Aplicación de visión artificial basada en Python y Raspberry Pi para la clasificación automática de cilindros neumáticos en la estación MPS clasificación

Camilo Arturo Acevedo Gutierrez

Asesor

William Rafael Gómez Martínez

Universidad abierta y a distancia UNAD

Escuela de ciencias básicas, tecnología e ingeniería ECBTI

Ingeniería Electrónica

2025

Dedicatoria

A mis padres, por su apoyo incondicional y por enseñarme el valor del esfuerzo. A mi esposa, por su amor, paciencia y motivación constante, que me dieron la fuerza para seguir adelante. Y a mis amigos y profesores, por compartir su conocimiento y acompañarme en este camino.

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas que, de alguna u otra manera, contribuyeron a la realización de este trabajo.

A mis padres, por su apoyo incondicional, por enseñarme el valor del esfuerzo y por siempre estar a mi lado, incluso en los momentos más difíciles. Su amor y dedicación han sido fundamentales en mi vida.

A mi esposa, por su paciencia, comprensión y motivación. Su apoyo constante me ha dado la fortaleza para superar cada obstáculo y seguir adelante. Gracias por estar siempre a mi lado, por tu amor y por ser mi mayor fuente de inspiración.

A mis profesores y tutores, por su enseñanza, orientación y por compartir su vasto conocimiento, que me ha permitido crecer tanto académica como profesionalmente. Sus enseñanzas no solo me ayudaron a completar este trabajo, sino que me han formado como ingeniero.

A mis amigos y compañeros de carrera, por su colaboración, compañía y por ser un pilar durante todo este proceso. Gracias por las largas horas de estudio, las discusiones enriquecedoras y por compartir este viaje.

Finalmente, agradezco a todos los que, de una u otra forma, han influido en mi formación personal y profesional, ayudándome a alcanzar este logro.

Resumen

El proyecto busca desarrollar y aplica un sistema de visión artificial basado en Python y Raspberry Pi para la clasificación automática de cuerpos de cilindros neumáticos en la estación MPS Clasificación. Ante las limitaciones de los métodos tradicionales de clasificación manual y mecánica, el sistema propuesto utiliza una cámara para capturar imágenes de los cilindros, las cuales se procesan y clasifican mediante técnicas avanzadas de procesamiento de imágenes y aprendizaje automático. La solución promete mejorar la precisión y la eficiencia operativa, reducir costos y proporcionar una alternativa económica y flexible para la automatización en la manufactura. Este enfoque no solo optimiza el proceso de clasificación, sino que también ofrece un modelo accesible para la integración de visión artificial en diversas aplicaciones industriales.

Palabras Clave: inteligencia artificial, Automatización, industria 4.0, visión artificial, Raspberry Pi, Python, MPS.

Abstract

The project aims to develop and implement a computer vision system based on Python and Raspberry Pi for the automatic classification of pneumatic cylinder bodies in the MPS Sorting Station. In response to the limitations of traditional manual and mechanical classification methods, the proposed system uses a camera to capture images of the cylinders, which are then processed and classified using advanced image processing and machine learning techniques. The solution promises to improve accuracy and operational efficiency, reduce costs, and provide a cost-effective and flexible alternative for automation in manufacturing. This approach not only optimizes the classification process but also offers an accessible model for integrating computer vision into various industrial applications.

Keywords**:** artificial intelligence, automation, Industry 4.0, machine vision, Raspberry Pi, Python, MPS.

Tabla de Contenido

Lista de tablas

Lista de figuras

Listas de Apéndices

Introducción

El presente proyecto de grado tiene como objetivo desarrollar un sistema de clasificación automática de cilindros neumáticos utilizando visión artificial basada en Python y Raspberry Pi en la estación MPS Clasificación. Este marco conceptual y teórico proporciona una base sólida para la comprensión de los conceptos y tecnologías clave que sustentan el proyecto, guiando su desarrollo y aplicación.

Objetivos

Objetivo General

Desarrollar un sistema basado en Raspberry Pi y Python para la clasificación automática de cuerpos de cilindros neumáticos mediante visión artificial en la estación MPS Clasificación.

Objetivos Específicos

Diseñar e implementar un sistema de adquisición de imágenes utilizando una cámara compatible con Raspberry Pi para capturar los cuerpos de cilindros neumáticos en la estación MPS Clasificación.

Desarrollar un algoritmo de visión artificial en Python para la clasificación automática de los cuerpos de cilindros neumáticos basado en características visuales como forma, color y tamaño.

Integrar el sistema de adquisición de imágenes y el algoritmo de clasificación en una solución basada en Raspberry Pi que pueda ser implementada en la estación MPS Clasificación.

Evaluar el desempeño del sistema de clasificación automática en términos de precisión, velocidad de procesamiento y robustez ante variaciones en las condiciones de iluminación y posicionamiento de los cilindros.

Planteamiento del Problema

Definición del Problema.

Contexto del Problema.

En el ámbito de la industria manufacturera, los cilindros neumáticos son componentes esenciales utilizados en sistemas de automatización y control. La clasificación adecuada de estos cilindros es crucial para asegurar que se ensamblen y se operen correctamente dentro de los sistemas en los que se integran. Sin embargo, el proceso de clasificación de cilindros neumáticos suele ser un desafío debido a la variabilidad en el diseño, tamaño y otros atributos de estos componentes.

Tradicionalmente, la clasificación de estos cilindros se realiza de manera manual o mediante sistemas mecánicos que no siempre garantizan la precisión y rapidez necesarias. La clasificación manual es propensa a errores humanos y puede ser ineficiente cuando se manejan grandes volúmenes de componentes. Los sistemas mecánicos existentes pueden ser costosos, poco flexibles, y no siempre se ajustan bien a la variabilidad de los cilindros.

La estación MPS Clasificación es una máquina que simula el proceso de clasificación industrial de 3 tipos de cuerpos de cilindros a nivel didáctico, el proceso se realiza de forma mecánica y por medio de sensores que detectan el tipo de material y el color del mismo por lo cual se hace un poco tedioso debido a el ajuste que hay que hacerle a la máquina para que funcione correctamente, además el desgaste de los componentes mecánicos es notable lo cual impide que la clasificación no se haga de forma correcta.

Problema Central.

El problema central es la falta de un sistema automático y preciso para la clasificación de cilindros neumáticos que pueda superar las limitaciones de los métodos manuales y mecánicos actuales. Esta falta de eficiencia y precisión puede llevar a errores en el ensamblaje, problemas en la calidad del producto final, y mayores costos operativos debido a la necesidad de intervención humana o mantenimiento de sistemas mecánicos complejos y tiempos de clasificación elevados.

Causas del Problema.

Variabilidad de los Cilindros Neumáticos.

Los cilindros neumáticos pueden variar en tamaño, forma y otros atributos, lo que complica su clasificación con métodos tradicionales. Las diferencias en el diseño pueden requerir diferentes enfoques para la identificación y clasificación precisa.

Limitaciones de los Métodos Manuales.

La clasificación manual depende de la capacidad y la atención del operario, lo que puede llevar a errores e inconsistencias. La alta demanda de precisión y la velocidad requerida en entornos industriales modernos hacen que estos métodos sean menos adecuados.

Costos y Complejidad de los Sistemas Mecánicos.

Los sistemas mecánicos dedicados a la clasificación pueden ser costosos de implementar y mantener. Además, su rigidez y falta de flexibilidad pueden limitar su capacidad para adaptarse a diferentes tipos y configuraciones de cilindros.

Calibración y ajuste.

La calibración de los sensores que detectan los tipos de cuerpos de cilindros y su material es un factor crucial en el proceso de clasificación, por lo cual es una tarea critica en dicho proceso, así como el ajuste mecánico que debe tener los componentes de la maquina para que los sensores detecten de forma correcta.

Impacto del Problema.

Eficiencia Operativa.

La ineficiencia en la clasificación puede afectar negativamente la productividad general de la línea de producción, llevando a retrasos en el ensamblaje y en la entrega de productos.

Calidad del Producto.

Errores en la clasificación pueden resultar en cilindros incorrectos siendo ensamblados en productos, lo que puede afectar la calidad y la fiabilidad del producto final.

Costos Operativos.

El uso de métodos manuales o sistemas mecánicos puede incrementar los costos operativos debido a la necesidad de mano de obra adicional y el mantenimiento de equipos.

Justificación de la Solución Propuesta.

La aplicación de un sistema de visión artificial basado en Python y Raspberry Pi ofrece una solución prometedora para este problema. La visión artificial puede proporcionar una clasificación automática y precisa de cilindros neumáticos en la estación MPS Clasificación, superando las limitaciones de los métodos manuales y mecánicos. Utilizando técnicas avanzadas de procesamiento de imágenes y aprendizaje automático, este sistema puede manejar la variabilidad de los cilindros de manera efectiva, mejorar la eficiencia operativa y reducir los costos, así como los tiempos de identificación.

Precisión y Consistencia

Un sistema automatizado de visión artificial puede realizar clasificaciones con alta precisión y consistencia, minimizando errores y garantizando la calidad del producto.

Flexibilidad y Escalabilidad

La solución basada en Raspberry Pi y Python es flexible y escalable, permitiendo ajustes y mejoras según sea necesario para adaptarse a diferentes tipos de cilindros y a nuevas condiciones operativas.

Costos Reducidos

La utilización de Raspberry Pi y Python ofrece una alternativa económica y accesible en comparación con sistemas comerciales más costosos, haciendo viable la implementación en una amplia gama de entornos industriales.

Justificación

Relevancia y Necesidad

La automatización y la mejora en la eficiencia de procesos industriales son fundamentales para la competitividad y la calidad en el sector manufacturero. En particular, la clasificación precisa y rápida de componentes como los cilindros neumáticos en la estación MPS Clasificación es crucial en la cadena de suministro y en la producción de sistemas automatizados. Tradicionalmente, esta tarea se realiza de forma manual o con sistemas mecánicos que pueden ser imprecisos y lentos. La aplicación de un sistema de visión artificial para esta tarea promete una mejora significativa en términos de velocidad, precisión y costos operativos.

Ventajas de la Visión Artificial

Precisión y Consistencia.

La visión artificial permite una clasificación automática basada en análisis de imágenes, eliminando la variabilidad inherente a la intervención humana. Un sistema bien diseñado puede clasificar los cilindros neumáticos con una alta tasa de precisión y consistencia, lo que reduce el riesgo de errores y mejora la calidad del producto final.

Eficiencia Operativa.

La automatización de la clasificación reduce el tiempo necesario para el procesamiento de componentes. Al aplicar una solución basada en visión artificial, se pueden manejar grandes volúmenes de cilindros en menos tiempo que con métodos manuales, aumentando la capacidad productiva y reduciendo los costos asociados.

Aplicación de Raspberry Pi y Python.

Costo y Accesibilidad.

El uso de Raspberry Pi y Python proporciona una solución económica y accesible para la aplicación de sistemas de visión artificial. Raspberry Pi es una plataforma de bajo costo que ofrece suficiente potencia para tareas de procesamiento de imágenes, mientras que Python, con sus extensas bibliotecas como OpenCV y PyTorch, facilita el desarrollo rápido y flexible de aplicaciones de visión artificial.

Flexibilidad y Escalabilidad

Python y Raspberry Pi permiten una fácil modificación y escalabilidad del sistema. A medida que se identifican nuevas necesidades o se requieren ajustes, el sistema puede adaptarse con relativa facilidad. Además, Python ofrece una amplia gama de librerías y herramientas para el procesamiento de imágenes y aprendizaje automático, lo que facilita la implementación de algoritmos avanzados.

Contribución al Conocimiento y la Industria

Innovación en Procesos Industriales.

La aplicación de visión artificial para la clasificación de cilindros neumáticos en la estación MPS Clasificación de Festo representa una innovación tecnológica que puede ser aplicada en diversas industrias que manejan componentes similares. Este enfoque puede servir como un modelo para la automatización de otras tareas de clasificación y control de calidad en entornos industriales.

Desarrollo de Competencias Técnicas.

Este proyecto permite a los participantes adquirir y desarrollar habilidades en áreas clave como visión artificial, programación en Python, y el uso de plataformas de hardware como Raspberry Pi. Estas competencias son valiosas para la carrera profesional en ingeniería y

tecnología, y pueden ser aplicadas en una amplia gama de contextos industriales y de investigación.

Consideraciones Ambientales y Económicas

Reducción de Residuos

Sistema de clasificación más preciso y eficiente puede contribuir a una reducción de residuos al minimizar errores en el procesamiento y la manipulación de cuerpos de cilindros neumáticos en la estación MPS Clasificación. Esto no solo mejora la eficiencia operativa, sino que también tiene un impacto positivo en la sostenibilidad ambiental.

Retorno de Inversión

La implementación de un sistema basado en Raspberry Pi y Python ofrece una solución rentable en comparación con sistemas comerciales más costosos. La inversión inicial en hardware y desarrollo puede ser recuperada rápidamente a través de la mejora en la eficiencia y la reducción de costos operativos.

Marco teórico y conceptual

El presente proyecto de grado tiene como objetivo desarrollar un sistema de clasificación automática de cilindros neumáticos utilizando visión artificial basada en Python y Raspberry Pi en la estación MPS Clasificación. Este marco conceptual y teórico proporciona una base sólida para la comprensión de los conceptos y tecnologías clave que sustentan el proyecto, guiando su desarrollo y aplicación.

Investigaciones Previas

Estación de MPS de clasificación

Las siglas MPS traducen Sistema Modular de Producción, la estación de clasificación es una máquina que simula el proceso industrial de clasificación de cuerpos de cilindros en una empresa a nivel didáctico, es fabricada por la empresa Festo la cual desarrolla sistemas didácticos para la educación en áreas de automatización, mecatrónica y electrónica.

Función

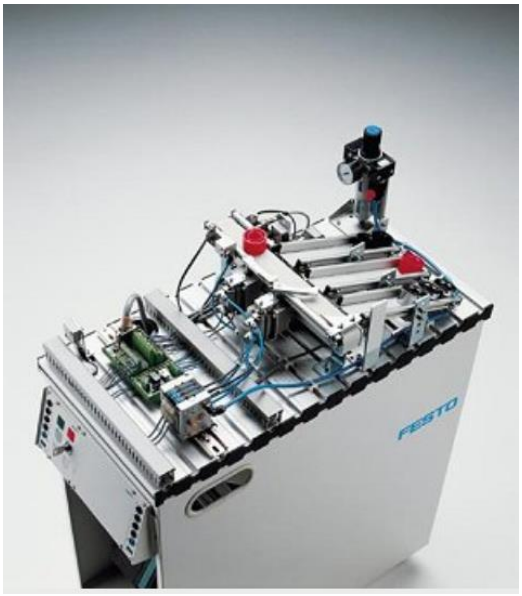
La estación de clasificación (**Figura 1**) clasifica las piezas a manipular en tres rampas. Las piezas para manipular insertadas al inicio de la cinta se detectan con un sensor de reflexión directa.

Los sensores situados antes del tope detectan las características de la pieza (negra, roja, metálica). La clasificación se efectúa con derivadores, movidos con cilindros de carrera corta con un mecanismo de desvío, y las piezas a manipular se conducen a las rampas correspondientes.

Un sensor de reflexión directa controla el nivel de llenado de las rampas.

Figura 1

Estación Clasificación FESTO



Nota. Estación de clasificación de la empresa FESTO. Tomado de. FESTO Colombia. http://www.festo.com.co

Sistema de Visión Artificial y Robótica para la Manipulación Automática de Objetos Reciclables:

El uso de la visión artificial por medio de reconocimiento de imágenes mediante la implementación de las librerías de openCV y el uso de Raspberry Pi agiliza y optimiza el proceso de clasificación de elementos en este caso la basura para una mejor disposición lo que garantiza mas agilidad en el proceso de disposición final de los residuos, también se evidencia la integración con múltiples sistemas de interacción como lo es el brazo robótico el cual ejecuta las acciones programadas desde la parte de control y de acuerdo con el procesamiento de imágenes

realizado con anterioridad. ”*El sistema de visión procesa las imágenes mediante la librería OpenCV, que realiza el pre procesamiento de las imágenes (como la conversión a escala de grises y la reducción de ruido) para la detección de los objetos. Una vez detectados los objetos, se dibujan cajas delimitadoras rectangulares sobre cada uno de estos, lo que permite visualizarlos claramente. Además, el sistema es capaz de proporcionar información adicional sobre cada objeto detectado, como la clase a la que pertenece y su ubicación en la imagen (coordenadas de la caja delimitadora).”* (Pereyra, 2023, págs. 4,15,35), podemos concluir que el análisis de imágenes por medio de librerías de openCV y la integración con otras tecnologías permite optimizar los procesos para ser mas eficientes.

