**Eje temático:** Análisis de señales, modelados y simulación

**Modalidad:** Trabajo en desarrollo – estudiante de grado

# Identificación de objetos tipo industrial por medio de análisis de imágenes con técnicas de aprendizaje profundo

## Identification of industrial objects through image analysis using deep learning techniques

Presentación 30/10/2025

Melani Fátima Faulkner

Universidad Tecnológica Nacional, Facultad Regional Reconquista, Grupo de Investigación en Programación, Electrónica y Control, Reconquista, Santa Fe, Argentina.

E-mail de contacto: mfaulkner2699@comunidad.frrq.utn.edu.ar

**Resumen**

La identificación de imágenes en el ámbito industrial se ha vuelto una herramienta esencial para realizar inspecciones visuales rápidas y precisas en la línea de producción. Permite reemplazar el trabajo de las personas en tareas repetitivas, mejorando la calidad del producto y reduciendo costos a largo plazo. El objetivo de este trabajo es proponer un sistema embebido de clasificación de imágenes de objetos tipo industrial, a través de plataformas de hardware sencillo como Raspberry Pi. Estos sistemas son económicos y versátiles, lo que facilita implementar una red neuronal convolucional (CNN) con alta exactitud y eficiencia. Además, se explican las técnicas de aumento de datos aplicadas para incrementar la robustez y la precisión del modelo. El desarrollo se realizó en Python y generó como resultado una gráfica que registra la evolución de la exactitud durante el proceso de entrenamiento y un programa de prueba para comprobar los resultados de la predicción.

**Palabras clave:** sistema embebido, red neuronal convolucional, EfficientNet, Raspberry Pi

**Abstract**

Image identification in the industrial field has become an essential tool for performing quick and accurate visual inspections on the production line. It allows the replacement of human labor in repetitive tasks, improving product quality and reducing long-term costs. The objective of this work is to propose an embedded system for classifying images of industrial objects, using simple hardware platforms such as Raspberry Pi. These systems are economical and versatile, making it easy to implement a convolutional neural network (CNN) with high accuracy and efficiency. In addition, the data augmentation techniques applied to increase the robustness and accuracy of the model are explained. The development was carried out in Python and resulted in a graph that records the evolution of accuracy during the training process and a test program to verify the prediction results.

**Keywords:** embedded system, convolutional neural network, EfficientNet, Raspberry Pi

## 1. Introducción

Los sistemas embebidos son dispositivos electrónicos diseñados para cumplir funciones específicas con recursos de procesamiento limitados. Estas restricciones de hardware, en términos de cómputo y memoria, exigen aplicaciones de clasificación de objetos eficientes. Sin embargo, pese a estas limitaciones, constituyen una solución económica y versátil para diversas aplicaciones industriales. En particular, mediante la implementación de plataformas como la Raspberry Pi (Halfacree, 2020) con una cámara integrada, es posible desarrollar aplicaciones de visión artificial con alta exactitud y bajo costo.

La propuesta de este trabajo consiste en el desarrollo de una aplicación académica basada en un sistema embebido que clasifique imágenes de objetos tipo industrial transportados en una cinta modelo ubicada en el laboratorio de automatización de la universidad. Este proyecto busca introducir a los participantes en el desarrollo de pipelines de AIoT “on the edge” para sistemas industriales.

Los pipelines de AIoT “on the edge” son flujos de procesamiento que integran dispositivos de Internet de las Cosas con algoritmos de inteligencia artificial. Estos ejecutan la captura de datos, su análisis y la generación de decisiones directamente sin depender de servidores externos.

El contenido del artículo se organiza de la siguiente manera: en la Sección 2 se describe la toma de datos, las técnicas empleadas para el procesamiento de imágenes y el entrenamiento del modelo; en la Sección 3 se presentan los resultados del entrenamiento mediante una gráfica y un programa de prueba. Finalmente, en la Sección 4 se exponen las conclusiones y se plantean futuras líneas de investigación.

## 2. Metodología

La Figura 1 muestra un resumen esquemático de las principales etapas del desarrollo. En primer lugar, se construyó la base de datos utilizando una Raspberry Pi 4 MB y una PiCamera. Con estos dispositivos se capturaron imágenes de los potes en distintas posiciones. Estas imágenes se clasificaron en dos categorías, fueron sometidas a aumento de datos *(data-augmentation)* y se dividieron en conjuntos de entrenamiento y validación.

Posteriormente, se entrenó un modelo basado en EfficientNetB0, aplicando las técnicas de aprendizaje por transferencia *(transfer learning)* y ajuste fino *(fine tuning)* en las capas superiores. Este procedimiento se realiza a través de un aprendizaje supervisado ya que el modelo aprende a partir de ejemplos previamente clasificados.

