UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO

FACULTAD DE INGENIERÍA CIVIL, DE SISTEMAS Y DE ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS

Sistema robótico inteligente para la detección de defectos y clasificación de calidad en rocotos utilizando redes neuronales convolucionales

Para obtener el Título Profesional de:

Ingeniero(a) de Sistemas

Autores:

Bach. Delgado Bautista, Maria Shadira

Bach. Tocas Diaz, Cristhian Omar

Asesor

Dr. Ing. Villegas Cubas, Juan Elias

Lambayeque, Perú

24 de agosto del 2026

UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO

FACULTAD DE INGENIERÍA CIVIL, DE SISTEMAS Y DE ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS

Sistema robótico inteligente para la detección de defectos y clasificación de calidad en rocotos utilizando redes neuronales convolucionales

Para obtener el Título Profesional de:

Ingeniero(a) de Sistemas

Aprobado por el siguiente jurado:

MSC. ING. OMAR WILTON SAAVEDRA SALAZA
PRESIDENTE

MSC. ING. OSCAR EFRAIN CAPUÑAY UCEDA
VOCAL

MSC. ING. ROBERTO CARLOS ARTEAGA LORA
SECRETARIO

DR. ING. JUAN ELIAS VILLEGAS CUBAS
ASESOR

UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO

FACULTAD DE INGENIERÍA CIVIL, DE SISTEMAS Y DE ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS

Sistema robótico inteligente para la detección de defectos y clasificación de calidad en rocotos utilizando redes neuronales convolucionales

Para obtener el Título Profesional de:

Ingeniero(a) de Sistemas

Autores:



Bach. MARIA SHADIRA DELGADO BAUTISTA

AUTOR



Bach. CRISTHIAN OMAR TOCAS DIAZ

AUTOR

UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO
FACULTAD DE INGENIERÍA CIVIL DE SISTEMAS Y DE ARQUITECTURA
UNIDAD DE INVESTIGACIÓN



ACTA DE SUSTENTACIÓN N°014-2026-UI-FICSA



Siendo las 12:00 pm horas del día 24 de agosto del 2026, se reunieron de manera presencial los miembros de jurado de la tesis titulada: **"SISTEMA ROBÓTICO INTELIGENTE PARA LA DETECCIÓN DE DEFECTOS Y CLASIFICACIÓN DE CALIDAD EN ROCOTOS UTILIZANDO REDES NEURONALES CONVOLUCIONALES", con código N° IS_V_2025_013,** designado por **Resolución Decanal N° 540-2025-UNPRG-FICSA** con la finalidad de Evaluar y Calificar la sustentación de la tesis antes mencionada, conformado por los siguientes docentes:

MSC. ING. OMAR WILTON SAAVEDRA SALAZAR	PRESIDENTE
MSC. ING. ROBERTO CARLOS ARTEAGA LORA	SECRETARIO
MSC. ING. OSCAR EFRAIN CAPUÑAY UCEDA	VOCAL

Asesorado por **DR. ING. JUAN ELIAS VILLEGAS CUBAS.**

El acto de sustentación fue autorizado por **OFICIO VIRTUAL N° 199-2026-UIFICSA**, la tesis fue presentada y sustentada por los Bachilleres: **MARIA SHADIRA DELGADO BAUTISTA Y CRISTHIAN OMAR TOCAS DIAZ,** tuvo una duración de 120 minutos. Después de la sustentación, y absueltas las preguntas y observaciones de los miembros del jurado; se procedió a la calificación respectiva:

	NUMERO	LETRAS	CALIFICATIVO
MARIA SHADIRA DELGADO BAUTISTA	20	Veinte	Excelente
CRISTHIAN OMAR TOCAS DIAZ	20	Veinte	Excelente

Por lo que quedan **APTOS** para obtener el Título Profesional de **INGENIERO (A) DE SISTEMAS** de acuerdo con la Ley Universitaria 30220 y la normatividad vigente de la Facultad de Ingeniería Civil De Sistemas y de Arquitectura de la Universidad Nacional Pedro Ruiz Gallo.

Siendo las 2:00PM Se dio por concluido el presente acto académico, dándose conformidad al presente acto, con la firma de los miembros del jurado.

MSC. ING. OMAR WILTON SAAVEDRA SALAZAR
PRESIDENTE

MSC. ING. ROBERTO CARLOS ARTEAGA LORA
SECRETARIO

MSC. ING. OSCAR EFRAIN CAPUÑAY UCEDA
VOCAL

DR. ING. JUAN ELIAS VILLEGAS CUBAS
ASESOR

JHFF/bcm

Escaneado con CamScanner

UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO FACULTAD DE INGENIERÍA CIVIL, DE SISTEMAS Y DE ARQUITECTURA
UNIDAD DE INVESTIGACIÓN

CONSTANCIA DE VERIFICACIÓN DE ORIGINALIDAD

Yo, Dr. Ing. Juan Elias Villegas Cubas**, asesor de tesis de los bachilleres**:

Delgado Bautista, Maria Shadira y

Tocas Diaz, Cristhian Omar

TITULADA:

"Sistema robótico inteligente para la detección de defectos y clasificación de calidad en rocotos utilizando redes neuronales convolucionales"

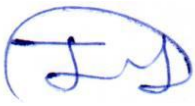
Luego de la revisión exhaustiva del documento constato que la misma tiene un índice de similitud de 12% verificable en el reporte de similitud del programa TURNITIN.

El suscrito analizó dicho reporte y concluyó que cada una de las coincidencias detectadas NO CONSTITUYEN PLAGIO. A mi leal saber y entender la tesis cumple con todas las normas para el uso de citas y referencias establecidas por la Universidad Nacional Pedro Ruiz Gallo.

Se expide la presente según lo dispuesto en la Resolución Nº 659-2020-R, de fecha 8 de setiembre de 2020 formativa para la obtención de Grados y Títulos de la UNPRG:

Lambayeque, 7 de julio del 2026

ATENTAMENTE,



Dr. Ing. Juan Elias Villegas Cubas
DNI. 80103991
Asesor

Se adjunta:
Recibo digital de Turnitin
Revisión de informe en Turnitin

Sistema robótico inteligente para la detección de defectos y clasificación de calidad en rocotos utilizando redes neuronales convolucionales

INFORME DE ORIGINALIDAD

12%	10%	3%	5%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	repositorio.unprg.edu.pe Fuente de Internet	1%
2	Submitted to UNIBA Trabajo del estudiante	<1%
3	www.coursehero.com Fuente de Internet	<1%
4	ri-ng.uaq.mx Fuente de Internet	<1%
5	repositorio.unap.edu.pe Fuente de Internet	<1%
6	repositorio.utdt.edu Fuente de Internet	<1%
7	Submitted to Universidad Católica Boliviana "San Pablo" Trabajo del estudiante	<1%
8	Submitted to Universidad Internacional de la Rioja Trabajo del estudiante	<1%
9	www.agroperu.pe Fuente de Internet	<1%
10	www.mdpi.com Fuente de Internet	<1%
11	repositorio.chapingo.edu.mx Fuente de Internet	<1%
12	Submitted to Universidad Europea de Madrid Trabajo del estudiante	<1%
13	hdl.handle.net Fuente de Internet	<1%
14	Submitted to Universidad Mariano Gálvez de Guatemala Trabajo del estudiante	<1%
	www.campustudionline.com	