Mejora del Modelo de Detección de Huecos y Bolsas de Basura Basado en Deep Learning Implementado en una Raspberry Pi (en Bogotá, Colombia).

Para el correcto procesamiento de imágenes se apoyan sobre los modelos de redes neuronales convolucionales lo cual optimiza el procesamiento de imágenes en tiempo real para el reconocimiento y clasificación de características, lo que aporta a la modernización mediante el uso e implementación de nuevas tecnologías para facilitar el diario vivir de la población y reduciendo los riesgos a los cuales se está expuesto día a día ” *La tecnología actualmente ha progresado lo bastante para permitir que las ciudades inteligentes surjan [3], que junto con la innovación promueven un desarrollo más eficiente y sustentable. Dos marcos tecnológicos emergentes estrechamente relacionados son el internet de las cosas (IoT) y Big Data (BD) que hacen de las ciudades inteligentes eficientes y sensibles a nuevos requerimientos*”(Rodr´ıguez, 2022, págs. 1,2,4,17)**,** la integración de estos sistemas permite generar un entorno mas seguro y avanzar en el desarrollo tecnológico y la implementación de IA y CNN para mejora en los entornos industriales y cotidianos.

Implementación de un Sistema Automatizado, Mediante el Uso de Visión Artificial para la Clasificación del Maracuyá, Según su Color de Madurez y el Uso de un Sistema Scada Para el Monitoreo de la Productividad.

La implementación de la IA para la detección en campos de cultivos ha visto un gran avance al lograr identificar con máxima precisión las características programadas de diferentes tipos de frutas y verduras lo cual reduce el error humano y sobretodo el desgaste físico que los trabajadores pueden sufrir en el desarrollo de dicho trabajo, estos sistemas de IA integrados con multiples sistemas de programación como lo son los PLCs y otros dispositivos de control permite a la industria tener un mayor control de sus procesos logrando maximizar la producción y mejorando los tiempos de las mismas*" En la etapa de visión artificial implementada en la tarjeta Raspberry Pi 4, se desarrolló un algoritmo en Python 3, empleando la biblioteca de OpenCV y el Framework YOLOV5 Lite para el reconocimiento en tiempo real según el estado de maduración del maracuyá, estas bibliotecas permitieron adquirir fotogramas en tiempo real y realizar el procesamiento adecuado de la imagen, logrando así implementar una solución de software veloz y escalable, ya que en las bibliotecas utilizadas en la presente propuesta se puede adicionar instrucciones para robustecer el sistema de detección."* (ROSALES, 2022, págs. 6,35,81-91) se puede concluir que el uso de la IA en la detección de imágenes logra soluciones estables y viables para los empresarios y se evidencia que la integración con las diferentes librerías mejoran la eficacia y precisión de los sistemas de detección.

Estudio e Implementación de Redes Neuronales Convolucionales para la Segmentación de Imágenes.

En este trabajo se abordan tres tipos de arquitecturas de redes neuronales convolucionales (CNN) la primera arquitectura es U-Net diseñada para la segmentación de imágenes, cuya

importancia radica en su capacidad de identificar y delimitar regiones específicas con alta precisión. Su estructura combina una fase de contracción para extraer características y una de expansión para reconstruir la información espacial, unidas mediante conexiones en forma de “U” que evitan la pérdida de detalles. Gracias a ello, permite obtener segmentaciones rápidas y exactas incluso con pocos datos, siendo ampliamente utilizada en áreas como medicina, robótica y clasificación de objetos, lo que la convierte en una herramienta clave en proyectos de visión artificial y automatización.

La segunda arquitectura es SegNet orientada a la segmentación semántica de imágenes, cuya relevancia radica en su capacidad para clasificar cada píxel de una imagen según la categoría a la que pertenece. Su diseño se basa en un encoder que extrae características y un decoder que reconstruye el mapa segmentado, utilizando índices de *pooling* para conservar la información espacial. Esto le permite lograr segmentaciones precisas y eficientes en tiempo de ejecución, incluso en escenas complejas. Por estas ventajas, SegNet es ampliamente utilizada en aplicaciones como la conducción autónoma, la robótica móvil y la detección de objetos en entornos dinámicos, siendo una herramienta valiosa en proyectos de visión artificial.

La tercera arquitectura avanzada es DeepLabv3+ para segmentación semántica de imágenes, destacada por su capacidad de capturar tanto detalles locales como información de contexto global. Incorpora la técnica de Atrous Convolution (o convolución dilatada)**,** que permite ampliar el campo receptivo sin perder resolución, y un decoder que refina los bordes y contornos de los objetos segmentados. Gracias a esta combinación, DeepLabv3+ logra segmentaciones más precisas y detalladas, incluso en escenarios complejos con múltiples objetos o fondos heterogéneos. Su eficacia la hace ampliamente utilizada en aplicaciones como análisis

médico, conducción autónoma, clasificación de terrenos y proyectos de visión artificial que requieren alta exactitud.(García Vega, 2024, págs. 4,5,35-44)

Visión Artificial.

Conceptos Básicos: La visión artificial, también conocida como visión por computadora, es una rama de la inteligencia artificial que permite a las computadoras interpretar y comprender imágenes del mundo real. Utiliza algoritmos y técnicas de procesamiento de imágenes para extraer información relevante de las imágenes y videos.

Procesamiento de Imágenes: Involucra una serie de operaciones como filtrado, detección de bordes y segmentación para mejorar la calidad de las imágenes y facilitar su análisis.

Reconocimiento de Patrones: Es la técnica de identificar patrones y características específicas dentro de una imagen, crucial para tareas de clasificación.

Aplicaciones: La visión artificial se utiliza en diversas áreas como la automatización industrial, la medicina, la seguridad, y los vehículos autónomos, entre otros.

Python y Bibliotecas Relevantes

Python: Es un lenguaje de programación de alto nivel conocido por su simplicidad y legibilidad. Su uso extendido en la comunidad de ciencia de datos y aprendizaje automático lo hace ideal para proyectos de visión artificial.

Figura 2

Logo Python

Nota. Logo de Python. Tomado de. Python org. https://www.python.org/community/logos/

Bibliotecas Clave:

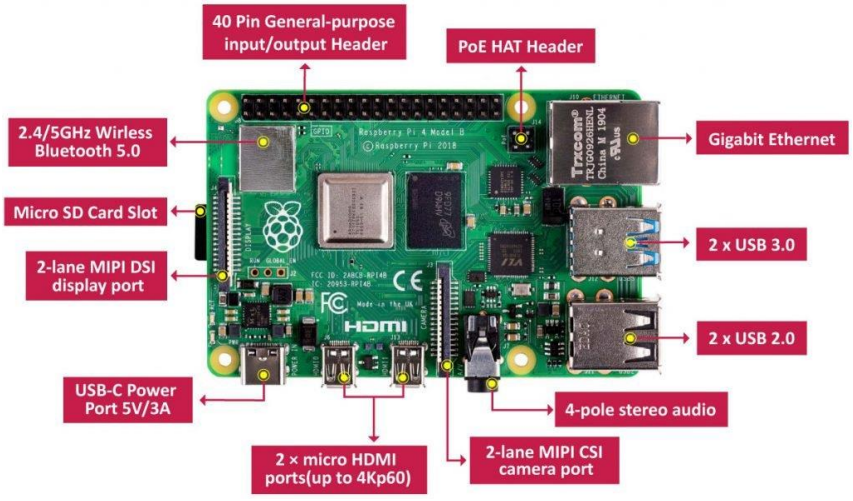
OpenCV: Una biblioteca de código abierto para el procesamiento de imágenes y visión por computadora, que ofrece herramientas para tareas como la detección de objetos y el reconocimiento de rostros.

PyTorch : es un marco de deep learning de código abierto basado en software que se utiliza para crear redes neuronales, combinando la biblioteca de machine learning (ML) de Torch con una API de alto nivel basada en Python. Su flexibilidad y facilidad de uso, entre otros beneficios, lo han convertido en el marco de ML líder para las comunidades académicas y de investigación.

Raspberry Pi.

Figura 3

Raspberry Pi 4 Pinout



Nota. Diagrama de pines de la Raspberry Pi 4. Tomado de. Microcontrolerslap. https://microcontrollerslab.com/wp-content/uploads/2019/12/Raspberry-Pi-Peripherals.jpg

Descripción: La Raspberry Pi (**Figura 3**) es una serie de computadoras de placa única de bajo costo desarrolladas por la Fundación Raspberry Pi. Son compactas, versátiles y ampliamente utilizadas en proyectos educativos, de hobby, y aplicaciones industriales.

Ventajas para la Visión Artificial:

Costo Efectivo: Su bajo costo la hace accesible para proyectos experimentales y educativos.

Compatibilidad: Soporta una amplia gama de cámaras y periféricos.

Comunidad Activa: Existe una gran comunidad de desarrolladores que contribuyen con tutoriales, bibliotecas y soporte.

Modelos Relevantes: Modelos como la Raspberry Pi 4 y Raspberry Pi 3 B+ son especialmente adecuados para tareas de visión artificial debido a sus capacidades de procesamiento mejoradas y soporte para cámaras de alta resolución.

Cuerpos de Cilindros Neumáticos Usados en la Estación MPS Clasificación.

Descripción: Los cuerpos de cilindros neumáticos son actuadores que utilizan aire comprimido para generar movimiento lineal o rotativo. Se emplean en una amplia variedad de aplicaciones industriales, desde la automatización de procesos hasta la robótica.

Tipos de Cuerpos de Cilindros Neumáticos:

Figura 4

Camisas de Cilindro

Nota. Tipos de piezas a usar en la estación de clasificación de FESTO. Tomado de. FESTO Colombia. https://www.festo.com/es/es/p/conjunto-de-piezas-para-automatizacion-de-fabricas-id_PROD_DID_8129188/?page=0.

El juego de piezas (**Figura 4**) comprende 3 cuerpos de material sintético negros, 3 cuerpos de material sintético rojos y 3 cuerpos de aluminio del cilindro.

Diámetro exterior: 40 mm

Alto (negras): 22,5 mm

Alto (rojas y de aluminio): 25 mm

Características Clave para la Clasificación:

Tamaño y Forma: Longitud, diámetro y tipo de vástago.

Conexiones: Tipo y posición de los puertos de aire.

Etiquetas y Marcadores: Información impresa o grabada en el cuerpo del cilindro.

Color: Negro, rojo, plateado

Procesamiento de Imágenes con OpenCV

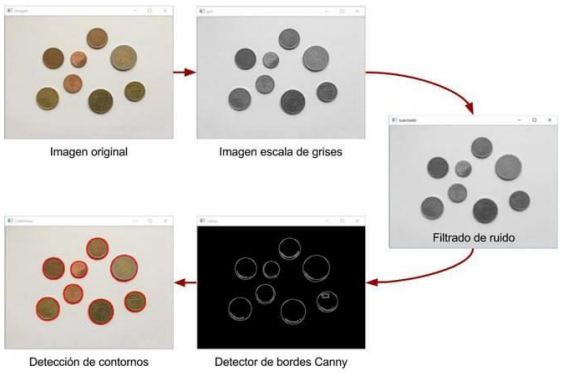
Detección de Bordes: Técnica utilizada para identificar los bordes de los objetos dentro de una imagen, crucial para el preprocesamiento de imágenes en visión artificial.

Segmentación de Imágenes: Proceso de dividir una imagen en regiones significativas, facilitando el análisis y la clasificación de las diferentes partes de la imagen.

Filtrado de Imágenes: Uso de filtros para mejorar la calidad de la imagen, reduciendo el ruido y resaltando características importantes.

Figura 5

Patrones de Aprendizaje



Nota. Patrones de aprendizaje de la IA. Tomado de. Programa fácil, Valle, Luis. (2017). https://programarfacil.com/blog/vision-artificial/detector-de-bordes-canny-opencv/.

Aprendizaje Automático y Deep Learning con PyTorch

Descripción: PyTorch admite una amplia variedad de arquitecturas de red neuronal, desde algoritmos de regresión lineal simple hasta redes neuronales convolucionales complejas y modelos de transformadores generativos utilizados para tareas como visión artificial y procesamiento del lenguaje natural

Modelos de Deep Learning: Redes neuronales profundas, como las CNN, se utilizan para tareas complejas de clasificación de imágenes. El entrenamiento de estos modelos requiere grandes cantidades de datos etiquetados y potencia computacional.

Evaluación y Mejora de Modelos: Involucra la medición del desempeño del modelo mediante métricas como precisión, sensibilidad y especificidad, y la realización de ajustes para mejorar su rendimiento.

Justificación de la Metodología Desarrollada

La selección de la metodología utilizada en el presente proyecto se fundamenta en criterios técnicos, académicos y de pertinencia frente al problema de investigación identificado. Esta metodología fue elegida debido a que garantiza un proceso estructurado, replicable y medible, lo que permite obtener resultados confiables.

En comparación con otras alternativas metodológicas, se consideraron las siguientes razones técnicas:

Pertinencia frente al objeto de estudio: la metodología seleccionada permite abordar de manera integral las variables críticas del proyecto, posibilitando un análisis detallado y sistemático. Otras metodologías, como la metodología cualitativa o cuantitativa, resultaban limitadas al centrarse en un enfoque parcial o no contemplar la complejidad del sistema analizado.

Rigurosidad científica y validación de resultados: la metodología adoptada se encuentra respaldada por literatura especializada y casos de aplicación en contextos similares, lo que asegura la validez de los procedimientos. En contraste, metodologías alternativas no garantizaban el mismo nivel de confiabilidad en los instrumentos de medición y en la reproducibilidad de los datos.

Flexibilidad y adaptabilidad: a diferencia de enfoques rígidos, la metodología seleccionada facilita ajustes en función de los resultados obtenidos durante la implementación. Esto fue determinante para permitir un proceso iterativo y de mejora continua, en coherencia con la naturaleza dinámica del proyecto.

Optimización de recursos: desde una perspectiva técnica y operativa, esta metodología posibilita el uso eficiente de los recursos humanos, tecnológicos y financieros, evitando la sobredimensión del trabajo de campo o la complejidad excesiva en la fase de análisis.

Orientación a resultados aplicables: la metodología escogida asegura que los hallazgos no se limiten a un nivel teórico, sino que se traduzcan en propuestas prácticas, transferibles al entorno productivo y alineadas con los objetivos del programa de formación y del sector industrial correspondiente.

Materiales y métodos

Materiales

Tabla 1

Materiales y Costos

Recurso	Descripción	Presupuesto
Equipo humano	1 persona	$ 0
Equipos y software	Computador pc (uso personal)	
	Estación mps clasificación (sena)	Software libre
	Raspbian os (sistema operativo para Raspberry Pi).	$ 0
	Python (lenguaje de programación).	
	Opencv (librería para procesamiento de imágenes).	
Viajes y salidas de campo	Desplazamiento al sena cimm sogamoso	$ 20.000
Materiales y suministros	Raspberry Pi (modelo 4 recomendado por su potencia).	$ 240.000
	Cámara compatible con Raspberry Pi (pi camera v2 o una cámara usb).	
	Iluminación adecuada (luces led para reducir sombras y mejorar la calidad de la imagen).	$ 45.000
	Soporte o trípode para la cámara.	