En la tercera etapa, se implementó un programa de prueba con una interfaz gráfica para verificar el funcionamiento del modelo con imágenes cargadas manualmente. Esta herramienta se utilizó como paso previo de verificación antes de su integración con el hardware.

Finalmente, se definió la puesta en marcha del sistema sobre la cinta transportadora. Esta etapa incluyó la detección de potes mediante sensor infrarrojo, captura automática de fotografías, predicción del modelo y comunicación de resultados hacia el PLC en un esquema cliente-servidor. Las señales se replicaron en un conjunto de LEDs conectados a la Raspberry Pi, lo que facilitó la visualización de los eventos.


*Figura 1: Esquemático del desarrollo del sistema de clasificación de potes*

*2.1. Diseño del Soporte*

Se construyó un único soporte para los dispositivos *Raspberry Pi 4 Modelo B* y *PiCamera Rev. 1.3*. El diseño de basó en planos oficiales del fabricante y consideró los siguientes criterios: (i) Ranuras de ventilación en la parte inferior para la Raspberry Pi; (ii) El soporte de la cámara funciona como una "tarjeta" que se inserta en el soporte inferior, lo que facilita su manipulación; (iii) Una ranura en la parte superior para la salida de la cámara; (iv) Una tapa para proteger el cuerpo de la cámara.

En las Figuras 2 y 3 se muestra el ensamblaje en Autodesk Inventor y el soporte impreso en 2D ya montado en la Raspberry Pi.

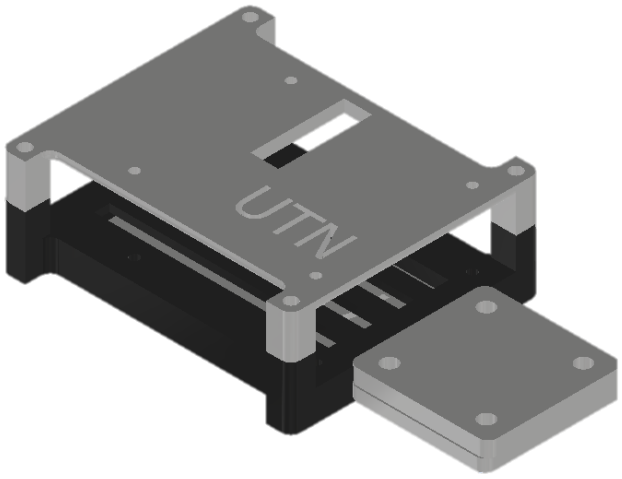


*Figura 2: Ensamble del soporte de Raspberry Pi en Autodesk Inventor*

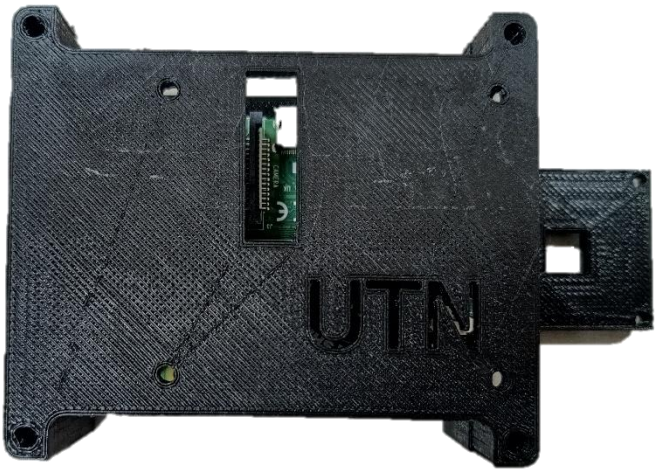


*Figura 3: Soporte realizado en Impresión 3D*

*2.2. Base de datos*

El sistema clasifica potes de helado transportados en la cinta en dos categorías: *"bien posicionados"* o *"mal posicionados"*. Para pasar al proceso de tapado, los potes deben estar centrados en la banda, con su boca hacia arriba y sin inclinaciones. Esta condición asegura que puedan ingresar al mecanismo de tapado correctamente.

Para crear la base de datos se capturaron 158 fotografías con un teléfono celular, divididas en las dos categorías mencionadas. Posteriormente, se aplicó la técnica de aumento de datos (Majida Kazmi, 2023), que consiste en ampliar artificialmente la cantidad de imágenes para el entrenamiento. El procedimiento se programó en Python 3.9.13 con la librería de TensorFlow.

En total, se aplicaron seis transformaciones aleatorias: cambios de iluminación, ruido, contraste, saturación, tonalidad y rotación. Las imágenes generadas se guardaron junto a las originales, lo que expandió la base a 948 fotografías.