Dr. Ing. Juan Elias Villegas Cubas
DNI. 80103991
Asesor

15	Fuente de Internet	<1 %
16	Submitted to Universidad TecMilenio Trabajo del estudiante	<1 %
17	repositorio.uss.edu.pe Fuente de Internet	<1 %
18	Submitted to Universitat Oberta de Catalunya Trabajo del estudiante	<1 %
19	Submitted to Universidad de Málaga Trabajo del estudiante	<1 %
20	Submitted to Universitat Politècnica de València Trabajo del estudiante	<1 %
21	repositorio.upagu.edu.pe Fuente de Internet	<1 %
22	repository.unipiloto.edu.co:8080 Fuente de Internet	<1 %
23	repositorioacademico.upc.edu.pe Fuente de Internet	<1 %
24	Submitted to Consorcio CIXUG Trabajo del estudiante	<1 %
25	Aza de La Espriella, Daniel Felipe. "Predicción de modificaciones en la contratación estatal colombiana.", Universidad de La Sabana (Colombia) Publicación	<1 %
26	repositorio.unjbg.edu.pe Fuente de Internet	<1 %
27	mdpi-res.com Fuente de Internet	<1 %
28	crea.ujaen.es Fuente de Internet	<1 %
29	sedici.unlp.edu.ar Fuente de Internet	<1 %
30	Submitted to Uniminuto Virtual Trabajo del estudiante	<1 %
31	laccei.org Fuente de Internet	<1 %



Dr. Ing. Juan Elias Villegas Cubas
DNI. 80103991
Asesor

32	oa.upm.es Fuente de Internet	<1 %
33	repositorio.upea.bo Fuente de Internet	<1 %
34	repository.usta.edu.co Fuente de Internet	<1 %
35	Submitted to Fundación Universitaria del Area Andina Trabajo del estudiante	<1 %
36	dokumen.pub Fuente de Internet	<1 %
37	repositorio.uceva.edu.co Fuente de Internet	<1 %
38	Submitted to Universidad de Piura Trabajo del estudiante	<1 %
39	repositorio.usm.cl Fuente de Internet	<1 %
40	Submitted to Universidad de Salamanca Trabajo del estudiante	<1 %
41	repositorio.utn.edu.ec Fuente de Internet	<1 %
42	Calli Olvea, Javier. "Deep learning para la detección e identificación de estudiantes en el Instituto de Educación Superior Tecnológico Publico Espinar, 2023", Universidad Nacional del Altiplano de Puno (Peru), 2025 Publicación	<1 %
43	repositorio.ulp.edu.pe Fuente de Internet	<1 %
44	repositorio.uta.edu.ec Fuente de Internet	<1 %
45	repositorio.unicesmag.edu.co:8080 Fuente de Internet	<1 %
46	vitela.javerianacali.edu.co Fuente de Internet	<1 %
47	www.risti.xyz Fuente de Internet	<1 %
48	www.researchgate.net Fuente de Internet	



Dr. Ing. Juan Elias Villegas Cubas
DNI. 80103991
Asesor

		<1 %
49	Submitted to Instituto Tecnologico de Costa Rica Trabajo del estudiante	<1 %
50	Submitted to University of La Guajira Trabajo del estudiante	<1 %
51	dspace.ups.edu.ec Fuente de Internet	<1 %
52	repositorio.uladech.edu.pe Fuente de Internet	<1 %
53	www.slideshare.net Fuente de Internet	<1 %
54	Submitted to Universidad Carlos III de Madrid - EUR Trabajo del estudiante	<1 %
55	Submitted to Universidad Nacional de Loja Trabajo del estudiante	<1 %
56	plantuml.com Fuente de Internet	<1 %
57	alicia.concytec.gob.pe Fuente de Internet	<1 %
58	repositorio.continental.edu.pe Fuente de Internet	<1 %
59	revistas.lamolina.edu.pe Fuente de Internet	<1 %
60	Submitted to ITESM: Instituto Tecnologico y de Estudios Superiores de Monterrey Trabajo del estudiante	<1 %
61	Submitted to Universidad Estatal Amazonica- Trabajo del estudiante	<1 %
62	Submitted to Universidad Internacional Isabel I de Castilla Trabajo del estudiante	<1 %
63	es.unionpedia.org Fuente de Internet	<1 %
64	repositorio.uigv.edu.pe Fuente de Internet	<1 %
65	Submitted to University of Wolverhampton Trabajo del estudiante	



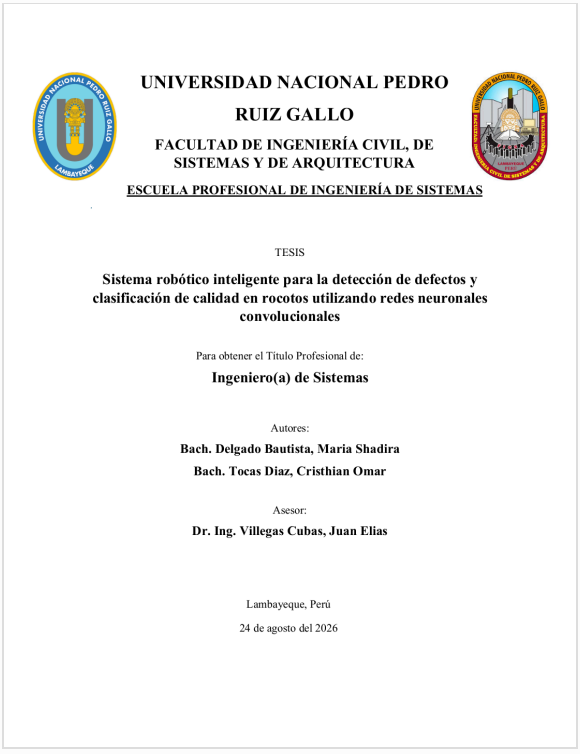
Dr. Ing. Juan Elias Villegas Cubas
DNI. 80103991
Asesor

Recibo digital

Este recibo confirma quesu trabajo ha sido recibido por Turnitin. A continuación podrá ver la información del recibo con respecto a su entrega.

La primera página de tus entregas se muestra abajo.

Autor de la entrega:	Delgado Bautista, Maria Shadira Tocas Diaz, Cristhian Omar
Título del ejercicio:	Quick Submit
Título de la entrega:	Sistema robótico inteligente para la detección de defectos y cl...
Nombre del archivo:	Tesis_Rocobot_Delgado_Tocas.pdf
Tamaño del archivo:	21.86M
Total páginas:	167
Total de palabras:	41,850
Total de caracteres:	241,127
Fecha de entrega:	21-sept-2026 11:00a. m. (UTC-0500)
Identificador de la entrega:	3038244345



UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO

FACULTAD DE INGENIERÍA CIVIL, DE SISTEMAS Y DE ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS

Sistema robótico inteligente para la detección de defectos y clasificación de calidad en rocotos utilizando redes neuronales convolucionales

Para obtener el Título Profesional de:

Ingeniero(a) de Sistemas

Autores:

Bach. Delgado Bautista, Maria Shadira

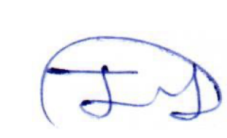
Bach. Tocas Diaz, Cristhian Omar

Asesor:

Dr. Ing. Villegas Cubas, Juan Elias

Lambayeque, Perú

24 de agosto del 2026



Dr. Ing. Juan Elias Villegas Cubas
DNI. 80103991
Asesor

Derechos de autor 2026 Turnitin. Todos los derechos reservados.