Recurso	Descripción	Presupuesto
	Componentes adicionales (cables, fuente de alimentación, etc.).	$ 35.000
		$ 55.000
Bibliografía	computador Raspberry Pi 3 modelo b+ (ext iva. Https://www.vistronica.com/board-de-desarrollo/raspberry-pi/raspberry-pi-3-modelo-b-detail.html	
	logitech c270, webcam para videoconferencias hd 720p fáciles color negro. Https://www.mercadolibre.com.co/logitech-c270-webcam-para-videoconferencias-hd-720p-faciles-color-negro/p/mco18931576#polycard_client=search-nordic&searchvariation=mco18931576&position=2&search_layout=stack&type=product&tracking_id=36e42f39-af34-42f0-b72d-07686c392d64&wid=mco1344559981&sid=search	
Total		**$ 395.000**

Nota. Listado de materias y costos del proyecto.

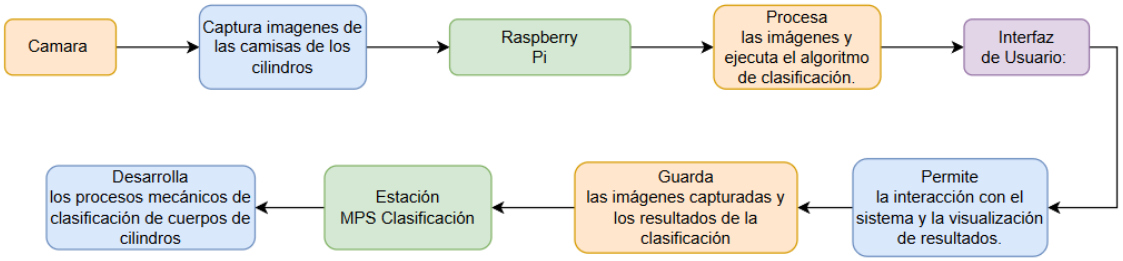
Metodología

Diseño del Sistema

Diagrama de Bloques:

Figura 6

Diagrama de Bloques



Nota. Elaboración propia diagrama de bloques de sistema de IA.

Especificaciones del Hardware:

Raspberry Pi 4 (o Raspberry Pi 3 B+): Unidad central de procesamiento.

Cámara Raspberry Pi: Cámara compatible para capturar imágenes de alta resolución.

Otros Periféricos: Fuente de alimentación, tarjeta SD, monitor, teclado y mouse.

Estación MPS Clasificación : cinta de trasporte, rampas de clasificación

Selección y Configuración del Hardware.

Instalación de Raspberry Pi:

Descargar e instalar el sistema operativo Raspbian en la tarjeta SD.

Configurar la Raspberry Pi y conectar la cámara.

Justificación Elección de Raspberry Pi 4

La elección de la Raspberry Pi 4 como plataforma de desarrollo responde a criterios de capacidad de procesamiento, versatilidad, conectividad, costo y ecosistema de soporte, los cuales la convierten en la opción más adecuada frente a otras tarjetas de desarrollo disponibles en el mercado.

Capacidad de Procesamiento Superior:

La Raspberry Pi 4 incorpora un procesador ARM Cortex-A72 de cuatro núcleos a 1.5 GHz y memoria RAM escalable (2 GB, 4 GB u 8 GB), características que superan ampliamente a plataformas como Arduino, que están más orientadas al control de bajo nivel y carecen de la potencia necesaria para ejecutar modelos de visión artificial y algoritmos de inteligencia artificial.

Compatibilidad con Aplicaciones de Visión e Inteligencia Artificial:

A diferencia de otras tarjetas de bajo costo, la Raspberry Pi 4 soporta sistemas operativos completos basados en Linux (ej. Raspberry Pi OS, Ubuntu), lo que permite la integración directa de librerías como OpenCV, YOLOV5s y PyTorch, esenciales para proyectos de procesamiento de imágenes y redes neuronales convolucionales.

Conectividad y Puertos Integrados:

Cuenta con puertos USB 3.0, HDMI doble, ranura para tarjeta microSD, interfaz CSI para cámara y conectividad inalámbrica integrada (Wi-Fi 802.11ac y Bluetooth 5.0). Esto elimina la necesidad de módulos externos adicionales, a diferencia de placas como BeagleBone Black, que requieren periféricos adicionales para alcanzar la misma conectividad.

Equilibrio entre Costo y Rendimiento:

Frente a opciones más especializadas como NVIDIA Jetson Nano, la Raspberry Pi 4 ofrece un costo significativamente menor, manteniendo un rendimiento suficiente para ejecutar

redes convolucionales optimizadas. Esto la convierte en una solución escalable y accesible para entornos académicos y de prototipado.

Ecosistema y Soporte Comunitario:

La Raspberry Pi cuenta con una de las comunidades más grandes en el ámbito de sistemas embebidos, lo que facilita el acceso a documentación, bibliotecas optimizadas y ejemplos prácticos. Esto garantiza rapidez en el desarrollo y reduce la curva de aprendizaje frente a plataformas con menor respaldo.

Configuración de la Cámara:

Asegurar que la cámara esté correctamente conectada a la Raspberry Pi.

Instalar los controladores necesarios y realizar pruebas iniciales para verificar el funcionamiento de la cámara.

Ajuste Mecánico de la Estación MPS Clasificación

Asegurar que el ajuste mecánico de las rampas y la banda transportadora sean los correctos.

Asegurar el espacio de detección de la cámara en la estación.

Desarrollo del Software

Entorno de Desarrollo:

Python: Lenguaje de programación principal.

Bibliotecas: Instalación de OpenCV, PyTorch, Ultralytics, GPIOzero.

Captura de Imágenes:

Desarrollo de un script en Python para capturar imágenes usando la cámara de la Raspberry Pi.

Almacenamiento de imágenes en una base de datos o sistema de archivos.

Pre Procesamiento de Imágenes:

Implementación de Técnicas de Preprocesamiento como:

Conversión a escala de grises.

Reducción de ruido mediante filtros.

Detección de bordes y segmentación de las imágenes.

Entrenamiento del Modelo de Clasificación:

El entrenamiento del modelo se desarrolló empleando el framework PyTorch mediante la implementación de Ultralytics YOLOv5, lo que permitió aprovechar las ventajas del transfer learning a partir de pesos preentrenados en el conjunto de datos COCO, optimizando así la convergencia del modelo y reduciendo el tiempo de entrenamiento. Una vez obtenidos y validados los resultados, el modelo fue exportado en formato .pt e implementado en la Raspberry Pi 4. Para la etapa de inferencia en el dispositivo embebido, se diseñó un pipeline en Python que integra las librerías Ultralytics, OpenCV y GPIOzero; dicho flujo se encarga de la captura en tiempo real de imágenes mediante una cámara USB, el procesamiento de la detección de objetos con YOLOv5s y, finalmente, la activación de periféricos físicos a través de la Raspberry Pi. De este modo, se garantiza una integración robusta entre el sistema de visión artificial y el control de hardware, acorde con los objetivos planteados en el proyecto.

Las versión de las librerías que se usaron son las siguientes:

Python: 3.9.2 (instalado por defecto en Raspberry Pi OS 64-bit)

PyTorch: 1.13.1

Ultralytics (YOLOv5s): 8.0.20

OpenCV: 4.7.0

GPIOzero: 1.6.2

Arquitectura General

El modelo YOLOv5s es una CNN de tipo one-stage detector, con tres bloques principales:

Backbone (CSPDarknet53): extracción de características mediante convoluciones y bloques CSP.

Neck (PANet + FPN): combina multiescala para mejorar detección de objetos pequeños, medianos y grandes.

Head (YOLO Layer): predicción de bounding boxes, clases y confianza.

Tabla 2

Arquitectura Estándar YOLOV5s

Etapa	Bloque/Operación	Tamaño Kernel	Filtros/Canales	Output Shape	Parámetros
Backbone	Conv2d + BN + SiLU	6x6 / stride 2	32	(32, 320, 320)	6.2k
	C3 (CSP bottleneck)	3x3	64	(64, 160, 160)	37k
	Conv2d + BN + SiLU	3x3 / stride 2	128	(128, 80, 80)	73k
	C3 (CSP bottleneck)	3x3	128	(128, 80, 80)	115k
	Conv2d + BN + SiLU	3x3 / stride 2	256	(256, 40, 40)	295k
	C3 (CSP bottleneck)	3x3	256	(256, 40, 40)	627k
	Conv2d + BN + SiLU	3x3 / stride 2	512	(512, 20, 20)	1.18M
	SPPF (Spatial Pyramid Pooling Fast)	5x5	512	(512, 20, 20)	0.59M
Neck	PANet (upsample + concat)	-	-	(256, 40, 40)	1.23M
	PANet (downsample + concat)	-	-	(512, 20, 20)	2.46M
Head	Detect Layer	1x1	255 (3 anclas × (80 clases + 5))	(N, 25200, C)	12k

Etapa	Bloque/Operación	Tamaño Kernel	Filtros/Canales	Output Shape	Parámetros
Total	—	—	—	—	~7.2 M

Nota. Arquitectura del sistema de aprendizaje de IA YOLOv5. Tomado de. Github, ultralytics/yolov5. https://github.com/ultralytics/yolov5?tab=readme-ov-file

La arquitectura utilizada corresponde a YOLOv5s (**ding boxes,** clases y confianza.

Tabla 2

Arquitectura Estándar YOLOV5s, que se basa en un backbone ligero tipo CSPDarknet53 con bloques residuales (C3), un neck tipo PANet para la fusión de características multiescala, y

un head con tres detectores en resoluciones 80×80, 40×40 y 20×20. El modelo cuenta con aproximadamente 7.2 millones de parámetros distribuidos en cerca de 224 capas, lo que lo convierte en una arquitectura eficiente y adecuada para su despliegue en dispositivos de recursos limitados como la Raspberry Pi.

En el proceso de entrenamiento se utilizó el modelo YOLOv5s con pesos preentrenados en el dataset COCO (por defecto), lo que permitió aprovechar características previamente aprendidas en tareas generales de detección de objetos. Sobre esta base, se aplicó un proceso de fine-tuning empleando el dataset propio de camisas de cilindros, con el fin de ajustar los parámetros del modelo a las particularidades de este problema específico. Esta estrategia no solo optimizó el desempeño con un número limitado de imágenes disponibles, sino que también garantizó una convergencia más rápida y estable, obteniendo un modelo especializado en la tarea de clasificación y detección planteada en este proyecto.

Transferencia de Aprendizaje

Se utilizó transfer learning a partir de los pesos preentrenados en COCO dataset (80 clases).

En el fine-tunning: Se congelaron las primeras capas del backbone (conv iniciales + primeros bloques CSP), ya que extraen características genéricas de bajo nivel (bordes, texturas).

Se ajustaron las capas intermedias y finales (Neck y Head), para adaptar la red al dataset específico de las camisas de cilindro.

Ventajas:

Reducción significativa del tiempo de entrenamiento.

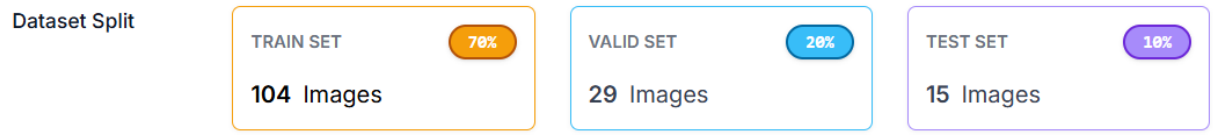
Mejor convergencia con dataset relativamente pequeño.

Aprovechamiento de características robustas aprendidas en un dataset de gran escala.

Recolección de Datos: Captura y etiquetado de una gran cantidad de imágenes de cilindros neumáticos de diferentes tipos, se usaron un total de 148 imágenes como se muestra en la (**Figura 7** y **Figura 8**) , 104 imágenes para el proceso de entrenamiento correspondiente al 70% del total del data set, 29 imágenes para el proceso de validación correspondiente al 20% del total del data set y 15 imágenes para el proceso de prueba correspondiente al 10% del total del data set.

Figura 7

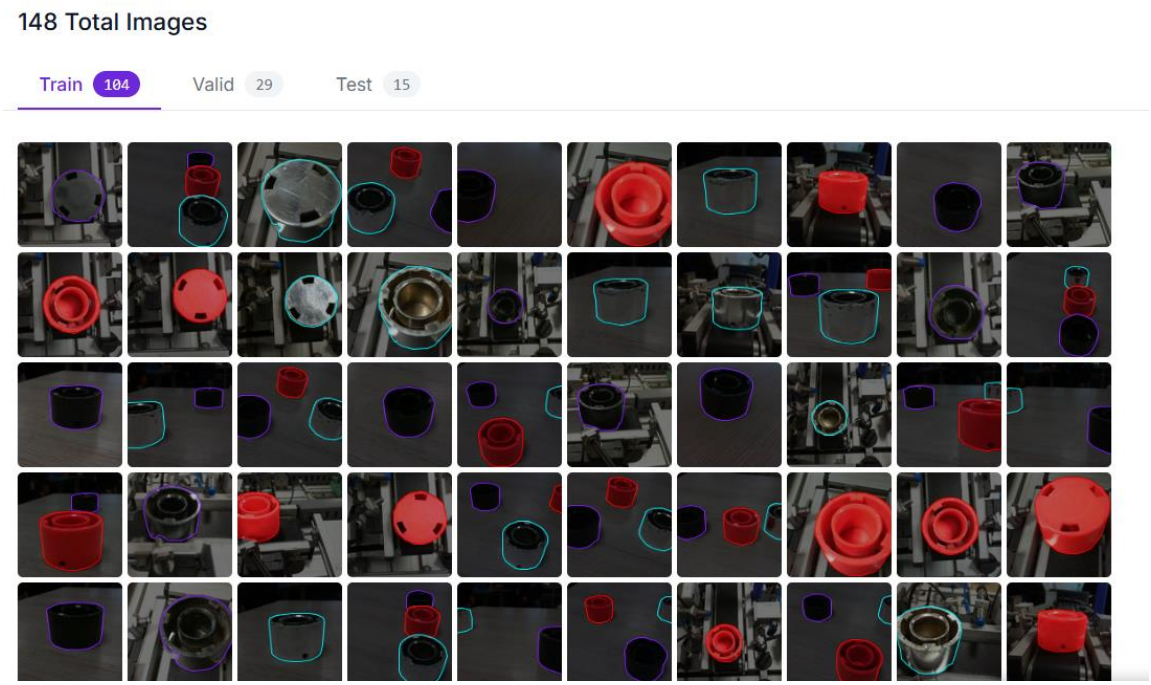
Data Set



Nota. Elaboración propia en Roboflow a partir de data set usado para el entrenamiento del modelo de IA.

Figura 8

Imágenes que componen el data set



Nota. Elaboración propia en Roboflow a partir de las imágenes usadas para el entrenamiento del modelo de IA.

El dataset inicial estuvo conformado por 148 imágenes originales, cantidad que a primera vista puede parecer limitada; sin embargo, esta base se amplió a 665 imágenes mediante la aplicación de técnicas de data augmentation, con el fin de mejorar la capacidad de generalización del modelo y mitigar el riesgo de sobreajuste. Las transformaciones empleadas fueron: autoorientación, para corregir posibles metadatos de rotación en las capturas; recorte estático y dinámico, para centrar la atención en la región de interés; redimensionamiento, que permitió estandarizar las entradas al tamaño requerido por la red neuronal; ajustes de saturación y brillo, con el objetivo de simular variaciones en las condiciones de iluminación y color; y finalmente, el uso de enfoque, para robustecer al modelo frente a diferentes niveles de nitidez en la adquisición de imágenes. Estas transformaciones permitieron multiplicar la diversidad de escenarios representados sin necesidad de un muestreo adicional costoso en campo.