La aplicación de esta técnica previene el sobreajuste *(overfitting)* del modelo. Esto se debe al incremento en la cantidad y variabilidad de los datos. Además, las transformaciones aumentan la robustez del modelo, haciéndolo menos sensible a las imperfecciones en las fotografías.

*2.3. Selección del modelo de entrenamiento*

Las redes neuronales convolucionales (CNN) (Zhao y otros, 2024) son un tipo de arquitectura de aprendizaje profundo diseñada para procesar datos en forma de imágenes. Estas utilizan capas convolucionales que extraen las características o patrones de los datos de entrada, como bordes, texturas o formas.

En modelos anteriores, los hiperparámetros de una red neuronal (ancho, profundidad y resolución) se escalaban de forma independiente para aumentar la exactitud, lo que incrementaba el costo computacional. Con EfficientNet (Mingxing Tan, 2019) se descubrió que escalar los tres parámetros de forma balanceada ofrece una mayor precisión con un costo similar. Este método se denominó escalado compuesto (*compound scaling method*).

EfficientNetB0 es la primera versión de esta familia. Su arquitectura combina capas de convolución inicial, bloques MBConv con conexiones residuales y convoluciones separables. Finalmente, incluye capas de batch nomalization, average pooling y una densa de clasificación.

En este proyecto se utilizó el aprendizaje por transferencia congelando las capas iniciales y medias para conservar las características generales de la red. Posteriormente, se aplicó la técnica de ajuste fino en las capas superiores, donde se ajustaron los pesos para especializar el modelo en la clasificación de potes de helado. Estas ventajas lo convierten en una opción adecuada para dispositivos de recursos limitados, como la Raspberry Pi, donde es necesario optimizar tanto la precisión del modelo como el consumo computacional.

### *2.4. Entrenamiento del modelo*

El entrenamiento del modelo se llevó a cabo utilizando las librerías PIL y TensorFlow en Python. Dado que las imágenes provenían de un dispositivo móvil, fue necesario eliminar la información contenida en el *Exchangeable Image Format (EXIF)*, que contiene metadatos sobre la orientación de la imagen. Esta información provocaba que, al cargar el conjunto de datos, las imágenes se rotaran automáticamente, generando problemas durante el proceso de entrenamiento.

Una vez corregido este inconveniente, se procedió a cargar la base de datos correspondiente a cada categoría, reservando un 20 % de las muestras para el proceso de validación. Antes del entrenamiento, se realizó una verificación visual de las primeras nueve imágenes de la base de datos para corroborar que la orientación se corrigió correctamente y se definió una función de aumento de datos en memoria (*on-the-fly data augmentation*) lo que permitió incrementar la variabilidad de las muestras sin necesidad de generar ni almacenar nuevas imágenes en las carpetas.

Los datos de entrada son redimensionados a una resolución de 224x224 píxeles, requerida por el modelo seleccionado, y se procesaron en lotes (*batches*). Para mejorar la eficiencia del entrenamiento, se implementaron técnicas de procesamiento paralelo de las imágenes, almacenamiento en caché y precarga de lotes en segundo plano, lo que aceleró significativamente el entrenamiento y optimizó el uso de memoria.

Se aplicó aprendizaje por transferencia con pesos preentrenados y ajuste fino en las capas superiores. El entrenamiento se ejecutó durante 10 épocas *(epochs)*. Finalmente, se guardaron los pesos del modelo para reutilizarlos en pruebas posteriores.

### *2.5. Programa de prueba para el modelo*

Previo a la puesta en marcha del sistema en la cinta transportadora, se desarrolló un programa de prueba con el paquete Tkinter que permite realizar el testeo del modelo con fotografías de los potes. La interfaz se muestra en la Figura 4.

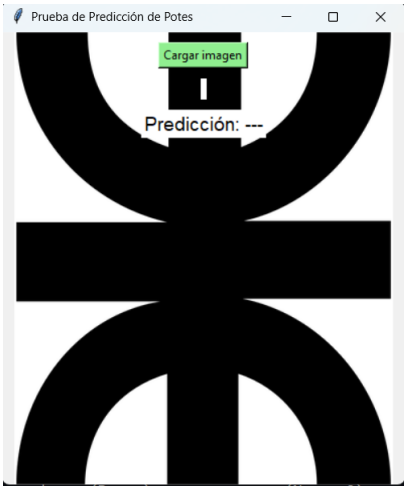


*Figura 4: Interfaz de prueba para el modelo*

Esta permite seleccionar imágenes almacenadas, mostrar la fotografía y visualizar al resultado de la clasificación, lo que facilitó verificar los pesos entrenados, el preprocesamiento de imágenes y la coherencia entre la fotografía y la etiqueta resultante de la predicción.

