. (s. f.). CU-CAI. Recuperado 22 de julio de 2025, de https://ai.engineering.columbia.edu/ai-vs-machine-learning/*

[42]. *Bcnvision. (2024, mayo 30). Visión Artificial e IA en la Industria Automotriz. https://bcnvision.es/blog-vision-artificial/las-soluciones-de-vision-artificial-para-el-sector-de-la-automocion/*

[43]. *Bhatia, M. S., & Kumar, S. (2022). Critical Success Factors of Industry 4.0 in Automotive Manufacturing Industry. IEEE Transactions on Engineering Management, 69(5), 2439-2453. IEEE Transactions on Engineering Management. https://doi.org/10.1109/TEM.2020.3017004*

[44]. *Buettgenbach, M. H. (2021, noviembre 8). Explain like I'm five: Artificial neurons. Towards Data Science. https://towardsdatascience.com/explain-like-im-five-artificial-neurons-b7c475b56189/*

[45]. *Dai, W., Li, D., Tang, D., Jiang, Q., Wang, D., Wang, H., & Peng, Y. (2021). Deep learning assisted vision inspection of resistance spot welds. Journal of Manufacturing Processes, 62, 262-274. https://doi.org/10.1016/j.jmapro.2020.12.015*

[46]. *Diwan, T., Anirudh, G., & Tembhurne, J. V. (2023). Object detection using YOLO: Challenges, architectural successors, datasets and applications. Multimedia Tools and Applications, 82(6), 9243-9275. https://doi.org/10.1007/s11042-022-13644-y*

[47]. *Figure 2.4: Convolution operation by a (3 × 3) filter with stride=1,... (s. f.). ResearchGate. Recuperado 28 de julio de 2025, de https://www.researchgate.net/figure/Convolution-operation-by-a-3-3-filter-with-stride1-padding1-results-same-size_fig1_344197871*

[48]. *Gamez, M. J. (s. f.). Objetivos y metas de desarrollo sostenible. Desarrollo Sostenible. Recuperado 19 de julio de 2025, de https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/*

[49]. *Gill, R., Srivastava, D., Hooda, S., Singla, C., & Chaudhary, R. (2024). Unleashing Sustainable Efficiency: The Integration of Computer Vision into Industry 4.0. Engineering Management Journal. https://www.tandfonline.com/doi/abs/10.1080/10429247.2024.2383518*

[50]. He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition (arXiv:1512.03385). arXiv. https://doi.org/10.48550/arXiv.1512.03385

[51]. Introducción a las redes neuronales profundas. (s. f.). Recuperado 22 de julio de 2025, de https://www.datacamp.com/tutorial/introduction-to-deep-neural-networks

[52]. Leyva, E. R., & Baltazar, V. H. B. (s. f.). Desarrollo de un sistema de visión artificial para la evaluación de calidad de conectores en bolsas de aire automotrices.

[53]. Mazzetto, M., Teixeira, M., Rodrigues, É. O., & Casanova, D. (2020, julio 2). Deep Learning Models for Visual Inspection on Automotive Assembling Line. arXiv.Org. https://doi.org/10.22161/ijaers.74.56

[54]. Qais, A. (2021, junio 11). Introduction to Machine Learning. Analytics Vidhya. https://medium.com/analytics-vidhya/introduction-to-machine-learning-e1b9c055039c

[55]. ¿Qué es el deep learning? | IBM. (2024, junio 17). https://www.ibm.com/es-es/topics/deep-learning

[56]. ¿Qué es una red neuronal profunda? (s. f.). Recuperado 22 de julio de 2025, de https://botpress.com/es/blog/deep-neural-network

[57]. Revolucionando la industria automotriz con la visión artificial. (s. f.). Interempresas. Recuperado 22 de noviembre de 2024, de https://www.interempresas.net/Sector-Automocion/Articulos/568835-Revolucionando-la-industria-automotriz-con-la-vision-artificial.html

[58]. S, I. G., & S, V. C. (2015). La visión artificial y los campos de aplicación. Tierra Infinita, 1(1), Article 1. https://doi.org/10.32645/26028131.76

[59]. Two-Stream Network One-Class Classification Model for Defect Inspections. (s. f.). Recuperado 1 de abril de 2025, de https://www.mdpi.com/1424-8220/23/12/5768

[60]. Ultralytics. (s. f.). YOLO11 NUEVO. Recuperado 22 de noviembre de 2024, de https://docs.ultralytics.com/es/models/yolo11

[61]. Visión Artificial en Automoción—ATRIA Innovation. (2022, junio 7). https://atriainnovation.com/blog/vision-artificial-en-automocion/

[62]. Weld Seam Inspection in Battery Production |VITRONIC | Overview. (s. f.). Recuperado 19 de junio de 2025, de https://www.vitronic.com/en-us/automotive/weld-seam-inspection-in-battery-production

[63]. YOLO Algorithm for Object Detection Explained [+Examples]. (s. f.). Recuperado 1 de abril de 2025, de https://www.v7labs.com/blog/yolo-object-detection

[64]. What is EfficientNet? The Ultimate Guide. (s. f.). Recuperado 28 de junio de 2025, de https://blog.roboflow.com/what-is-efficientnet/

[65]. Architecture of EfficientNet-B0 with MBConv as Basic building blocks. | Download Scientific Diagram. (s. f.). Recuperado 28 de abril de 2025, de https://www.researchgate.net/figure/Architecture-of-EfficientNet-B0-with-MBConv-as-Basic-building-blocks_fig4_344410350?ref=blog.roboflow.com

[66]. Comprensión de la Matriz de Confusión y Cómo Implementarla en Python. (2020, mayo 20). DataSource.ai. https://www.datasource.ai/es/data-science-articles/view-source:https://www.datasource.ai/es/data-science-articles/comprension-de-la-matriz-de-confusion-y-como-implementarla-en-python

[67]. Conjuntos de datos: División del conjunto de datos original | Machine Learning. (s. f.). Google for Developers. Recuperado 5 de junio de 2025, de https://developers.google.com/machine-learning/crash-course/overfitting/dividing-datasets?hl=es-419