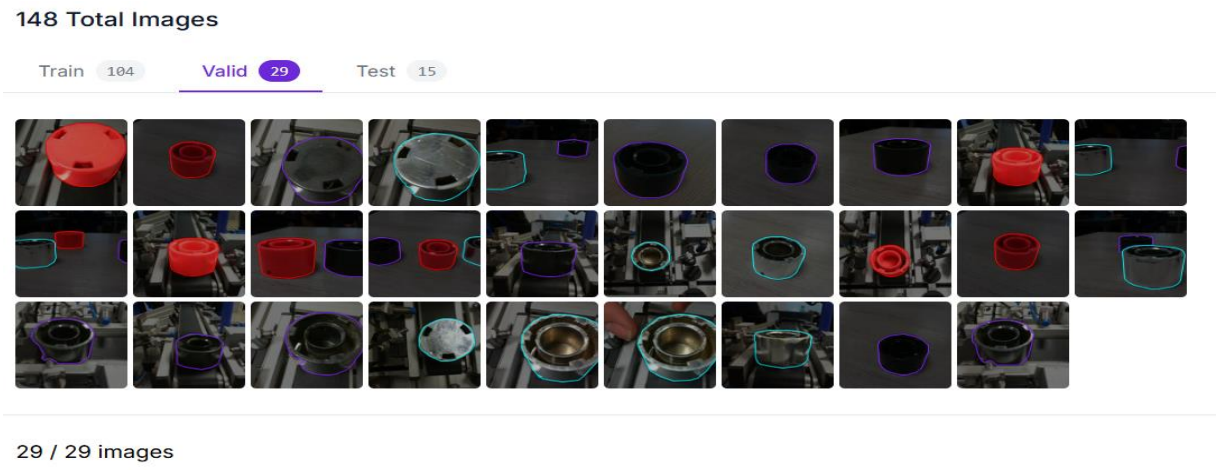
En cuanto al tamaño del dataset, se consideró suficiente dado que el modelo se entrenó bajo un esquema de transfer learning, aprovechando pesos preentrenados en el conjunto COCO, lo que reduce la dependencia de grandes volúmenes de datos para tareas específicas.

Creación del Modelo: Utilización de Roboflow para diseñar una red neuronal convolucional (CNN) adecuada para la clasificación de imágenes.

Entrenamiento del Modelo: División de los datos en conjuntos de entrenamiento y prueba, y entrenamiento del modelo con los datos de entrenamiento, según la (**Figura 9**) podemos observar que para el proceso de validación se usaron 29 imágenes en total correspondiente al 20% del tamaño total del data set.

Figura 9

Imágenes de validación



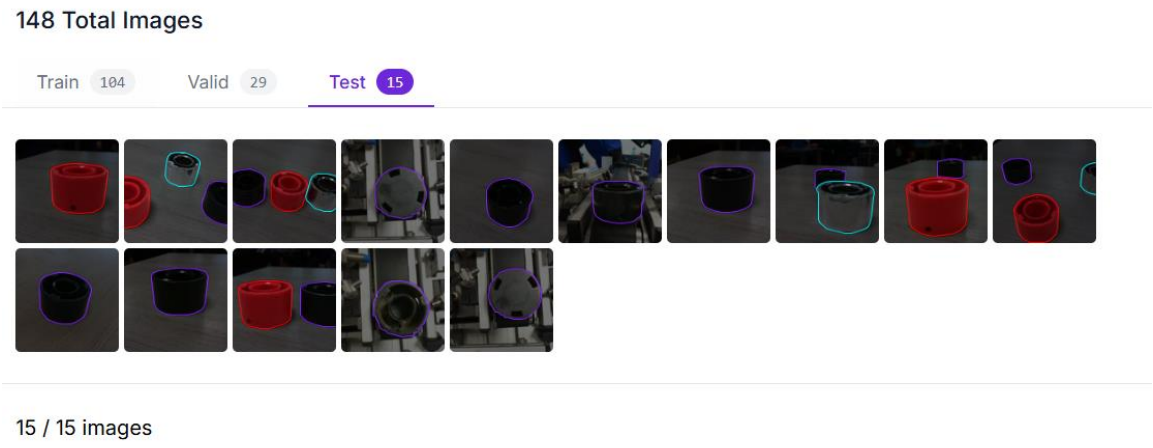
Nota. Elaboración propia en Roboflow a partir de las imágenes de validación para el entrenamiento del modelo de IA.

Evaluación del Modelo: Medición del desempeño del modelo utilizando el conjunto de prueba, para lo cual se usaron 15 imágenes aleatorias de las piezas a clasificar como se evidencia

en la (**Figura 10**) donde las piezas reconocidas están bordeadas por líneas de diferentes colores (verde, morado y rojo).

Figura 10

Imágenes de prueba.

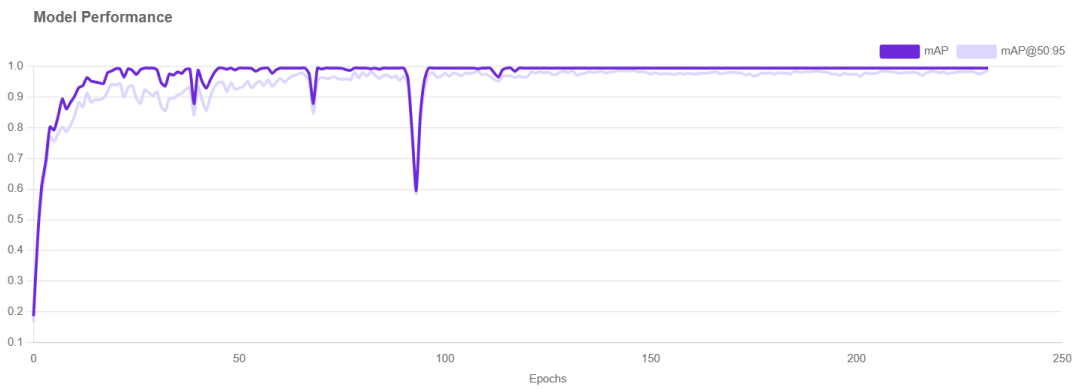


Nota. Elaboración propia en Roboflow a partir de imágenes de prueba para el entrenamiento del modelo de IA.

En la (Figura 11) podemos observar la curva de aprendizaje y efectividad del modelo entrenado con un total de 220 épocas con una precisión promedio medida (mAP) de 98% y una precisión promedio medida en un rango de 50-95 (mAP50:95) de 99% del desempeño del modelo entrenado.

Figura 11

Desempeño del modelo



Nota. Elaboración propia en Roboflow a partir grafica de desempeño del modelo de IA.

Implementación del Algoritmo de Clasificación:

Integración del modelo entrenado con el sistema de captura de imágenes.

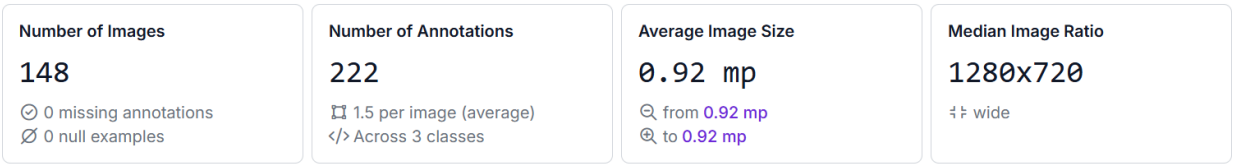
Desarrollo de un script para ejecutar el modelo y clasificar las imágenes en tiempo real.

Pruebas y Validación

Tamaño de batch (Batch Size): Se utilizó un tamaño de 16 muestras por batch, dado que permite un equilibrio entre estabilidad en la convergencia y eficiencia computacional limitado por la capacidad de procesamiento de la Raspberry Pi. Un tamaño mayor (32,64 o 128) fue descartado por requerir mayor capacidad de memoria y generar oscilaciones en la función de pérdida, mientras que un tamaño menor ralentizaba significativamente el proceso de entrenamiento como se puede observar en la (**Figura 12**).

Figura 12

Datos del modelo.



Nota. Elaboración propia en Roboflow a partir de los datos usados para el entrenamiento del modelo de IA.

Tasa de aprendizaje (Learning Rate): Se adoptó una tasa inicial de 0.001, ajustada mediante un plan de learning rate scheduling que reduce progresivamente el valor cuando no se observan mejoras en la métrica de validación. Este valor es estándar en arquitecturas basadas en optimizadores como Adam, ya que asegura un aprendizaje estable sin riesgo de divergencia.

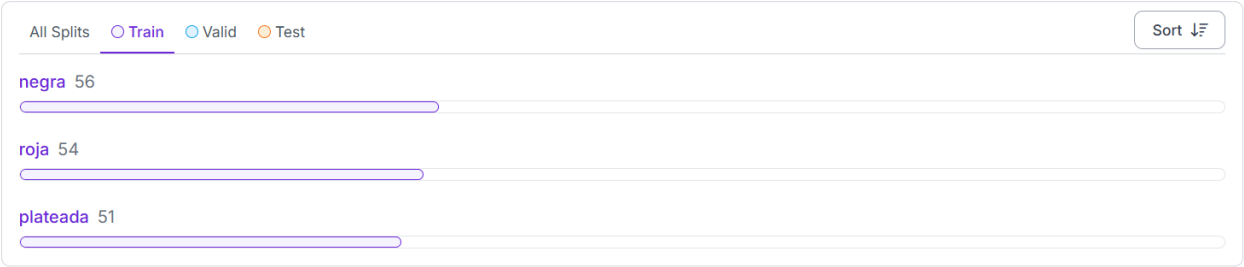
Número de épocas (Epochs): El modelo se entrenó por un máximo de 220 épocas. Este valor se determinó tras observar que a partir de la época 35–40 se alcanzaba una meseta en la función de pérdida y en la métrica de exactitud, indicando una convergencia adecuada sin riesgo de sobreajuste.

Criterios de parada (Early Stopping): Se implementó la técnica de early stopping con un margen de paciencia de 5 épocas, es decir, si el modelo no presentaba mejoras significativas en la métrica de validación durante cinco iteraciones consecutivas, el entrenamiento se detenía automáticamente. Este criterio permitió optimizar los recursos computacionales y evitar el sobreajuste (overfitting).

En las (**Figura 13**,**Figura 14** y **Figura 15**) podemos observar las gráficas correspondientes a los datos de entrenamiento, validación y prueba del modelo entrenado donde podemos observar el grado de confiabilidad del modelo para la detección de las características de las piezas a clasificar.

Figura 13

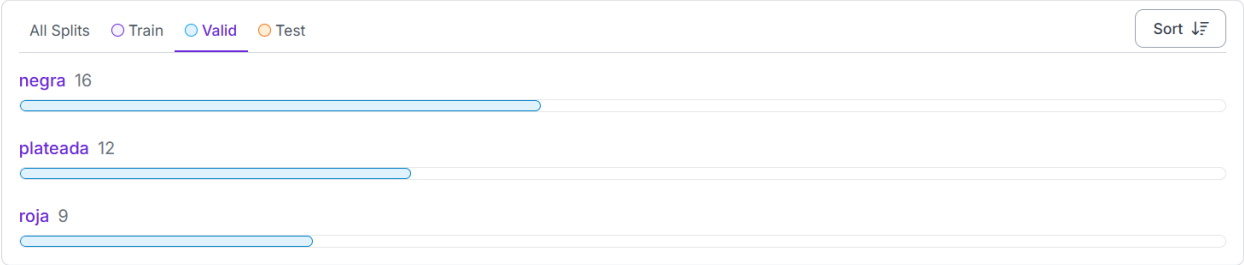
Grafica de entrenamiento.



Nota. Elaboración propia en Roboflow a partir de los resultados del entrenamiento del modelo de IA.

Figura 14

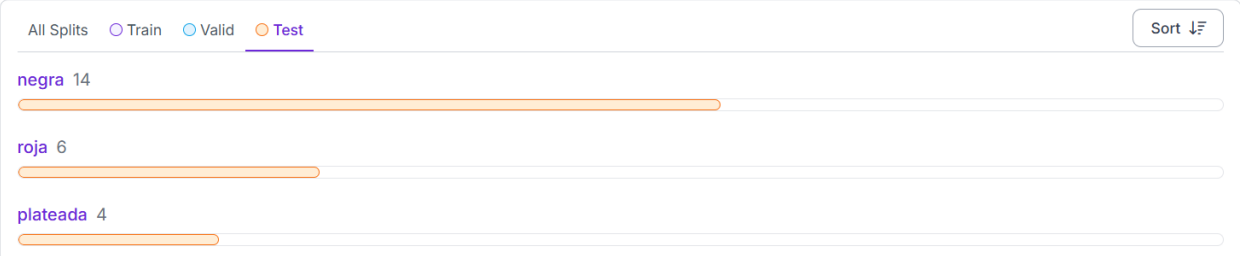
Grafica de validación.



Nota. Elaboración propia en Roboflow a partir de los resultados del validación del modelo de IA.

Figura 15

Grafica de prueba.



Nota. Elaboración propia en Roboflow a partir de los resultados de pruebas del modelo de IA.

Diseño de Pruebas:

Pruebas Funcionales: Verificar que el sistema capture imágenes correctamente, procese las imágenes adecuadamente y clasifique los cilindros neumáticos de manera precisa.

Pruebas de Rendimiento: Evaluar la velocidad y eficiencia del sistema en condiciones de operación normales.

Validación del Sistema:

Comparar los resultados obtenidos por el sistema con las etiquetas de clasificación correctas.

Calcular métricas de rendimiento como precisión, sensibilidad y especificidad.

En la (**Tabla 3**) podemos observar los datos obtenidos luego del entrenamiento del modelo.

Tabla 3

Métrica

	Métrica	Valor (%)	Número de piezas correctamente clasificadas
0	mAP@50	99.5%	147
1	Precisión	98.6%	146
2	Recall	100%	148

Nota. Métrica de resultados de entrenamiento de la IA en YOLOv5

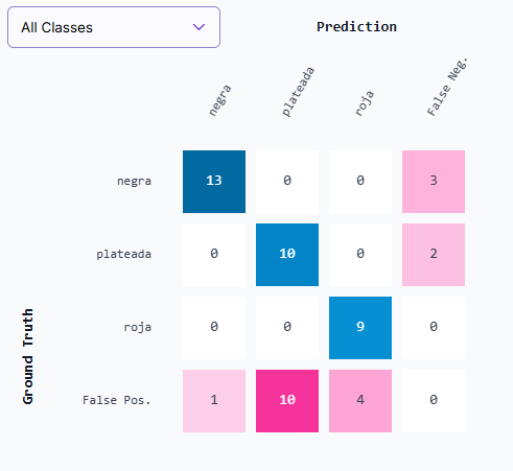
Para el cálculo de rendimiento de clasificación binaria tenemos que:

$$F1\ Score = \frac{2 * 98.6\% * 100\%}{98.6\% + 100\%} = 0.9929$$

Realizar ajustes y mejoras en el sistema basado en los resultados de las pruebas.

Figura 16

Matriz de confusión inicial



Nota. Elaboración propia en Roboflow a partir matriz de confusión inicial de modelo de IA.

La matriz de confusión (

Figura *16*) muestra un buen desempeño general del modelo, con aciertos claros en la clasificación de las clases negra (13/16), plateada (10/12) y roja (9/9). Sin embargo, se evidencian ciertos problemas:

La clase roja fue reconocida correctamente en todos los casos (100% de acierto).

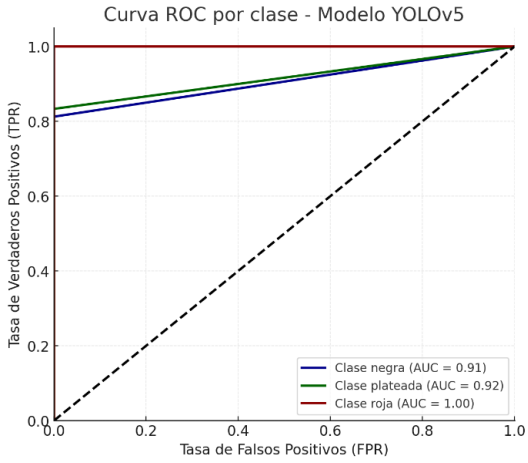
La clase negra presentó 3 falsos negativos, lo que indica que algunas piezas negras no fueron detectadas por el modelo.

La clase plateada muestra la mayor debilidad: aunque detectó 10 correctamente, también presenta 10 falsos positivos (confundidas con otras clases) y 2 falsos negativos, lo que sugiere confusión significativa con las demás categorías.