Una vez validado el funcionamiento del sistema de predicción, este puede trasladarse a la Raspberry Pi e integrarse con la cinta transportadora.

## 3. Resultados

Durante el entrenamiento, se registró la evolución de la exactitud tanto en el conjunto de entrenamiento en cada época. En la Figura 5 se visualiza la gráfica resultante.

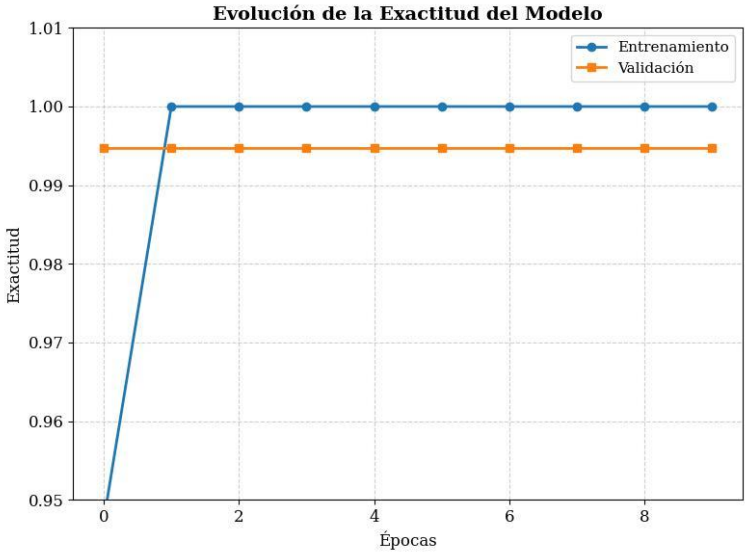


*Figura 5: Gráfica de exactitud por época de entrenamiento*

Durante el entrenamiento, el modelo alcanzó una exactitud cercana al 100% desde las primeras épocas, manteniéndose estable en ese valor. En paralelo, la exactitud sobre el conjunto de validación se mantuvo alrededor del 99% sin mostrar caídas ni fluctuaciones. La estabilidad en ambas curvas refleja una adecuada generalización del modelo, sin indicios de sobreajuste.

Además, en la Figura 6 se pueden ver los resultados de la predicción en el programa de prueba desarrollado. Estos son satisfactorios lo que permite continuar con la puesta en marcha final del sistema.


*Figura 6: Resultado en el programa de verificación*

## 4. Conclusiones

En este trabajo se implementó un sistema de clasificación de imágenes aplicando técnicas de aumento de datos y optimización durante el entrenamiento.

Las simulaciones y pruebas realizadas en el entorno de entrenamiento fueron satisfactorias, evitando el sobreajuste y reduciendo errores en el reconocimiento. Esto demuestra la viabilidad del sistema para su futura puesta en marcha en la cinta transportadora.

La etapa de integración con la cinta aún no se realizó y forma parte de los próximos pasos del proyecto. En particular, se considera desarrollar un programa que capture nuevas fotografías mediante un pulsador en una protoboard. Dichas imágenes conformarán una nueva base de datos obtenida directamente desde la cinta, aplicando nuevamente la técnica de aumento de datos.

El sistema completo se conectará con un PLC mediante el paquete pymodbus, incluyendo un sensor infrarrojo para la detección de potes y LEDs de notificación. El procedimiento propuesto contempla la detección del pote, la detención de la cinta, la captura de la imagen, la ejecución de la predicción y, en función del resultado, el avance hacia el proceso de tapado o descarte.

Este plan de puesta en marcha constituye la línea de trabajo inmediata. Además, se plantean desarrollos futuros de otros sistemas embebidos para su implementación en líneas de producción industriales, como la integración de cámaras termográficas para monitorear en tiempo real la temperatura de preformas, evaluando su aptitud antes del proceso de moldeo.

## Referencias Bibliográficas


Halfacree, G. (2020). *The official Raspberry Pi Beginner's Guide.* Raspberry Pi Press.

Majida Kazmi, B. H. (2023). *A deep learning-based framework for visual inspection of plastic bottles*, 5. https://doi.org/https://doi.org/10.1109/ACCESS.2023.3329565

Mingxing Tan, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *International Conference on Machine Learning*, 2-6. https://doi.org/https://doi.org/10.48550/arXiv.1905.11946

TensorFlow Developers. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. https://www.tensorflow.org/about/bib?hl=es-419

Zhao, X., Wang, L., Zhang, Y., Han, X., & Parmar, M. D. (2024). A review of convolutional neural networks in computer vision. *Aritf Intell Rev 57*, 1-15. https://doi.org/https://doi.org/10.1007/s10462-024-10721-6