DEDICATORIA

A Dios, por acompañarme y darme la fortaleza para llegar hasta aquí. A mi mamá, que desde el cielo sigue guiando mis pasos; por su amor, sus enseñanzas y por ser mi mayor inspiración. A mi padre Timoteo, por su apoyo incondicional, confianza y por enseñarme el valor del esfuerzo. A mi familia Delgado Bautista, por ser parte fundamental de mi vida, por su esfuerzo, unión y apoyo constante; especialmente a mis hermanos Lenin, Nieves y Nancy, por estar siempre a mi lado. Este logro es para ustedes y para nuestra familia, como una forma de retribuir todo su esfuerzo y como un aporte para mejorar el proceso de control de calidad del rocoto en nuestro negocio familiar. A Jhustin, por su apoyo y compañía durante este camino. Con todo mi amor y gratitud.

María Shadira Delgado Bautista

A Dios, por iluminar mi camino, darme fortaleza en los momentos difíciles y bendecir cada paso que me condujo hasta esta meta. A mi mamá, Sulema Díaz, por su amor incondicional, sacrificio y esfuerzo constante, que han sido la base de cada uno de mis logros. A mis abuelitos, Saturnino Díaz y Elva León, por su cariño, sus enseñanzas y el ejemplo de perseverancia que me han brindado a lo largo de mi vida. A mi compañera de vida, Geral Llatas, por acompañarme en este camino con paciencia, confianza y motivación, impulsándome a seguir adelante incluso en los momentos más desafiantes. Este logro también les pertenece, porque en cada página de este trabajo se refleja una parte del amor, apoyo y confianza que me brindaron para hacerlo posible.

Cristhian Omar Tocas Diaz

AGRADECIMEINTO

A Dios, por guiarnos en cada etapa de este camino y brindarnos la fortaleza, sabiduría y perseverancia necesarias para culminar satisfactoriamente esta importante etapa de nuestra formación profesional.

A la Universidad Nacional Pedro Ruiz Gallo y a la Escuela Profesional de Ingeniería de Sistemas, por brindarnos la formación académica y las herramientas necesarias para nuestro desarrollo profesional.

A nuestros docentes, por compartir sus conocimientos, experiencias y enseñanzas, contribuyendo significativamente a nuestra formación.

De manera especial, expresamos nuestro profundo agradecimiento a nuestro asesor, Ing. Juan Villegas, por su constante apoyo, orientación y confianza durante el desarrollo de esta investigación. Agradecemos su disposición para acompañarnos en cada etapa, sus valiosas recomendaciones y, sobre todo, su compromiso para guiarnos hasta la culminación de este trabajo.

Asimismo, agradecemos a todas las personas e instituciones que, de manera directa o indirecta, contribuyeron al desarrollo de esta investigación.

A nuestras familias, por su comprensión, confianza y apoyo constante, y a nuestros amigos y compañeros, por la motivación y experiencias compartidas a lo largo de este camino.

A todos ellos, nuestro más sincero agradecimiento.

Los Autores

RESUMEN

La clasificación manual del rocoto (*Capsicum pubescens*) en la industria agroalimentaria peruana presenta limitaciones como subjetividad en la evaluación, variabilidad en los criterios de calidad, altos costos de mano de obra y baja eficiencia ante grandes volúmenes de producción. Frente a ello, la presente investigación tuvo como objetivo desarrollar un prototipo de sistema robótico, denominado ROCOBOT, para la detección y clasificación automática del estado del rocoto mediante técnicas de Deep Learning integradas con visión artificial, procesamiento embebido y un aplicativo web para la supervisión remota del proceso. La investigación es de tipo aplicada, de enfoque cuantitativo y de diseño tecnológico-experimental, desarrollada bajo la metodología CRISP-DM; se construyeron dos datasets propios, uno de detección de 5,421 imágenes obtenidas al aplicar *data augmentation* exclusivamente sobre las 1,348 imágenes etiquetadas del subconjunto de entrenamiento dentro de un conjunto base de 2,725 imágenes, y uno de clasificación balanceado de 2,732 imágenes (1,366 por clase), sobre los cuales se entrenaron y compararon seis modelos de detección (YOLOv5s, YOLOv5m, YOLOv5l, YOLOv8n, YOLOv8s e YOLOv11n) y dos de clasificación (ResNet18 y EfficientNetB0) mediante transfer learning con fine-tuning. YOLOv8n obtuvo el mejor desempeño en detección (mAP@0.5:0.95 = 81.3%, precisión = 96.2%, recall = 96.1%) y ResNet18 el mejor en clasificación (accuracy = 99.64%, AUC = 0.9999); con ambos modelos se diseñó, construyó e implementó el prototipo físico ROCOBOT y el aplicativo web ROCOBOT Web para el monitoreo en tiempo real y la trazabilidad de resultados, validando el sistema mediante pruebas piloto sobre 150 rocotos con un accuracy de 89.33% frente al criterio de un experto. Los resultados demuestran la viabilidad técnica de automatizar el control de calidad del rocoto, reduciendo la dependencia de la inspección manual y mejorando la eficiencia agroindustrial.

Palabras clave: Deep Learning, visión artificial, clasificación de rocotos, YOLOv8, ResNet18

ABSTRACT

The manual classification of rocoto pepper (*Capsicum pubescens*) in the Peruvian agri-food industry presents limitations such as subjectivity in evaluation, variability in quality criteria, high labor costs, and low efficiency with large production volumes. Accordingly, this research aimed to develop a robotic system prototype, named ROCOBOT, for the automatic detection and classification of rocoto pepper condition through Deep Learning techniques integrated with computer vision, embedded processing, and a web application for remote supervision of the process. The research is applied in purpose, quantitative in approach, and technological-experimental in design, developed under the CRISP-DM methodology; two proprietary datasets were built, one for detection comprising 5,421 images obtained by applying data augmentation exclusively to the 1,348 labeled images of the training subset within a base set of 2,725 images, and one for classification comprising 2,732 balanced images (1,366 per class), on which six detection models (YOLOv5s, YOLOv5m, YOLOv5l, YOLOv8n, YOLOv8s, and YOLOv11n) and two classification models (ResNet18 and EfficientNetB0) were trained and compared using transfer learning with fine-tuning. YOLOv8n achieved the best detection performance (mAP@0.5:0.95 = 81.3%, precision = 96.2%, recall = 96.1%) and ResNet18 the best classification performance (accuracy = 99.64%, AUC = 0.9999); both models were used to design, build, and implement the physical ROCOBOT prototype and the ROCOBOT Web application for real-time monitoring and result traceability, validating the system through pilot tests on 150 rocoto peppers with an accuracy of 89.33% against expert criteria. The results demonstrate the technical viability of automating rocoto pepper quality control, reducing dependence on manual inspection and improving agro-industrial efficiency.