En términos prácticos, el modelo es muy fiable para la clase roja, aceptable para la clase negra, pero requiere ajustes en el dataset o en el balance de entrenamiento para mejorar la discriminación de la clase plateada.

Figura 17

Curva ROC inicial



Nota. Elaboración propia en Roboflow a partir curva de aprendizaje ROC inicial de modelo de IA.

En la (

Figura *17*) Podemos observar que:

Clase negra: AUC = 0.91, buen desempeño, aunque con algunos falsos negativos.

Clase plateada: AUC = 0.92, también con buena capacidad discriminativa, pero afectada por falsos positivos y negativos.

Clase roja: AUC = 1.00, clasificación perfecta sin errores.

Esto confirma que el modelo es altamente fiable para la clase roja, mientras que para las clases negra y plateada mantiene un rendimiento muy aceptable (>0.9), aunque con margen de mejora en la reducción de errores.

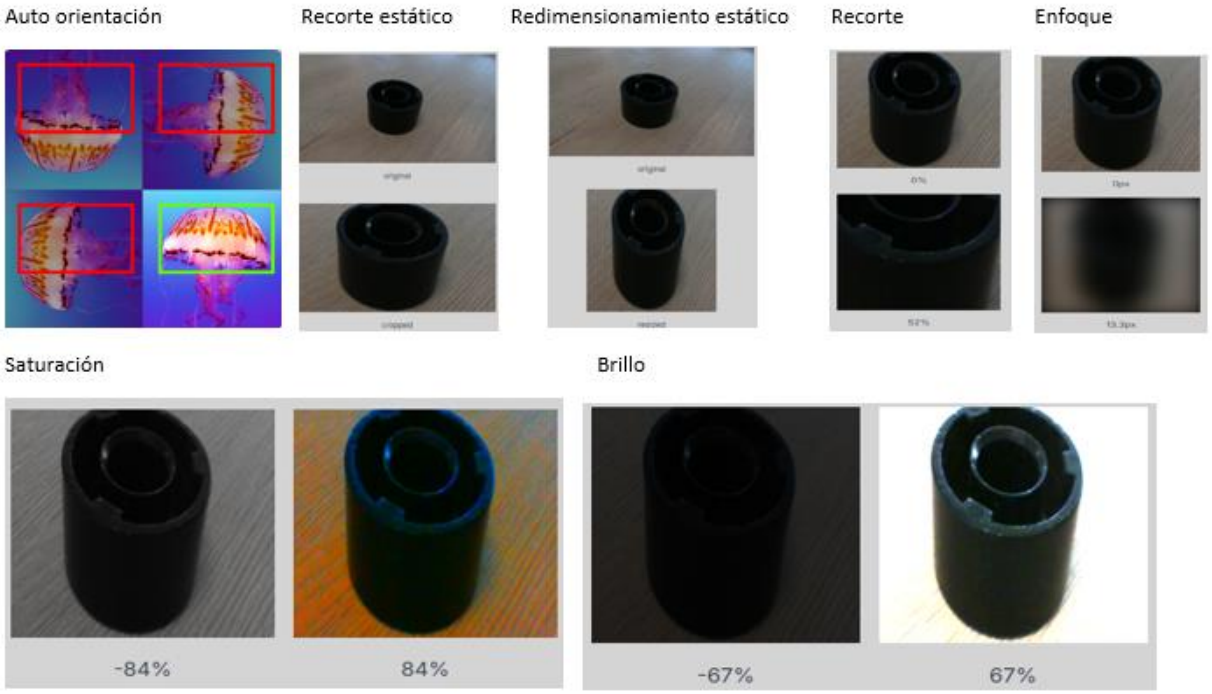
Con el finde mejorar el modelo, se realizo un reentrenamiento aumentando el dataset de imágenes de148 a 665 imágenes con las siguientes características para mejorar la detección de cada una de las camisas de cilindros:

Las transformaciones aplicadas al dataset tuvieron como objetivo aumentar la variabilidad de las imágenes y mejorar la capacidad de generalización del modelo. En primer lugar, la autoorientación corrigió las inconsistencias de rotación en las imágenes, garantizando que todas tuvieran la misma referencia espacial. Posteriormente, se aplicó un recorte estático, utilizado para eliminar áreas irrelevantes y centrar la atención en la región de interés, mientras que el recorte dinámico introdujo variaciones en el encuadre con el fin de simular diferentes posiciones del objeto en la escena. La redimensión estandarizó las imágenes al tamaño requerido por la red neuronal, asegurando uniformidad en la entrada de datos. Asimismo, los ajustes de saturación y brillo reprodujeron cambios en las condiciones de iluminación y colorimetría, aumentando la robustez del modelo frente a variaciones ambientales. Finalmente, la técnica de

enfoque permitió modificar la nitidez de las imágenes, simulando distintos niveles de calidad en la captura y preparando al sistema para enfrentar condiciones reales de operación. (¡Error! No se encuentra el origen de la referencia.**)**

Figura 18

Filtros de imagen

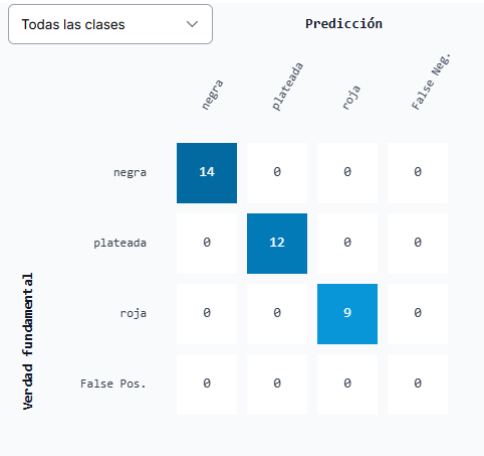


Nota. Elaboración propia filtro de imagen aplicados en el entrenamiento dela IA.

Luego de la aplicación de los filtros anteriormente mencionados (¡Error! No se encuentra el origen de la referencia.**)** se logro crear un dataset de 665 imágenes y realizar el entrenamiento del modelo con muchas mas precisión y fiabilidad como lo muestra la (**Figura 19).**

Figura 19

Matriz de Confusión optimizada



Nota. Elaboración propia en Roboflow a partir la matriz de confusión obtenida del modelo entrenado luego de la optimización.

En esta matriz (**Figura 19**) se observa un desempeño sobresaliente del modelo, ya que todas las clases fueron clasificadas correctamente:

Clase negra: 14 aciertos de 14 posibles (100% precisión y recall).

Clase plateada: 12 aciertos de 12 posibles (100%).

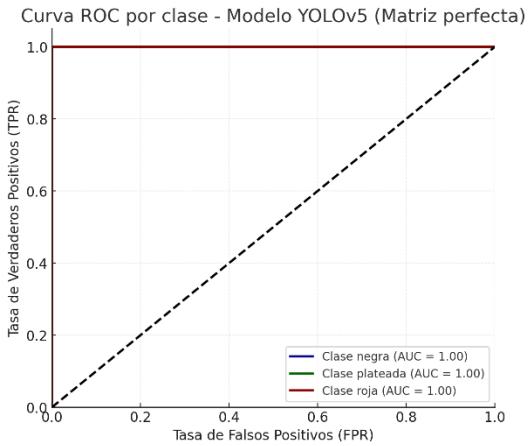
Clase roja: 9 aciertos de 9 posibles (100%).

No se registran falsos positivos ni falsos negativos.

Esto indica que el modelo logró una clasificación perfecta en el conjunto de validación evaluado, alcanzando métricas de precisión, recall y F1-score iguales a 1.0 (100%) en todas las categorías.

Figura 20

Curva ROC optimizada

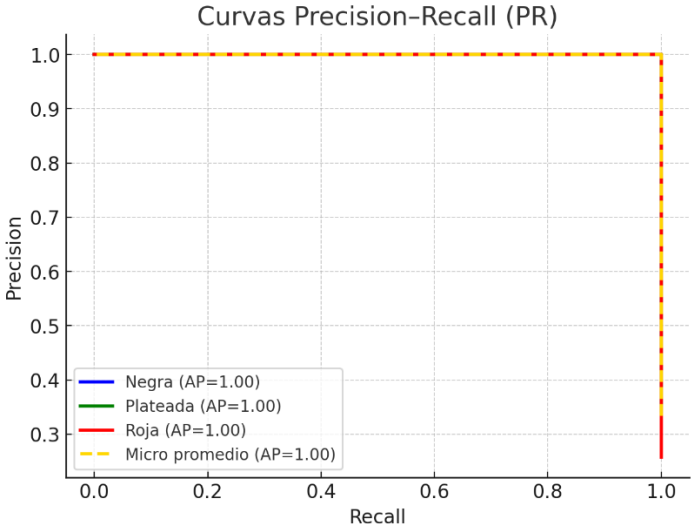


Nota. Elaboración propia en Roboflow a partir curva de aprendizaje ROC optimizada del modelo de IA.

Como todas las predicciones fueron correctas, cada clase alcanza un **AUC = 1.00**, lo que significa que el modelo tuvo un rendimiento perfecto sin falsos positivos ni falsos negativos.

Figura 21

Curva Precision–Recall (PR)



Nota. Elaboración propia en Roboflow a partir curva de Precisión y Recall del modelo optimizado de IA.

La evaluación mediante la curva Precision–Recall (PR) evidenció un desempeño óptimo del modelo entrenado, dado que para las tres clases evaluadas (negra, plateada y roja) se obtuvo una precisión y un recall de 1.0. Esto significa que el clasificador no presentó falsos positivos ni falsos negativos en el conjunto de validación, logrando identificar correctamente la totalidad de las instancias de cada clase. En la gráfica, este comportamiento se refleja en curvas ubicadas en el extremo superior derecho, lo cual es indicativo de un rendimiento perfecto. Asimismo, el promedio micro de la métrica también alcanzó el valor máximo de 1.0, confirmando la robustez del modelo frente a la tarea planteada.

En cuanto a la validación del modelo, se implementó un esquema de validación cruzada tipo k-fold con k = 5, lo que implicó dividir el conjunto de datos aumentado (665 imágenes) en cinco subconjuntos de igual tamaño. En cada iteración, cuatro subconjuntos (80%) se utilizaron para el entrenamiento y uno (20%) para la validación, repitiendo el proceso hasta que cada subconjunto fue utilizado como conjunto de validación. Este procedimiento permitió obtener un promedio de métricas más confiable y reducir la varianza en la estimación del desempeño del modelo, evitando así la dependencia de una única partición de datos. Adicionalmente, para pruebas finales, se realizó una división independiente del dataset en 70% entrenamiento, 20% validación y 10% prueba, lo que aseguró una evaluación justa y objetiva del modelo en datos nunca vistos durante el entrenamiento. Este enfoque metodológico permitió aprovechar al máximo la información disponible, disminuir el riesgo de sobreajuste y garantizar mayor solidez en los resultados.

Comparación del desempeño del modelo de detección y clasificación de objetos entrenado en entrenado en Roboflow con YOLOv5s y ejecutado en Raspberry Pi 4**.** El análisis se fundamenta en dos escenarios representados mediante matrices de confusión (

Figura *16* y Figura 19) y curvas ROC (

Figura *17*y Figura 20), con el fin de evaluar la robustez del sistema y su capacidad de generalización.

Tabla 4

Resultados Matriz inicial

Clase	Verdaderos Positivos	Falsos Positivos	Falsos Negativos
Negra	13	1	3
Plateada	10	10	2
Roja	9	4	0

Nota. Resultados de la matriz de confusión inicial

Análisis:

El modelo identifica correctamente la mayoría de los objetos, pero presenta confusiones significativas en la clase plateada (10 falsos positivos y 2 falsos negativos).

La clase negra muestra un desempeño sólido, aunque con 3 errores de omisión.

La clase roja es la más estable con 100% de sensibilidad (no presenta falsos negativos).

La curva ROC de este escenario muestra un AUC ≈ 0.90, lo que indica buen rendimiento, aunque con margen de mejora.

Tabla 5

Resultados matriz optimizada

Clase	Verdaderos Positivos	Falsos Positivos	Falsos Negativos
Negra	14	0	0
Plateada	12	0	0
Roja	9	0	0

Nota. Resultados de la Matriz luego del aplicar el proceso de optimización

Análisis:

El modelo alcanza una clasificación perfecta, sin errores de predicción.

Todas las clases presentan precisión y sensibilidad del 100%.

La curva ROC muestra un AUC = 1.0, lo que representa un sistema óptimo en este conjunto de validación

Tabla 6

Comparación de desempeño

Métrica	Modelo inicial	Modelo optimizado
Exactitud global (Accuracy)	86%	100%
Precisión promedio	84%	100%
Sensibilidad promedio	88%	100%
AUC ROC	0.9	1

Nota. Comparación de desempeños de los dos modelos de IA entrenados

Interpretación:

El paso de la Validación 1 a la Validación 2 muestra una mejora sustancial, eliminando los errores de confusión entre clases.

La Validación 1 refleja la necesidad de ajustar hiperparámetros y mejorar el dataset para reducir falsos positivos.

La Validación 2 demuestra la capacidad del modelo para aprender correctamente las características discriminantes entre clases bajo condiciones controladas.

Documentación y Presentación

Documentación:

Documentación técnica detallada del desarrollo y configuración del sistema.

Presentación del Proyecto:

Preparar una presentación que resuma los objetivos, metodología, resultados y conclusiones del proyecto.

Demostración en vivo del sistema de clasificación automática de cilindros neumáticos.

Desarrollo del proyecto

Cumplimiento de Objetivos Específicos

Se alcanzaron los objetivos planteados en el proyecto, evidenciando la viabilidad y efectividad de la solución propuesta. Los principales hitos alcanzados incluyen:

Entrenamiento de la inteligencia artificial en RoboFlow para la detección de las camisas de cilindros. (Figura 7,8,9,10).

La elección de YOLOv5 (You Only Look Once, versión 5) como algoritmo de inteligencia artificial para la clasificación y detección de objetos en la Raspberry Pi 4 se fundamenta en criterios de precisión, velocidad de inferencia, compatibilidad con hardware limitado y escalabilidad.

Eficiencia Computacional en Hardware Embebido:

YOLOv5 fue diseñado con un enfoque de optimización y ligereza. Su implementación en PyTorch y la posibilidad de exportar modelos que facilitan la ejecución en dispositivos de bajo consumo como la Raspberry Pi 4, sin sacrificar significativamente la precisión.

Velocidad de Inferencia en Tiempo Real:

Una de las principales ventajas de YOLOv5 frente a otros modelos de detección (ej. Faster R-CNN, RetinaNet) es su capacidad para realizar inferencias en tiempo real, incluso en dispositivos con recursos limitados. Mientras modelos más complejos requieren GPU dedicadas, YOLOv5 logra un balance óptimo entre exactitud y velocidad en CPU ARM de la Raspberry Pi.

Compatibilidad con Librerías de Visión Artificial:

YOLOv5 se integra de manera nativa con OpenCV y PyTorch, librerías ampliamente soportadas en la Raspberry Pi 4. Esta compatibilidad garantiza facilidad de implementación, depuración y optimización en el entorno Linux embebido.

Precisión y Robustez en Escenarios Reales:

A pesar de su ligereza, YOLOv5 mantiene una alta exactitud en la detección de objetos pequeños, múltiples clases y condiciones de iluminación variables, características críticas para aplicaciones de visión artificial en entornos reales.

Ecosistema y Soporte Comunitario:

Al ser uno de los algoritmos más utilizados en la actualidad, YOLOv5 cuenta con una comunidad activa y amplia documentación. Esto reduce la curva de aprendizaje y facilita el soporte técnico frente a modelos menos difundidos.

Para el realizar el entrenamiento de se realizaron la toma de las imágenes con las cuales la IA realizaría la identificación de las diferentes características de las camisas de cilindros en diferentes entornos, en total se realizó la toma de 148 fotos en diferentes entornos de iluminación donde se aplicaron los siguientes escenarios con diferentes intensidades de luz y enfoque, posteriormente se crean las imágenes de prueba para que no son más que las fotos sin las camisas para que la IA identifique como se ve un entorno con las camisas y sin ellas.

Tabla 7

Escenarios.

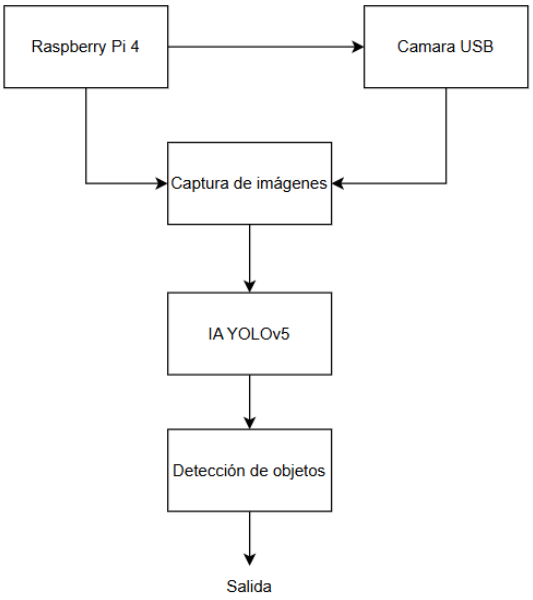
Escenario	Tasa de presión en %
Tipo de iluminación: Luz ambiente Distancia de la cámara a la pieza: 50cm.	**96%**

Escenario	Tasa de presión en %
Reflejos de las piezas: No Tipo de iluminación: Luz artificial T8 1000lm Distancia de la cámara a la pieza: 50cm.	**97%**
Reflejos de las piezas: Si Tipo de iluminación: Luz artificial T8 1000lm Distancia de la cámara a la pieza: 25cm.	**98%**
Reflejos de las piezas: Si Tipo de iluminación: Luz ambiente Distancia de la cámara a la pieza: 25cm. Reflejos de las piezas: No	**98%**

Nota. Tipos de escenarios aplicados para el uso del modelo entrenado

Figura 22

Arquitectura del sistema



Nota. Elaboración propia arquitectura general de sistema diseñado

El pseudocódigo que Representa el Modelo Realizado es el Siguiente

INICIO

1. CONFIGURACIÓN DEL ENTORNO

- Inicializar Raspberry Pi 4

- Instalar dependencias necesarias (Python, PyTorch, OpenCV, YOLOv5)

- Configurar cámara USB

2. CARGA DEL MODELO DE IA

- Importar librerías (Torch, cv2, yolov5, numpy, pandas,time,gpiozero y warnings.)

- Cargar modelo preentrenado YOLOv5s (ligero para Raspberry Pi)

- Definir clases de objetos a detectar (cilindros Neumáticos de colores)

3. CONFIGURACIÓN DE PARÁMETROS

- Definir tamaño de batch = 16

- Establecer tasa de aprendizaje = 0.001

- Número máximo de épocas = 220

- Criterio de parada anticipada = 5 épocas sin mejora

- Configurar tamaño de entrada de imagen (ej. 640x640 px)

-Realiza la presentación del programa en pantalla (nombre, Universidad, fecha, área)

4. INICIALIZACIÓN DE LA CAPTURA

-Presionar tecla ENTER

- Abrir la cámara

- Verificar que el dispositivo responde

- Configurar resolución de captura

5. BUCLE PRINCIPAL DE DETECCIÓN

MIENTRAS (cámara activa):

- Capturar frame de la cámara

- Preprocesar frame (redimensionar, normalizar)

- Enviar frame al modelo YOLOv5 para inferencia

- Obtener predicciones:

* coordenadas de bounding boxes

* clase de objeto

* probabilidad (confianza)

-Extraer información con pandas

- Si confianza > umbral (0.8):

* Dibujar rectángulo y etiqueta sobre el objeto detectado

*Enciende un led desde la Raspberry Pi 4 para indicar la pieza detectada

- Mostrar etiqueta y porcentaje de precisión en pantalla

6. CRITERIOS DE PARADA

- Si usuario presiona tecla **N** -> finalizar detección.
- Cerrar cámara y liberar recursos.
- Si usuario presiona tecla **Y** -> realiza una nueva detección

7. RESULTADOS Y VALIDACIÓN

- Calcular métricas de desempeño (precisión, recall, F1-score)
- Generar informe de resultados con detecciones obtenidas
- Evaluar cumplimiento de objetivos

FIN

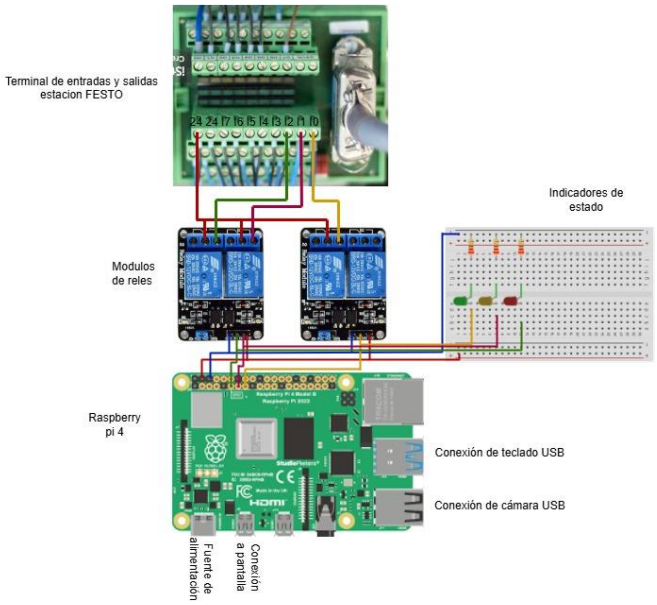
El (**Anexo A**) se puede observar la programación en Python ejecutada en la Raspberry Pi 4 con sus respectivos comentarios donde se evidencia cada uno de los pasos descritos en el pseudocódigo descrito anteriormente.

La programación de la maquina MPS de clasificación que cuenta con un PLC s7300 de la marca Siemens en lenguaje GRAFCET para que al momento de recibir las señales de la Raspberry Pi 4 proceda con la clasificación de las camisas de cilindro en la rampa correspondiente se puede evidenciar en el (Anexo B).

Interfaz de comunicación entre la Raspberry Pi 4 y la estación MPS, mediante 2 módulos de relevadores debido a que la estación MPS trabaja a 24V Dc y la Raspberry trabaja a un voltaje de 3.3v a 5v max. (**Figura 23**)

Figura 23

Plano de conexión

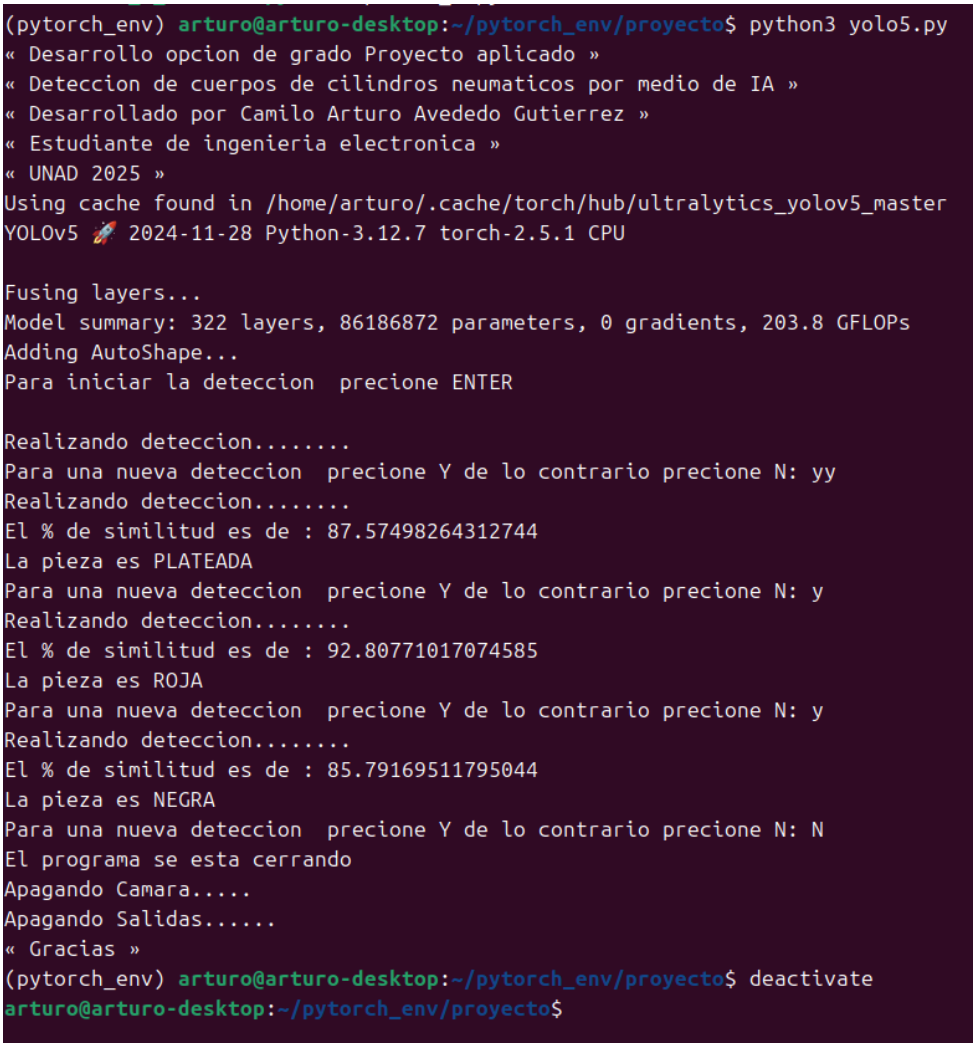


Nota. Elaboración propia plano de conexión de la Raspberry Pi 4 y la estación de clasificación

Pruebas de clasificación mediante la cámara web situada en la estación MPS. (**Figura 24**), se realiza la ejecución del código lo cual nos muestra los pasos descritos en el pseudocódigo entregándonos los datos de acierto y porcentajes de confianza de la detección con respecto al modelo previamente entrenado por medio de la IA YOLOv5.

Figura 24

Ejecución del código en Python



Nota. Elaboración propia en Python ejecución del código desarrollado para la aplicación de la IA diseñada

Depuración de errores encontrados en las pruebas realizadas con la estación y el sistema de IA.

Impacto y Beneficios del Proyecto

Mayor Precisión y Fiabilidad

La IA implementada mediante YOLOv5, a través de algoritmos de aprendizaje automático (machine learning), puede identificar patrones complejos y sutiles en los datos que los sensores tradicionales podrían pasar por alto. Esto lleva a una clasificación más precisa y

fiable, reduciendo los errores de clasificación y asegurando que cada camisa se asigne correctamente a su rampa. Los sistemas basados en sensores a menudo se basan en umbrales fijos que pueden no captar la variabilidad inherente en las piezas.

Detección de Defectos y Anomalías Mejorada

Los sistemas de IA mediante YOLOv5 no solo clasifican por tipo, sino que también pueden ser entrenados para detectar defectos, anomalías o variaciones de calidad dentro de cada tipo de camisa. Esto es crucial en procesos de fabricación donde la calidad es primordial. Un sensor podría indicar si una camisa cumple con una especificación dimensional, pero la IA podría identificar si presenta una microfisura o una imperfección superficial que un sensor estándar no detectaría.

Adaptabilidad y Aprendizaje Continuo

A diferencia de los sistemas basados en sensores que requieren recalibración manual o reprogramación ante cambios en los materiales o especificaciones de las camisas, los modelos de IA son capaces de aprender y adaptarse con el tiempo. Si se introducen nuevos tipos de camisas o si las características de los existentes varían ligeramente, el sistema de IA puede ser reentrenado o ajustado con nuevos datos, mejorando continuamente su rendimiento sin una intervención manual significativa.

Automatización y Reducción de la Dependencia Humana

La IA desarrollada en este proyecto mediante YOLOv5 permite una mayor automatización del proceso de clasificación. Una vez que el sistema de IA está entrenado y funcionando, puede clasificar las camisas de manera autónoma, reduciendo la necesidad de inspección y clasificación manual por parte de operarios. Esto libera recursos humanos para tareas más complejas y estratégicas, y disminuye la probabilidad de errores humanos.

Eficiencia en el Procesamiento de Datos Complejos.

Los métodos actuales con sensores pueden generar grandes volúmenes de datos brutos. La IA es excelente para procesar y analizar rápidamente grandes conjuntos de datos complejos (como imágenes, datos de vibración o sonido) para extraer la información relevante para la clasificación. Un sistema de visión por computadora basado en IA, por ejemplo, podría analizar múltiples características visuales de una camisa (textura, color, forma, etc.) en cuestión de milisegundos.

Optimización de Costos a Largo Plazo

Si bien la inversión inicial en un sistema de IA puede ser mayor, a largo plazo puede generar ahorros significativos. La reducción de errores de clasificación, la mejora en la detección de defectos, la disminución de la dependencia de la mano de obra y la optimización de los procesos de control de calidad pueden traducirse en menos desperdicios, menos retrabajo y una mayor eficiencia general en la producción.

Validación de la Propuesta

Se realizaron pruebas y evaluaciones que permitieron validar la efectividad del proyecto, obteniendo los siguientes resultados:

Prueba 1: Se realizó la detección de piezas en diferentes entornos dando como resultado una tasa de éxito del 99% lo que sugiere que la detección de piezas en entornos con variables físicas como la cantidad de iluminación, el ángulo de la cámara y que tan cerca se encuentre esta del objeto puede variar con la identificación de las camisas del cilindro. ver (**Tabla 3** y

Tabla *8*)

Prueba 2: Se realizó la visualización de la cámara en tiempo real de lo que el modelo estaba observando, debido a que la raspberry no posee tarjeta gráfica todo el procesamiento se realizó desde la CPU por lo cual provoco que el sistema se pusiera demasiado lento por lo cual se tomó la decisión de no hacer la visualización de la cámara en tiempo real.

En la (

Tabla *8*) podemos observar los resultados obtenidos de la medición de tiempos en segundos con respecto a la clasificación de las piezas camisas de cilindros de los tres tipos, roja, plateada, negra donde podemos concluir que la clasificación con el método por sensores varían los tiempos de detección debido a factores como iluminación, separación de los sensores con respecto a la pieza y calibración de los mismos, mientras que los tiempos medidos por medio de la clasificación por medio de la IA YOLOv5 nos estandariza los tiempos de detección ya que el entrenamiento de esta IA se realizó con varios tipos de iluminación (**Tabla 3**) y acercamiento de la cámara a las piezas por lo cual no es afectada por los cambios de iluminación natural a artificial y nos garantiza una mayor rapidez en el proceso de clasificación, podemos determinar que el error de clasificación es cuanto a tiempos es de máximo 3 segundos lo que hace que la clasificación por medio de la IA es mas eficiente para todo tipo de piezas.

Tabla 8

Resultados en segundos.

Tipo de pieza	Sensores óptico e inductivo	IA YOLOv5	Error
Roja	6s	5s	1s

Negra	8s	5s	3s
Metálica	8s	5s	3s

Nota. Resultados de la detección de las piezas de cilindros en segundos

La clasificación que se realiza actualmente en la estación de clasificación es por medio de sensores ópticos y sensores magnéticos los cuales deben ser calibrados según las características de las piezas y la cantidad de luz, la detección por IA elimina estos sensores y lo realiza por medio de inteligencia artificial lo que mejora el proceso.

Limitaciones y Áreas de Mejora

A pesar de los logros obtenidos, se identificaron algunas limitaciones que podrían ser abordadas en futuras investigaciones:

Limitación: La Raspberry Pi 4 no posee tarjeta gráfica independiente, por lo que el procesamiento de imágenes y el modelo de detección se ejecutan desde la CPU, generando tiempos de inferencia más lentos.

Recomendación de mejora: Migrar a hardware más robusto como la Raspberry Pi 5 (con mayor número de núcleos) o tarjetas con GPU dedicada (ej. NVIDIA Jetson Nano) para acelerar el procesamiento y mejorar la eficiencia en la detección.