Keywords: Deep Learning, computer vision, rocoto classification, YOLOv8, ResNet 18

Tabla de contenido

1. INTRODUCCIÓN 1

1.1. Síntesis de la situación problemática 1

1.2. Formulación del problema de investigación 2

1.3. Objetivos 2

1.3.1. Objetivo General 2

1.3.2. Objetivos específicos 2

1.4. Justificación de la investigación 2

1.5. Estructura del informe 3

2. DISEÑO TEÓRICO 4

2.1. Antecedentes de la investigación 4

2.2. Bases teóricas 12

2.2.1. Bases teóricas del Rocoto 12

2.2.2. Bases Teóricas de Inteligencia Artificial 17

2.2.3. Bases Teórica de robótica 29

2.2.4. Bases teóricas de desarrollo de software 31

2.3. Operacionalización de variable 35

2.3.1. Definición conceptual 35

2.3.2. Definición operativa 35

3. DISEÑO METODOLÓGICO 37

3.1. Tipo de investigación 37

3.2. Métodología empleada: CRISP-DM 37

3.3. Flujo de Trabajo del Sistema 38

3.4. Metodología de Trabajo 41

3.4.1. Principios de Kanban aplicados al proyecto 41

3.4.2. Estructura del tablero Kanban 42

3.5. Población y muestra 42

3.5.1. Población 42

3.5.2. Muestra 42

3.6. Técnicas, instrumentos, equipos y materiales 43

3.6.1. Técnicas 43

3.6.2. Instrumentos 44

3.6.3. Equipos 44

3.6.4. Materiales 44

3.6.5. Aspectos éticos de la investigación 44

4. RESULTADOS 46

4.1. Construcción de los datasets de imágenes de rocotos 46

4.1.1. Proceso de adquisición de imágenes 46

4.1.2. Dataset de detección 48

4.1.3. Dataset de clasificación 51

4.2. Entrenamiento y comparación de modelos de detección y clasificación 54

4.2.1. Condiciones de entrenamiento 54

4.2.2. Selección de modelo de detección 54

4.2.3. Entrenamiento de los modelos de detección 56

4.2.4. Selección del modelo de clasificación 79

4.2.5. Entrenamiento de los modelos de clasificación 80

4.3. Diseño, construcción e implementación del prototipo ROCOBOT 93

4.3.1. Selección y justificación de componentes 93

4.3.2. Componentes del prototipo 94

4.3.3. Diagrama de flujo de cada módulo 103

4.3.4. Diagramas de conexiones eléctricas del sistema ROCOBOT 107

4.3.5. Diseño de la estructura física 110

4.3.6. Construcción del prototipo final 114

4.4. Desarrollo del aplicativo web para la supervisión remota 115

4.4.1. Selección y justificación del sistema web 115

4.4.2. Arquitectura del sistema 116

4.4.3. Desarrollo del sistema web 119

4.4.4. Funcionamiento del sistema en tiempo real 127

4.5. Validación mediante pruebas piloto del sistema 130

4.5.1. Antecedentes de validación en sistemas robóticos de clasificación 130

4.5.2. Integración del sistema ROCOBOT 131

4.5.3. Resultados de las pruebas piloto 131

4.5.4. Validación contra el criterio experto 135

4.5.5. Análisis comparativo de la validación piloto 137

5. DISCUSIÓN DE RESULTADOS 140

6. CONCLUSIONES 143

7. RECOMENDACIONES..... 145

8. REFERENCIAS..... 146

9. ANEXOS..... 153

Índice de figuras

Figura 1. Mapa de distribución geográfica del rocoto en el Perú por región 14

Figura 2. Subáreas de la Inteligencia Artificial 16

Figura 3. Arquitectura de Red Neuronal Convolucional (CNN) 21

Figura 4. Arquitecturas YOLOv5 23

Figura 5. Arquitecturas YOLOv8 24

Figura 6. Arquitectura de EfficientNet 25

Figura 7. Bloque residual de la arquitectura ResNet 26

Figura 8. Metodología CRISP-DM aplicada al desarrollo del sistema ROCOBOT 36

Figura 9. Flujo de desarrollo del sistema ROCOBOT 37

Figura 10. Tablero Kanban aplicado al desarrollo del proyecto ROCOBOT 41

Figura 11. Flujo de construcción del dataset de detección ROCOBOT 46

Figura 12. Flujo de construcción del dataset de clasificación ROCOBOT 46

Figura 13. Comparación del tamaño del dataset de detección ROCOBOT vs. Antecedentes 47

Figura 14. Comparación del dataset de clasificación ROCOBOT vs. antecedentes de la literatura 51

Figura 15. Distribución de instancias y posición de cajas delimitadoras del dataset de detección .. 54

Figura 16. Correlograma de etiquetas del dataset de detección 55

Figura 17. Curvas de entrenamiento y validación del modelo YOLOv5s 57

Figura 18. Curva precisión-recall del modelo YOLOv5s 57

Figura 19. Matriz de confusión normalizada del modelo YOLOv5s 58

Figura 20. Curvas de entrenamiento y validación del modelo YOLOv5m 59

Figura 21. Curva precisión-recall del modelo YOLOv5m 61

Figura 22. Matriz de confusión normalizada del modelo YOLOv5m 62

Figura 23. Curvas de entrenamiento y validación del modelo YOLOv5l — 30 épocas 63

Figura 24. Curva precisión-recall del modelo YOLOv5l 64

Figura 25. Matriz de confusión normalizada del modelo YOLOv5l 64

Figura 26. Curvas de entrenamiento y validación del modelo YOLOv8n 66

Figura 27. Curva precisión-recall del modelo YOLOv8n 67

Figura 28. Matriz de confusión del modelo YOLOv8n 68

Figura 29. Curvas de entrenamiento y validación del modelo YOLOv8s — 70 épocas 70

Figura 30. Curva precisión-recall del modelo YOLOv8s 71

Figura 31. Matriz de confusión del modelo YOLOv8s 71

Figura 32. Curvas de entrenamiento y validación del modelo YOLOv11n 73

Figura 33. Curva precisión-recall del modelo YOLOv11n 74

Figura 34. Matriz de confusión del modelo YOLOv11n 75

Figura 35. Arquitectura del modelo YOLOv8n para la etapa de detección 79

Figura 36. Curvas de evolución de sus propiedades del modelo ResNet18 81

Figura 37. Evolución del F1-score y AUC del modelo ResNet18 82

Figura 38. Matriz de confusión del modelo ResNet18 — conjunto de prueba (n = 278) 82

Figura 39. Curva ROC y curva precisión-recall del modelo ResNet18 83

Figura 40. Curvas de evolución las propiedades del modelo EfficientNetB0 85

Figura 41. Evolución del F1-score y AUC del modelo EfficientNetB0 86

Figura 42. Matriz de confusión del modelo EfficientNetB0 — conjunto de prueba (n = 278) 87

Figura 43. Curva ROC y precisión-recall del modelo EfficientNetB0 88

Figura 44. Arquitectura del modelo ResNet18 para la etapa de clasificación 92

Figura 45. Flujo del módulo de alimentación y regulación energética 103

Figura 46. Flujo del módulo de banda transportadora 104

Figura 47. Flujo del módulo de detección 105

Figura 48. Flujo del módulo de clasificación 106

Figura 49. Diagrama de conexiones de los servomotores MG995 con convertidores LM2596 107

Figura 50. Diagrama de conexiones de los sensores FC-51 y con convertidores LM2596 108

Figura 51. Diagrama de conexiones del motoreductor JGB37-520 DC12V y driver L298N 109

Figura 52. Diagrama general de conexiones del sistema ROCOBOT 109

Figura 53. Vista superior del prototipo ROCOBOT — diseño 2D en AutoCAD 111

Figura 54. Vista isométrica 3D del prototipo ROCOBOT — modelo CAD 111

Figura 55. Vista general 3D del prototipo ROCOBOT 112

Figura 56. Vista Superior del Rocobot....113
Figura 57. Prototipo ROCOBOT construido — vista frontal y vista lateral....114
Figura 58. Arquitectura general del sistema ROCOBOT. 118
Figura 59. Estructura de carpetas del Backend ROCOBOT 120
Figura 60. Documentación interactiva Swagger UI del backend ROCOBOT....122
Figura 61. Entorno de producción del backend ROCOBOT en Railway 123
Figura 62. Estructura de carpetas del Frontend ROCOBOT....123
Figura 63. Plantillas de correo transaccional configuradas en EmailJS para ROCOBOT....124
Figura 64. Variables de entorno del frontend ROCOBOT configuradas en Vercel 125
Figura 65. Diagrama entidad-relación del esquema core de la base de datos ROCOBOT....126
Figura 66. Pantalla de inicio de sesión del aplicativo web ROCOBOT 128
Figura 67. Pantalla de registro de cuenta del aplicativo web ROCOBOT....129
Figura 68. Panel principal del aplicativo web — detección de rocoto en tiempo real....129
Figura 69. Cuadro de mando del aplicativo web — reporte histórico de clasificación 130
Figura 70. Muestra representativa de 10 ejecuciones — pruebas pilo....132
Figura 71. Distribución de clasificaciones obtenidas en las pruebas piloto (n=150)....133
Figura 72. Confianza promedio del modelo por clase — pruebas piloto (n=150)....134
Figura 73. Tiempo de procesamiento por ejecución — pruebas piloto (n=150)....134
Figura 74. Confianza por ejecución — pruebas piloto (n=150)....135
Figura 75. Comparación de clasificación real vs. predicha — pruebas piloto (n=150)....136
Figura 76. Matriz de confusión de las pruebas piloto (n=150) 137

Índice de tablas

Tabla 1. Matriz de confusión 17

Tabla 2. Operacional de variables del proyecto 34

Tabla 3. Distribución del dataset de detección por subconjunto y tipo de imagen 49

Tabla 4. Distribución final del dataset de detección (tras data augmentation en entrenamiento) 49

Tabla 5. Distribución del dataset de clasificación por subconjunto y clase 52

Tabla 6. Modelos de detección de objetos empleados en antecedentes de productos agrícolas 53

Tabla 7. Métricas de evaluación del modelo YOLOv5s — mejor epoch (época 27 de 30) 59

Tabla 8. Métricas de evaluación del modelo YOLOv5m — mejor epoch (época 21 de 30) 62

Tabla 9. Métricas de evaluación del modelo YOLOv5l — mejor epoch 65

Tabla 10. Métricas de evaluación del modelo YOLOv8n — mejor época 69

Tabla 11. Métricas de evaluación del modelo YOLOv8s — métricas finales 72

Tabla 12. Métricas de evaluación del modelo YOLOv11n — métricas finales 76

Tabla 13. Comparación de métricas de los seis modelos de detección 76

Tabla 14. Comparación de métricas de los seis modelos de detección vs. Antecedentes 77

Tabla 15. Modelos de clasificación empleados en antecedentes de productos agrícolas 79

Tabla 16. Métricas de evaluación del modelo ResNet18 — conjunto de prueba (n = 278) 84

Tabla 17. Métricas de evaluación del modelo EfficientNetB0 — conjunto de prueba (n = 278) 88

Tabla 18. Comparación directa ResNet18 vs. EfficientNetB0 89

Tabla 19. Comparación de métricas de los modelos de clasificación vs. Antecedentes 91

Tabla 20. Listado de materiales para ROCOBOT 95

Tabla 21. Comparación del aplicativo web ROCOBOT vs. antecedentes de la literatura 116

Tabla 22. Antecedentes de validación de sistemas robóticos de clasificación de frutos 131

Tabla 23. Indicadores de rendimiento operativo del sistema ROCOBOT 133

Tabla 24. Métricas de validación de las pruebas piloto — verificación experta (n=150) 138

Tabla 25. Comparación del accuracy de las pruebas piloto vs. Antecedentes 138

Tabla 26. Resumen de los esquemas documentados en el Diccionario de Base de Datos 155

1. INTRODUCCIÓN

1.1. Síntesis de la situación problemática

En la industria agroalimentaria, la clasificación de los frutos o productos agrícolas como el rocoto, es un proceso crítico que a la fecha demanda mucha inspección manual. Sin embargo, es necesario precisar que se ve limitada por diversos factores como la subjetividad en la evaluación, la variabilidad en los criterios de calidad, los altos costes por mano de obra y la baja eficiencia frente a grandes volúmenes de producción.

De acuerdo con el Programa de las Naciones Unidas para el Medio Ambiente (UNEP, 2021), aproximadamente el 17 % de los alimentos disponibles para consumo humano se desperdician, lo que equivale a 931 millones de toneladas, siendo los hogares, los restaurantes y el comercio minorista los principales responsables de este desperdicio.

Por otro lado, los avances científicos han permitido el desarrollo en diversos sectores como la medicina, la educación, la seguridad, el sector agrícola. Siendo este último en el que nos enfocaremos. Para abordar la problemática actual que es el desperdicio de los rocotos al no llevar un control de calidad eficiente (estado saludable y defectuoso), se propone la implementación de un Sistema Robótico inteligente para la detección de defectos y clasificación de calidad en Rocotos utilizando Redes Neuronales Convolucionales. Este sistema permitirá la detección automática de defectos y la clasificación de calidad de los rocotos, optimizando el proceso, reduciendo pérdidas y mejorando la eficiencia en la industria agroalimentaria.

La región Amazonas concentra una parte importante de la producción nacional de rocoto, actividad de la cual dependen económicamente numerosas familias de pequeños productores y distribuidores agroindustriales. La ausencia de un control de calidad automatizado no solo genera pérdidas económicas por el descarte incorrecto de frutos en buen estado, sino que también limita la capacidad de estos productores para competir en mercados que exigen estándares de calidad cada vez más exigentes, afectando directamente su sostenibilidad y desarrollo. En este sentido, el presente proyecto se enmarca dentro del área de investigación de Ingeniería de Sistemas y la línea de investigación de Ingenierías y Tecnologías, respondiendo tanto a una necesidad tecnológica del sector agroindustrial regional como a una demanda social de soluciones que optimicen procesos productivos tradicionales mediante la aplicación de tecnologías emergentes.