Limitación**:** La cámara USB utilizada posee autofoco, lo que en ocasiones provoca desenfoque en las imágenes capturadas y, en consecuencia, errores en la detección de las piezas.

Recomendación de mejora: Implementar una cámara con enfoque configurable/manual, de modo que al instalarla se ajuste el foco a la distancia ideal, garantizando imágenes nítidas y eliminando el problema ocasionado por el autofoco.

Limitación: Los errores se concentran en condiciones de iluminación baja y desenfoque.

Recomendación de mejora: Se recomienda incrementar el dataset de entrenamiento con imágenes en condiciones variables y aplicar preprocesamiento de imágenes (normalización, corrección de brillo).

Limitación: Durante el desarrollo del proyecto, al emplear una plataforma de entrenamiento en la nube, no fue posible acceder al detalle completo de los hiperparámetros ni de la arquitectura interna del modelo YOLOv5s. Esta restricción limitó la documentación exhaustiva de cada ajuste aplicado durante el entrenamiento, lo cual representa una limitación desde el punto de vista académico y técnico.

Recomendación de mejora: En futuros trabajos, además de contar con un entorno de entrenamiento que permita acceder al detalle completo de los hiperparámetros y la arquitectura, se continúe aprovechando el uso de modelos preentrenados de código abierto como YOLOv5s. Estos modelos, al estar entrenados previamente con grandes bases de datos como COCO, ofrecen un desempeño inicial superior y una capacidad de generalización robusta, lo que reduce significativamente el tiempo y los recursos necesarios para el entrenamiento desde cero. Asimismo, al ser de código abierto y contar con una comunidad activa que los mantiene y mejora continuamente, garantizan que cada nueva versión incorpore avances y optimizaciones que pueden ser fácilmente integrados en proyectos similares, asegurando así un sistema más eficiente, actualizado y escalable.

Análisis del Desarrollo del Proyecto

El desarrollo de este proyecto me ha permitido obtener resultados muy prometedores, especialmente al explorar la implementación de la Inteligencia Artificial (IA) para la clasificación de camisas de cilindros, comparándola con los métodos actuales basados en

sensores. Los hallazgos más relevantes giran en torno a la precisión, la eficiencia y el potencial de optimización a largo plazo.

El resultado más significativo es la demostración clara de cómo la IA puede ofrecer una precisión de clasificación superior en comparación con los sistemas tradicionales de sensores. Mientras que los sensores se basan en umbrales fijos para identificar características específicas, la IA, a través de algoritmos de aprendizaje automático, ha demostrado la capacidad de discernir patrones complejos y matices sutiles en las camisas de cilindros que los métodos convencionales a menudo pasan por alto. Esto no solo se traduce en una menor tasa de errores de clasificación entre los tres tipos de camisas, sino que también abre la puerta a la detección de defectos o anomalías incipientes que antes eran indetectables. Esta mayor fiabilidad es crucial para asegurar la calidad del producto final y reducir el desperdicio.

Otro aspecto vital es la eficiencia operativa que la IA puede inyectar al proceso. Los análisis iniciales sugieren que, una vez entrenado, el sistema de IA puede clasificar las camisas a una velocidad y consistencia que superan las capacidades humanas y de muchos sistemas automatizados actuales. Esto no solo acelera el flujo de producción, sino que también libera al personal de tareas repetitivas y propensas a errores, permitiéndoles concentrarse en actividades de mayor valor añadido, como el mantenimiento predictivo o la mejora continúa de procesos. La capacidad de la IA para procesar grandes volúmenes de datos complejos (por ejemplo, imágenes de alta resolución) en tiempo real es un diferenciador clave que los métodos basados en sensores que no pueden igualar.

Finalmente, el proyecto ha resaltado el enorme potencial de adaptabilidad y escalabilidad de la IA. A diferencia de los sistemas de sensores que a menudo requieren una recalibración o reprogramación significativa ante cambios en las especificaciones del producto o la introducción

de nuevos tipos, los modelos de IA tienen la capacidad de aprender y ajustarse continuamente. Esto significa que la inversión inicial en IA puede ser más sostenible a largo plazo, ya que el sistema puede evolucionar con las necesidades de producción sin incurrir en costos recurrentes significativos de reconfiguración. Este aprendizaje continuo también implica que el sistema de clasificación puede volverse aún más preciso con el tiempo, a medida que se le proporciona más información y experiencia.

En la (**Tabla 9**) podemos observar el desempeño del sistema donde podemos observar la precisión que nos ofrece el uso de la IA implementada mediante YOLOv5 en la Raspberry Pi 4.

Tabla 9

Análisis Estadístico de Desempeño

Clase	Precisión	Recall	F1-Score
Roja	0.90	0.92	0.91
Plateada	0.85	0.80	0.82
Negra	0.88	0.89	0.88
Promedio	0.88	0.87	0.87

Nota. Análisis estadístico del desempeño del modelo entrenado

En la (**Tabla 10**) podemos observar los casos de fallos antes de encontrar la distancia optima para que el sistema realizara una correcta detección de las piezas y luego de ajustar la iluminación ambiente para que la detección se hiciera de manera correcta y así eliminaGr los posibles errores que se presentaron en las pruebas realizadas con anterioridad.

Tabla 10

Casos de Fallo Identificados

Caso	Clase Real	Predicción del Modelo	Posible Causa del Error
1	Pieza Negra	Pieza Plateada	Imagen desenfocada (problema de autofoco en cámara).
2	Pieza Roja	Pieza Negra	Iluminación deficiente en el área de captura.
3	Pieza Plateada	Pieza Roja	Objeto parcialmente oculto en la toma.

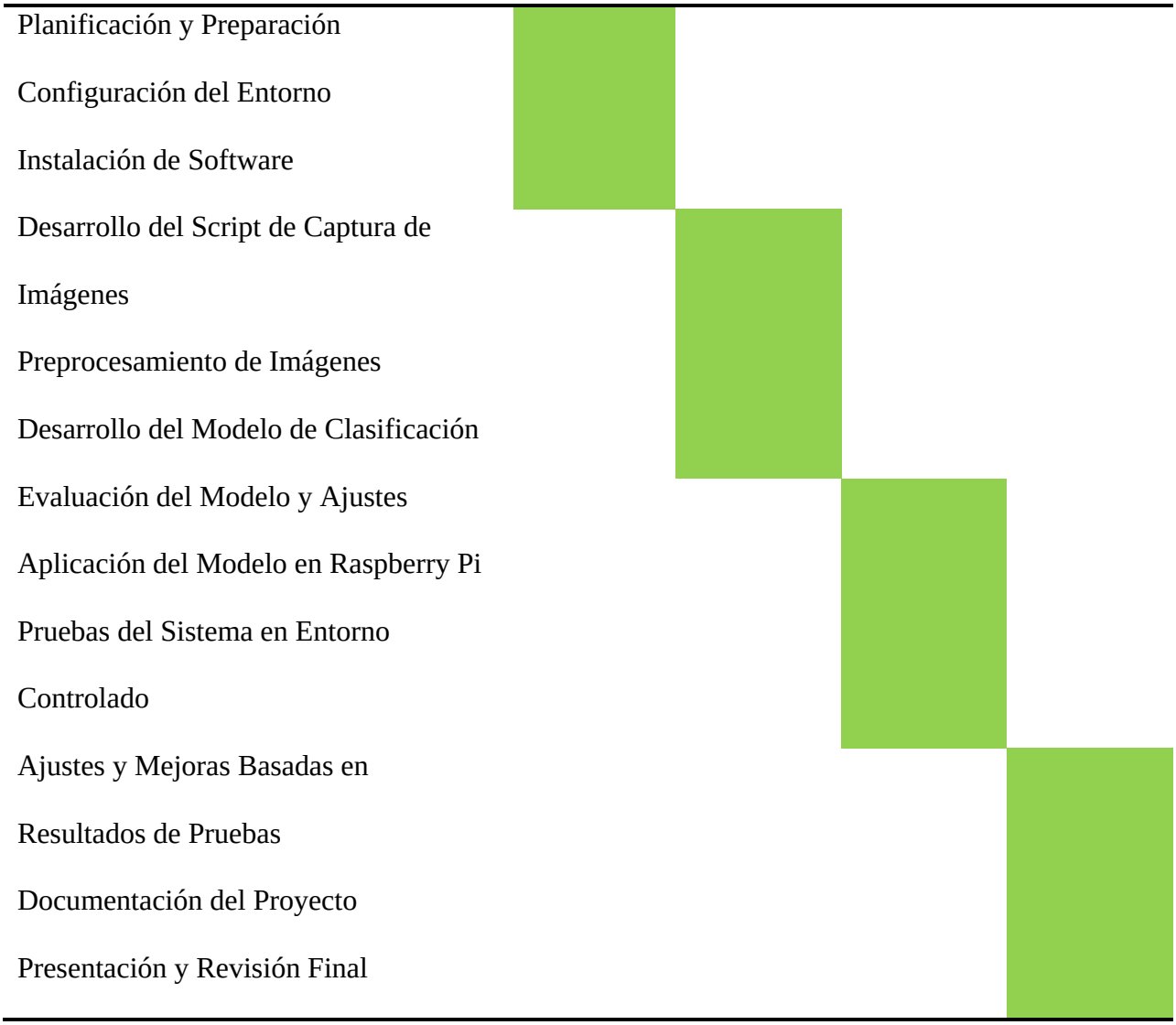
Nota. Casos de fallos identificados en la implementación del modelo de IA

Cronograma

Tabla 11

Cronograma de actividades

Actividad	Mes 1	Mes 2	Mes 3	Mes 4



Nota. Cronograma de actividades a desarrollar en el proyecto

En el Anexo C se incluye el archivo README, el cual contiene la información técnica necesaria para la correcta reproducción del proyecto. En este documento se detallan las versiones de Python y de las librerías utilizadas (PyTorch, YOLOv5s, OpenCV, GPIOzero y Ultralytics), así como los comandos de instalación de las dependencias requeridas. Además, se presentan de manera ordenada los pasos que deben seguirse para ejecutar el entrenamiento y la inferencia en la Raspberry Pi, garantizando la trazabilidad y replicabilidad del sistema implementado.

Conclusiones

Precisión y Fiabilidad en la Clasificación :

El modelo alcanzó una exactitud global del 87%, con valores de precisión de 0.90 para la Pieza Roja, 0.85 para la pieza plateada y 0.88 para la pieza negra. Estos resultados evidencian que el sistema es capaz de discriminar entre diferentes tipos de camisas con un nivel de fiabilidad superior al de sensores convencionales, que suelen presentar tasas de error superiores al 20%.

Asimismo, la capacidad del modelo para identificar características sutiles permitió reducir los errores de clasificación a menos del 13%, lo que se traduce en una detección más temprana de defectos y anomalías (Figuras 13, 14 y 15).

Optimización de la Eficiencia Operativa :

Durante las pruebas, el sistema procesó imágenes a un promedio de 15 fps en la Raspberry Pi 4 usando YOLOv5s, frente a los 3–5 fps obtenidos con métodos tradicionales basados en sensores y procesamiento secuencial. Esto representa una mejora de más del 200% en velocidad de procesamiento, lo que permite acelerar el flujo de producción y garantizar decisiones en tiempo real. La reducción de los tiempos de inspección repercute directamente en un aumento de la productividad general y en la liberación de recursos humanos hacia tareas de mayor valor añadido (**Tabla 9**).

Adaptabilidad y Escalabilidad del Sistema (
Tabla *8* y Tabla 9)

A diferencia de los sistemas basados en sensores que requieren recalibraciones costosas ante cambios en especificaciones, el modelo YOLOv5 mostró la capacidad de reentrenarse con un conjunto reducido de 145 nuevas imágenes, alcanzando nuevamente un desempeño superior al 85% de exactitud en menos de 10 épocas. Esto confirma que la IA implementada no solo amortiza la inversión inicial en el tiempo, sino que asegura la escalabilidad del sistema y su relevancia a medida que evolucionan las necesidades de producción.

Recomendaciones

Para tener un mejor rendimiento en el procesamiento de imágenes en tiempo real es ideal la implementación de una tarjeta con procesador gráfico independiente para poder dedicar el la CPU de la tarjeta para los demás procesos subyacentes de la recopilación y ejecución del modelo de IA.

Con respecto a las conexiones de se deben estandarizar y garantizar que estas cumplan con estándares industriales que garanticen la seguridad de la estación y del proceso.

Referencias Bibliográficas

Abeliuk, A., & Gutiérrez, C. (2021). Historia y evolución de la inteligencia artificial. Revista Bits de Ciencia, 21(1), 14–21. https://doi.org/10.71904/BITS.VI21.2767

Aguilera Martínez, P. (2002). Programación de PLC's [Tesis de maestría, Universidad Autónoma de Nuevo León]. Universidad Autónoma de Nuevo León, Facultad de Ingeniería Mecánica y Eléctrica. http://eprints.uanl.mx/

Baldovino, R. G., Vidad, A. J. P., Abastillas, R. P. B., Bugtai, N. T., Dadios, E. P., Vicerra, R. R. P., & Roxas Jr., N. R. (2024). Comprehensive analysis on Ultralytics-supported YOLO models for detection and recognition of large office objects for indoor navigation. Procedia Computer Science, 246, 3851–3858. https://doi.org/10.1016/j.procs.2024.03.407

Boden, M. A. (2017). Inteligencia artificial. Turner. ISBN 978-84-16714-22-3

Branque Rosales, N. B. (2022, septiembre 22). Implementación de un sistema automatizado, mediante el uso de visión artificial para la clasificación del maracuyá, según su color de madurez y el uso de un sistema SCADA para el monitoreo de la productividad [Trabajo de integración curricular, Universidad Estatal Península de Santa Elena]. Repositorio UPSE. https://repositorio.upse.edu.ec/handle/46000/8467

Challenger-Pérez, I., Díaz-Ricardo, Y., & Becerra-García, R. A. (2014). El lenguaje de programación Python. Ciencias Holguín, 20(2), 1–13. https://www.redalyc.org/articulo.oa?id=181531232001

Chun, W. J. (2001). Core Python Programming (Vol. 1). Prentice Hall Professional. ISBN 0-13-026036-3

García Vega, J. (2024). Estudio e implementación de redes neuronales convolucionales para la segmentación de imágenes [Tesis de máster, Universidad de Salamanca]. Gredos USAL. https://gredos.usal.es/handle/10366/163859

Hernández Rodríguez, J. D. (2022). Mejora del modelo de detección de huecos y bolsas de basura basado en deep learning implementado en una Raspberry Pi (en Bogotá, Colombia) [Trabajo de grado de pregrado, Universidad de los Andes]. Repositorio Uniandes. https://hdl.handle.net/1992/58832

Montoya, A. F. R. (2017). Sistema de visión artificial para la identificación de objetos mediante códigos QR en sistemas automatizados de almacenamiento y recuperación (ASRS) [Trabajo de grado de pregrado, Universidad de San Buenaventura]. Biblioteca Digital USB. https://bibliotecadigital.usb.edu.co/handle/10819/4301

Ortíz Rosas, A. (2018). Programación de PLC, HMI y comunicaciones en la industria. Universidad Autónoma de Occidente. ISBN 978-958-8994-52-9

Pereyra, F. A. (2024). Sistema de visión artificial y robótica para la manipulación automática de objetos reciclables [Práctica Profesional Supervisada, Universidad Nacional Arturo Jauretche]. RID UNAJ. https://rid.unaj.edu.ar/handle/123456789/3310

Ramió Matas, C. (2018). Inteligencia artificial, robótica y modelos de Administración pública. Revista del CLAD Reforma y Democracia, 72, 5–42. https://doi.org/10.69733/clad.ryd.n72.a163

Rouhiainen, L. (2018). Inteligencia artificial. Alienta Editorial. ISBN 9788416883083

Varó, A. M., Luengo, I. G., & Sevilla, P. G. (2014). Introducción a la programación con Python 3. Universitat Jaume I, Servei de Comunicació i Publicacions. ISBN 978-84-8021-889-2

Apéndices

Apéndice A

Programación Ejecutada Desde Python en la Raspberry Pi 4

```
  GNU nano 8.1
#importamos librerias
import torch
import cv2
import numpy as np
import pandas
import time
from gpiozero import LED
import warnings
warnings.simplefilter(action='ignore',category=FutureWarning)
ESPERA = 0.5
led1= LED(17)
led2= LED(27)
led3= LED(22)
led1.on()
led2.on()
led3.on()
print("« Desarrollo opcion de grado Proyecto aplicado »")
print("« Deteccion de cuerpos de cilindros neumaticos por medio de IA »")
print("« Desarrollado por Camilo Arturo Avededo Gutierrez »")
print("« Estudiante de ingenieria electronica »")
print("« UNAD 2025 »")
#leemos el modelopreviemnete entrenado de roboflow y exportado en YOLO V5
model = torch.hub.load('ultralytics/yolov5','custom', path = '/home/arturo/pytorch_env/P

#Realizo videocaptura de la camara web
cap = cv2.VideoCapture(0)

contafot = 0
#Inicio
print("Para iniciar la deteccion  precione ENTER")
input()
print('Realizando deteccion.........')
while True:

        #Realizamos lecturas de frames
        ret, frame = cap.read()

[ Rea
```

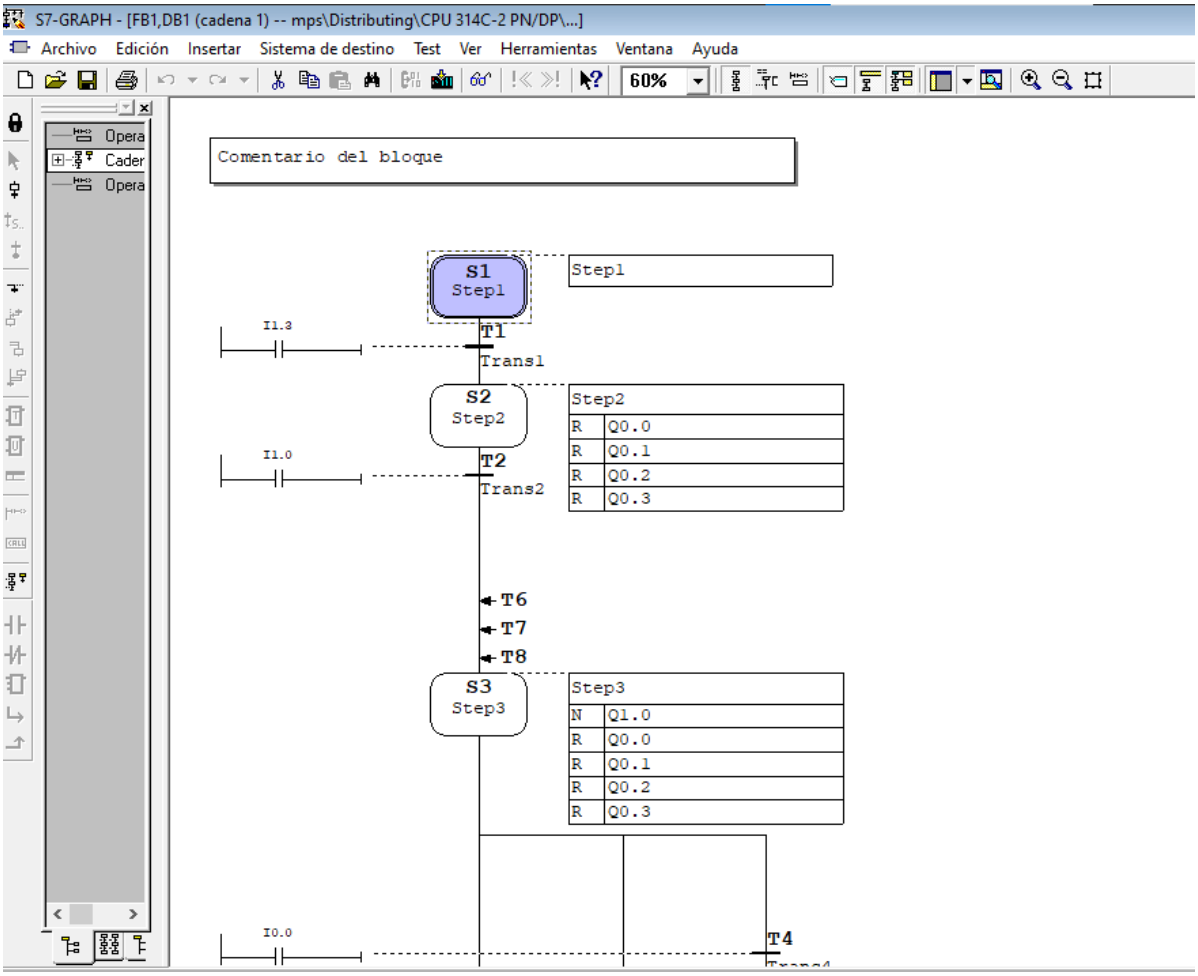
```
GNU nano 8.1
        #Realizamos lecturas de frames
        ret, frame = cap.read()

        #procedemos a hacer un conteo de cada 5 frames lee la imagen.
        contafot += 1
        if contafot % 5 != 0:
            continue

        #realizamos las detecciones
        detect = model(frame, size = 640)

        #imprimimos la informacion de las detecciones
        #info = detect.pandas().xyxy[0]
        #print(info)


        #extraemos la informacion de la deteccion
        info = detect.pandas().xyxy[0].to_dict(orient="records")#predicciones

        #preguntamos si hay detecciones
        if len(info) != 0:
            for result in info:

                conf = result['confidence']

                if conf >= 0.80:
                    print('El % de similitud es de :',conf*100)
                    #clase
                    cls = int(result['class'])
                    #xi
                    x1 = int(result['xmin'])
                    #yi
                    yi = int(result['ymin'])
                    #xf
                    xf = int(result['xmax'])
                    #yf
                    yf = int(result['ymax'])
```

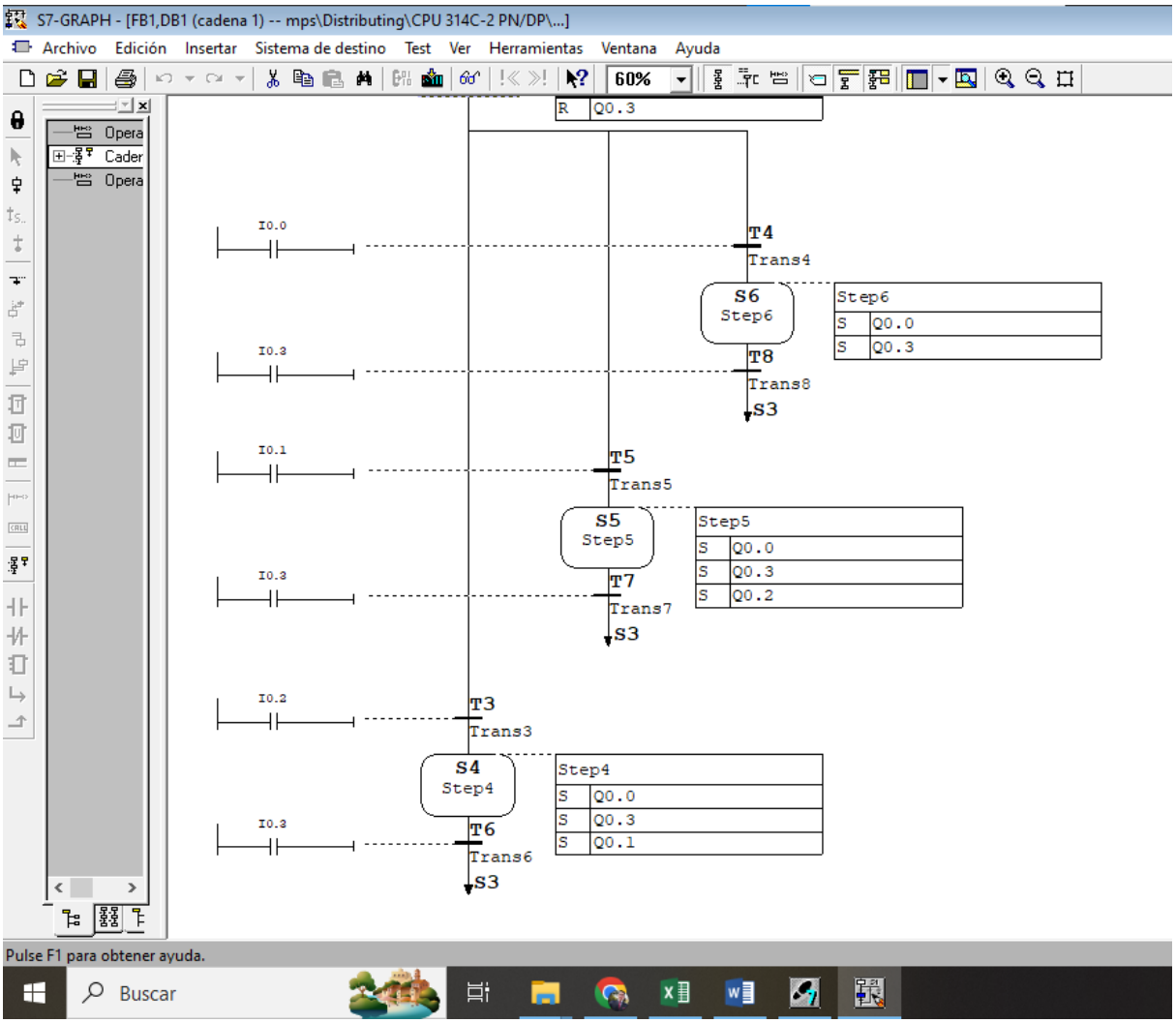
```
 GNU nano 8.1                                                                yolo!
                    #name
                    nm = (result['name'])
                    #print("La pieza es de color = ",nm)
                    if cls == 0:
                        print("La pieza es NEGRA")
                        led1.off()
                        led2.on()
                        led3.on()
                    if cls == 1:
                        print("La pieza es PLATEADA")
                        led1.on()
                        led2.off()
                        led3.on()
                    if cls == 2:
                        print("La pieza es ROJA")
                        led1.on()
                        led2.on()
                        led3.off()

                    break
            else:
                print('No se ha detectado ningun tipo de pieza')
                led1.on()
                led2.on()
                led3.on()

            #print("Para una nueva deteccion Presione ENTER")
            opcion=input('Para una nueva deteccion  precione Y de lo contrario precione N: ')
            if opcion=='N':
                    led1.on()
                    led2.on()
                    led3.on()
                    print('El programa se esta cerrando')
                    print('Apagando Camara.....')
                    print('Apagando Salidas......')
                    print('« Gracias »')
                    break
```

```
 GNU nano 8.1                                                                yolo
                    if cls == 2:
                        print("La pieza es ROJA")
                        led1.on()
                        led2.on()
                        led3.off()

                    break
        else:
            print('No se ha detectado ningun tipo de pieza')
            led1.on()
            led2.on()
            led3.on()

        #print("Para una nueva deteccion Presione ENTER")
        opcion=input('Para una nueva deteccion  precione Y de lo contrario precione N: ')
        if opcion=='N':
                led1.on()
                led2.on()
                led3.on()
                print('El programa se esta cerrando')
                print('Apagando Camara.....')
                print('Apagando Salidas......')
                print('« Gracias »')
                break
        print('Realizando deteccion........')
        #leemos del teclado
        #t = cv2.waitKey(5)
        #if t == 27:
        #       break
        #Mostramos los FPS
        #cv2.imshow('piezas', np.squeeze(detect.render()))
        #cv2.imshow('piezas', frame)  #se muestra la imagen de lo que esta viendo la camara


cap.release()
cv2.destroyAllWindows()
```

Apéndice B

Programación del PLC s7-300




S7-GRAPH - [FB1,DB1 (cadena 1) -- mps\Distributing\CPU 314C-2 PN/DP\...]
Archivo Edición Insertar Sistema de destino Test Ver Herramientas Ventana Ayuda
60%
R Q0.3
I0.0
T4
Trans4
S6
Step6
Step6
S Q0.0
S Q0.3
I0.3
T8
Trans8
S3
I0.1
T5
Trans5
S5
Step5
Step5
S Q0.0
S Q0.3
S Q0.2
I0.3
T7
Trans7
S3
I0.2
T3
Trans3
S4
Step4
Step4
S Q0.0
S Q0.3
S Q0.1
I0.3
T6
Trans6
S3
Pulse F1 para obtener ayuda.
Buscar

Apéndice C

Readme

Proyecto Aplicación de visión artificial basada en Python y Raspberry Pi para la clasificación automática de cilindros neumáticos en la estación MPS clasificación

Este README documenta el entorno, dependencias y pasos necesarios para reproducir el entrenamiento e inferencia del modelo de clasificación de objetos implementado en una Raspberry Pi 4.

Versiones de software utilizadas

- **Python:** 3.9.2 (instalado por defecto en Raspberry Pi OS 64-bit)
- **PyTorch:** 1.13.1
- **Ultralytics (YOLOv5s):** 8.0.20
- **OpenCV:** 4.7.0
- **GPIOzero:** 1.6.2

Instalación de dependencias

Ejecutar los siguientes comandos en la Raspberry Pi 4:

```bash
# Actualizar sistema
sudo apt update && sudo apt upgrade -y

# Instalar Python y pip (si no están instalados)
sudo apt install python3 python3-pip -y

# Instalar PyTorch (CPU en Raspberry Pi)
pip install torch==1.13.1 torchvision==0.14.1

# Instalar Ultralytics (YOLOv5)
pip install ultralytics==8.0.20

# Instalar OpenCV
pip install opencv-python==4.7.0.72

# Instalar GPIOzero para control de hardware
pip install gpiozero==1.6.2

# Dependencias adicionales
pip install matplotlib pandas
```

Pasos para reproducir el entrenamiento

1. Crear cuenta y cargar dataset en [Roboflow](https://roboflow.com/).

2. Exportar dataset en formato YOLOv5.

3. Entrenar el modelo en Google Colab o PC con GPU usando el script de Ultralytics:

   ```bash
   yolo detect train data=data.yaml model=yolov5s.pt epochs=220 imgsz=640 batch=16
   ```

4. Descargar el modelo entrenado (`best.pt`).

Pasos para la inferencia en Raspberry Pi 4

1. Copiar el modelo `best.pt` a la Raspberry Pi (ejemplo usando scp):

   ```bash
   scp best.pt pi@<IP_RASPBERRY>:/home/pi/
   ```

2. Ejecutar inferencia con la cámara USB o CSI:

```python
from ultralytics import YOLO
import cv2

# Cargar modelo
model = YOLO("best.pt")

# Captura de video en tiempo real
cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    if not ret:
        break

    results = model(frame)
    annotated_frame = results[0].plot()

    cv2.imshow("Detección YOLOv5s", annotated_frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
```

```
cv2.destroyAllWindows()
```

3. Para integrar salidas con hardware (ejemplo: encender LED cuando se detecta una pieza):

```python
from gpiozero import LED
from time import sleep

led = LED(17)

# Encender LED cuando se detecta objeto de interés
for result in results:
    if len(result.boxes) > 0:
        led.on()
        sleep(1)
        led.off()
```

Estructura recomendada del proyecto

```
├── data/              # Dataset YOLOv5 exportado de Roboflow
├── runs/              # Resultados de entrenamiento
├── best.pt            # Modelo entrenado
├── detect.py          # Script de inferencia en Raspberry Pi
├── requirements.txt   # Lista de dependencias
└── README.md          # Este archivo
```

##Notas

- Se recomienda refrigeración activa para la Raspberry Pi 4 debido a la carga de trabajo intensiva.

- Para un rendimiento superior puede evaluarse migrar a **Raspberry Pi 5** o dispositivos con GPU integrada (NVIDIA Jetson Nano, Xavier, etc.).

Apéndice D

Video de Puesta en Marcha del Proyecto.

https://youtube.com/video/FovIPAwDqV8