1.2. Formulación del problema de investigación

¿Un sistema robótico inteligente con redes neuronales convolucionales puede detectar defectos y clasificar la calidad en rocotos?

1.3. Objetivos

1.3.1. Objetivo General

Desarrollar un prototipo de sistema robótico, denominado ROCOBOT, para la detección y clasificación automática del estado del rocoto (Capsicum pubescens) mediante técnicas de Deep Learning integradas con visión artificial, procesamiento embebido y un aplicativo web para la supervisión remota del proceso.

1.3.2. Objetivos específicos

- Construir los datasets de imágenes de rocotos para detección, con imágenes etiquetadas, y para clasificación, con imágenes distribuidas en conjuntos de entrenamiento, validación y prueba.
- Entrenar y comparar modelos de redes neuronales convolucionales mediante Deep Learning, como YOLOv5, YOLOv8 e YOLOv11 para detección, ResNet18 y EfficientNetB0 para clasificación, empleando métricas de rendimiento para seleccionar el modelo óptimo en cada etapa.
- Diseñar, construir e implementar el prototipo ROCOBOT, integrando la infraestructura mecánica, sensores, servomotores, cámara y procesamiento embebido en Raspberry Pi 4B.
- Desarrollar un aplicativo web para la supervisión remota y monitoreo en tiempo real del proceso de clasificación automática del rocoto.
- Validar mediante pruebas piloto con rocotos (Capsicum pubescens) el funcionamiento integrado del sistema robótico ROCOBOT, evaluando el rendimiento del pipeline completo de detección y clasificación automática en condiciones operativas reales.

1.4. Justificación de la investigación

El presente proyecto se justifica en tres dimensiones complementarias. En primer lugar, desde una base tecnológica, al emplear arquitecturas de Deep Learning basadas en redes neuronales convolucionales (CNN) para la detección y clasificación automática del estado del

rocoto, diferenciando entre frutos saludables y defectuosos con alta precisión. En segundo lugar, desde una perspectiva de innovación aplicada, ya que el sistema representa una solución novedosa para la industria agroalimentaria, al automatizar el proceso de clasificación mediante una faja transportadora con rodillos y servomotores de clasificación física, reduciendo la dependencia de la inspección manual y minimizando errores humanos, lo que optimiza la calidad del producto final y reduce pérdidas por descartes inexactos. Finalmente, desde un enfoque de prototipado y validación, el proyecto desarrolla, implementa y valida el sistema robótico ROCOBOT en condiciones controladas, evaluando su rendimiento en términos de precisión, eficiencia y reducción del desperdicio, lo que garantiza su viabilidad como solución efectiva para el control de calidad en la industria del rocoto.

1.5. Estructura del informe

El presente informe se organiza en cuatro capítulos. El Capítulo I presenta el marco lógico de la investigación, incluyendo la síntesis de la situación problemática, la formulación del problema, los objetivos general y específicos, y la justificación de la investigación. El Capítulo II desarrolla el marco teórico, abarcando los antecedentes de la investigación, las bases teóricas relacionadas con el rocoto, la inteligencia artificial, la robótica y el desarrollo de software, y la operacionalización de las variables del estudio. El Capítulo III describe el marco metodológico empleado, incluyendo el análisis sistémico de la problemática, la metodología CRISP-DM, la metodología de trabajo Kanban, la población y muestra, y las técnicas e instrumentos utilizados. El Capítulo IV presenta los resultados obtenidos, correspondientes a la construcción de los datasets, el entrenamiento y comparación de los modelos de detección y clasificación, el diseño e implementación del prototipo físico, el desarrollo del aplicativo web y la validación mediante pruebas piloto del sistema. Finalmente, se presentan la discusión de resultados, las conclusiones, las recomendaciones, las referencias bibliográficas y los anexos correspondientes.

2. DISEÑO TEÓRICO

2.1. Antecedentes de la investigación

Kamboj y Diwan (2019) desarrollaron un sistema automático de clasificación implementado sobre una faja transportadora**,** controlado mediante un PLC con interfaz HMI para la supervisión y operación del prototipo. El sistema emplea una banda transportadora accionada por motor**,** sensores industriales para la detección de los objetos y una interfaz gráfica de usuario que permite controlar el funcionamiento de la faja, visualizar el estado del proceso y gestionar la clasificación en tiempo real. La investigación demuestra que el uso de interfaces de control integradas a prototipos con faja transportadora mejora la operación, monitoreo y confiabilidad de los sistemas de clasificación automatizada en entornos industriales

Hespeler et al. (2021) desarrollaron un modelo que aumenta el rendimiento de la detección en tiempo real y demuestra la posibilidad de ser usado durante la recolección robótica en tiempo real, su inspección y cosecha. Utilizaron imágenes térmicas en chiles y RGB en un entorno con diversas cantidades de escombros, superposición de pimientos e iluminación ambiental. Haciendo uso del conjunto de datos de ImageNet, donde se dividió en 15 clases y cada una constaba de 1000 imágenes, propusieron modelos de detección algoritmos avanzados de aprendizaje profundo, redes neuronales convolucionales regionales de máscara (Mask-RCNN) y You Only Look Once versión 3 (YOLOv3). Como resultado obtuvieron el mejor al utilizarse YOLOv3 donde alcanza un resultado del 100% tanto en el entrenamiento como en las pruebas.

Hyeon et al. (2021) en este estudio proponen un método para diagnosticar las enfermedades y plagas del pimiento picante, generaron imágenes similares a la imagen en consulta utilizando el algoritmo KNN un total de 23.868 imágenes de 19 tipos de enfermedades y plagas. Utilizaron Arquitecturas VGG y ResNet 50 con imágenes previamente entrenadas, bandas en un conjunto de datos de ImageNet, en donde podemos ver que los modelos propuestos logran precisiones del 96.02% y 99.61% respectivamente. También realizaron medidas de ajuste fino y las métricas de distancia, en donde las características en el ajuste fino muestran una precisión del 0.7%-7.38% y la distancia Bray-Courtis lora una precisión aproximada de 0.65% - 1.51% mayor que la distancia euclidiana. Además, al comparar tanto el rendimiento del modelo propuesto y el de clasificación obtuvieron una precisión del 97.87% y 97.88% respectivamente.

Ismail y Malik (2022) desarrollaron un sistema de inspección visual automatizada para la clasificación de frutas saludables y defectuosas, basado en aprendizaje profundo. El estudio utilizó 8 791 imágenes de manzanas y 300 imágenes de plátanos, aplicando técnicas de preprocesamiento y aumento de datos. Se evaluaron arquitecturas ResNet, DenseNet, MobileNetV2, NASNet y EfficientNet, destacando EfficientNet con una precisión del 99.2% en manzanas y 98.6% en plátanos en pruebas offline. Para la validación en tiempo real se implementó un prototipo físico de bajo costo, compuesto por Raspberry Pi, cámara digital, pantalla táctil y caja de iluminación, alcanzando precisiones del 96.7% y 93.8%, respectivamente, demostrando la viabilidad de sistemas embebidos para clasificación automatizada de frutas.

Li et al. (2022) en su proyecto presentaron un sistema robótico para la recolección de bayas maduras en racimos. Plantearon el objetivo de mejorar el proceso de recolección de frutas, específicamente para frutas que crecen en racimos o están rodeadas de obstáculos. La metodología combina dos redes neuronales que son el sistema de visión combina YOLOv4 para la detección de objetos y Deep SORT para el seguimiento de fresas maduras. Por otro lado, los componentes que utilizaron fueron una plataforma en forma de U y dos sistemas de recolección independientes montados a cada lado del arco, una cámara RGB-D, un brazo tipo SCARA, un gripper con sensores, un bucle de control visual y un panel LED. Los resultados del estudio muestran que el método propuesto fue más preciso en la separación de obstáculos sin reducir la velocidad, aumentando la tasa de éxito del proceso completo al 62,4% en racimos en el cultivador de fresas.

Ouyang et al. (2022) desarrollaron un sistema automático de monitoreo y clasificación de enfermedades del tomate basado en un robot móvil modular, integrando visión por computadora y aprendizaje profundo. El sistema evaluó modelos de detección YOLOv5, YOLOv7, Faster R-CNN y RetinaNet, utilizando 2 589 imágenes en diferentes condiciones, donde YOLOv5 obtuvo un mAP@0.5 de 90.4% y un recall de 85.2%. Para la etapa de clasificación se emplearon ResNet, MobileNet y DenseNet sobre un conjunto de 5 065 imágenes, destacando MobileNetV2 con una precisión del 96.7%. El sistema final combinó YOLOv5 y MobileNetV2, reduciendo falsos positivos y negativos, e incorporó una interfaz gráfica para la supervisión y control del prototipo robótico, validando su aplicación en entornos agrícolas automatizados.

Wang et al., (2022) aborda los desafíos de detectar la madurez de tomates en escenarios complejos, como la oclusión de ramas, solapamiento de frutos y variaciones en la iluminación, factores que limitan la precisión de métodos tradicionales y basados en inteligencia artificial.

Para resolverlo, propusieron MatDet, un modelo mejorado de Faster R-CNN que emplea ResNet-50 como backbone para una extracción robusta de características, RoIAlign para obtener cajas delimitadoras precisas, y una Red de Agregación de Rutas (PANet) para integrar características multi-escala. Usando un dataset diseñado para ambientes naturales, MatDet alcanzó una precisión promedio (mAP) de 96.14%, superando a otros modelos. Este enfoque asegura detecciones precisas y confiables en condiciones desafiantes, promoviendo la eficiencia en la recolección automatizada de tomates. El estudio resalta la superioridad del modelo al manejar oclusión, solapamiento y variaciones lumínicas, avanzando hacia sistemas agrícolas inteligentes.

Agegnehu et al. (2023) propusieron una red neuronal concatenada de las características extraídas de las redes VGG16 y AlexNet para desarrollar una clasificación de las enfermedades del pimiento, utilizando técnicas de procesamiento de imágenes y aprendizaje profundo que incluyen recopilación de datos, eliminación de ruido, segmentación, extracción y clasificación de características. El conjunto de datos estuvo conformado inicialmente por 1,596 imágenes, las cuales fueron ampliadas a 3,193 imágenes mediante técnicas de aumento de datos. Al evaluar el modelo, obtuvo una precisión de entrenamiento del 100%, una precisión de validación del 97.29% y una precisión en pruebas del 95.82%.

Dairath et al. (2023) desarrollaron un prototipo robótico integral de recolección y clasificación (picking cum grading) de mangos, basado en técnicas clásicas de visión por computadora (sin redes neuronales). El sistema captura las imágenes mediante una cámara RGB USB (Kodak T130, CMOS, 2 megapíxeles) conectada a una laptop, donde se ejecuta el esquema de procesamiento en Python (Anaconda Spyder IDE): conversión de espacio de color BGR a HSV, segmentación por umbral de matiz (hue), sustracción de fondo, dilatación morfológica y operación bitwise, calculando finalmente el porcentaje de área defectuosa mediante la razón entre píxeles de zona dañada y zona sana del fruto. El tiempo total del esquema de visión fue de 0.0125 s. La información de calidad se comunica a un microcontrolador Arduino UNO, que controla un brazo robótico de succión de 4 grados de libertad (4-DOF) con cuatro servomotores, el cual recoge únicamente los mangos de buena calidad y los deposita en un contenedor distinto; los de mala calidad continúan por la faja sin intervención. El sistema completo tomó un tiempo promedio de ciclo de 15 s para frutos de mala calidad (sin acción del brazo) y 21 s para frutos de buena calidad (con recolección activa mediante succión).

Dhanush et al. (2023) presentan en este artículo una revisión sistemática de los sistemas de visión artificial y los algoritmos de inteligencia artificial para detectar y cosechar productos

agrícolas. Los tipos de datos de imágenes más comunes utilizados en la automatización de la recolección son imágenes en blanco y negro (escala de grises), combina el blanco y el negro en n espectro continuo para crear una amplia variedad de tonos grises. Imágenes en color (RGB: rojo, verde, azul), imágenes RGB-D (rojo, verde, azul, profundidad), imágenes espectrales e imágenes térmicas. Se utilizaron técnicas de visión artificial para detectar y cosechar productos agrícolas así mismo también, técnicas en 2d para recolectar lo bidimensional e información en 3D para localizar el fruto o cultivo, así como técnicas de recopilación de datos, procesamiento de imágenes y Deep Learning, algoritmos usados para la detección y recolección de productos agrícolas. Entre los algoritmos de detección de una sola etapa utilizados en la recolección YOLOv3 y YOLOv4 tienen el promedio de precisión más alto y con respecto a los algoritmos usados para el modelo el que demuestra ser superior es MASK-RCNN tiene una precisión promedio (mAP) 89,9%.

Fan et al. (2023) En este estudio Construyeron un modelo de detección de malezas basado en el aprendizaje profundo en la etapa de plántula del campo de algodón, emplearon un modelo CBAM, una estructura BiFPN y el algoritmo de interpolación bilineal, este método propuesto alcanza un 98,43% de precisión, con una velocidad de interferencia mayor a la Faster R-CNN, aquí se emplearon 4694 imágenes representativas de algodón. Se implementó también un robot de pulverización que desarrollaron ellos mismos y al realizar las pruebas de campo dio un rendimiento con mAP del 97,42%, y una tasa de pulverización efectiva de 98,93%. El robot pulverizador es el portador del movimiento que se encarga de detectar las malezas y aplicar el pulverizador, incluye una plataforma móvil, suministro de energía, de comunicación remota, percepción visual, subsistema de movimiento, subsistema de pulverización, un controlador y un módulo de interacción hombre-computadora.

Sanwar y Ahmed (2023) desarrollaron un sistema automatizado de clasificación de objetos que integra procesamiento de imágenes en tiempo real, una faja transportadora con rodillos y un mecanismo robótico de tipo pick-and-place. La faja fue diseñada en SOLIDWORKS y fabricada con estructura de aluminio, tubos de rodillos, rodamientos, banda transportadora y un motor DC, permitiendo el desplazamiento continuo de los objetos hasta el área de captura visual. La adquisición de imágenes se realizó mediante una cámara web, y el procesamiento se ejecutó en un computador utilizando Python y OpenCV, aplicando técnicas clásicas de visión artificial para la detección de color y forma, sin emplear modelos de aprendizaje profundo ni datasets previamente entrenados. El sistema fue controlado por un microcontrolador Arduino Uno, encargado de accionar los servomotores del gripper robótico, logrando una clasificación correcta de los objetos según su forma y color en diferentes

posiciones angulares, demostrando la viabilidad de utilizar prototipos con faja transportadora y rodillos para procesos de clasificación automatizada.

Wang et al. (2023) Desarrollaron un prototipo de robot de envasado de melocotón basado en aprendizaje profundo, sin embargo, para facilitar el entrenamiento del modelo de detección del melocotón se recopilaron 430 imágenes RBG experimentales, posteriormente para el entrenamiento y validación trabajaron con 380 y 95 imágenes respectivamente. Se desarrolló también la detección de objetos melocotón y se entrenaron modelos YOLOv5 con diferentes anchos y profundidades de un extremo a otro, también se hizo uso de la calibración ocular manual "Eye-on Base" para resolver la matriz de transformación de coordenadas de la cámara, sistema de coordenadas base de un robot. Con respecto al éxito del agarre de los melocotones de acuerdo a su tamaño pequeño, mediano, grande fue del 100%, 97%, 97% respectivamente, todo el proceso propuesto en este estudio tomó un promedio de 252,81 ms para la percepción de profundidad. Finalmente, para la detección el modelo pudo alcanzar un 99,6%.

Bazame et al. (2023), enfocaron su investigación en la detección y clasificación de los estados de maduración de los frutos de café es crucial para optimizar la cosecha. Tradicionalmente, se utilizaban métodos manuales de extracción de características, pero el avance en visión por computadora ha permitido el uso de algoritmos de aprendizaje profundo como YOLO (You Only Look Once). En su estudio, implementaron YOLOv3 y YOLOv4 para clasificar frutos de Coffea arabica en inmaduros, maduros y sobremaduros, utilizando un conjunto de 387 imágenes RGB. Los resultados mostraron que YOLOv4 alcanzó una precisión media promedio (mAP) del 81%, superando a YOLOv3. A pesar de los desafíos en la detección de frutos inmaduros debido a la oclusión y similitud de color con el fondo, YOLOv4 demostró ser más robusto en condiciones de alta densidad de objetos, destacando el potencial de estas tecnologías en la agricultura de precisión.

Ibrahim et al. (2023) desarrollaron un sistema para la detección y clasificación automática de frutos de chile utilizando el algoritmo YOLOv5, con el objetivo de apoyar los procesos de cosecha y clasificación en robots agrícolas. Emplearon un dataset de 300 imágenes con resolución de 640 × 640 píxeles, dividido en 270 imágenes para entrenamiento y 30 para prueba. Las imágenes fueron etiquetadas mediante *bounding boxes* en formato YOLO y clasificadas en tres categorías: chile verde, chile rojo y chile podrido. El modelo alcanzó un mAP de 93.7 %, una precisión promedio de 88.0 % y un recall de 90.6 %, demostrando la viabilidad de YOLOv5 para la detección de frutos de chile en aplicaciones agrícolas. El estudio no reporta la aplicación de técnicas de *data augmentation*, indicando únicamente que las

imágenes fueron capturadas desde diferentes ángulos y distancias para incrementar la variabilidad del conjunto de datos.

Mohi-Alden et al. (2023) tuvo como objetivo desarrollar un sistema basado en visión artificial, con un enfoque de modelado inteligente para la clasificación en línea de pimientos morrones en muestras deseables e indeseables, con capacidad de predecir el nivel de madurez y el tamaño de los pimientos morrones deseables. Se capturaron imágenes de 640 pimientos frescos de un invernadero clasificados en tres niveles (verde, rojo completo e inmaduro) y diferentes tamaños. Las técnicas implementadas fueron 2; Redes Neuronales artificiales de perceptrón multicapa (MLP) y análisis discriminante lineal (LDA). Seleccionando la MLP por su precisión y desempeño. Así mismo se diseñó un sistema de clasificación con visión por computadora que incluía una cámara CCD industrial de alta resolución, un sistema de iluminación con lámparas LED ajustada para garantizar iluminación uniforme, un transporte PVC, para mover los pimientos bajo el sistema de visión, sensores magnéticos y eléctricos, para detectar y clasificar los objetos en línea. El modelo MLP clasificó los pimientos con un promedio de Precisión 93.2%, Sensibilidad 84% y Especificidad 95.7%, logrando procesar un aproximado de 3000 muestras por hora con un tiempo de clasificación de 0.2 segundos.

Moya et al. (2025) desarrollaron un sistema automatizado de clasificación postcosecha de tomates basado en YOLOv8, integrado a un prototipo físico con banda transportadora y servomotores. El modelo fue entrenado con un dataset de 2 145 imágenes de tomate**s**, capturadas bajo diferentes condiciones de iluminación, y permitió clasificar los frutos según su estado (verdes, rojos y dañados**)** y estimar su tamaño. El prototipo incorporó una cámara de visión artificial, sensores y un mecanismo de separación automática accionado por servomotores. Los resultados experimentales mostraron una accuracy del 99.6% en la clasificación**,** demostrando la alta eficiencia del uso de YOLOv8 en sistemas físicos de clasificación agrícola automatizada (Moya et al., 2025).

Abarquez et al. (2024) desarrollaron SiliAI, un sistema robótico inteligente para la clasificación automática de pimientos mediante visión por computadora. El sistema integró un modelo de detección de objetos YOLOv8 con un brazo robótico de seis grados de libertad, permitiendo identificar y clasificar los frutos en tres categorías: verde, rojo y podrido. Para el entrenamiento del modelo utilizaron un dataset de 5,000 imágenes de pimientos capturadas en diferentes condiciones. Los resultados evidenciaron un F1-score máximo de 93% y una precisión macro de 95.9% con un IoU de 0.5, alcanzando precisiones por clase de 99.5% para pimientos verdes, 97.0% para rojos y 91.3% para podridos. Los autores concluyeron que la

integración de YOLOv8 con un sistema robótico constituye una solución eficiente para automatizar el proceso de clasificación de pimientos en aplicaciones agrícolas.

Abdal et al. (2024) El documento describe un sistema automatizado que utiliza procesamiento de imágenes (Python + OpenCV) para clasificar objetos según su color y tamaño mediante una Raspberry Pi 4 Model B como controlador central. El hardware incluye una cinta transportadora accionada por motor DC, sensores infrarrojos ubicados en dos puntos del transportador para detectar la llegada del objeto, una cámara Raspberry Pi Sync 1080p Full HD de gran angular, un módulo controlador de motor L293D (puente H) para el motor DC, un servomotor para el mecanismo de separación física, y dos convertidores DC-DC tipo buck (LM2596) para regular el voltaje de alimentación del sistema. El algoritmo de clasificación aplica segmentación de color en el espacio HSV y detección de contornos (findContours de OpenCV) para determinar el área del objeto: si el área supera un umbral de 5,000 píxeles, el objeto se clasifica como apto y es dirigido por el servomotor; en caso contrario, se descarta como pequeño. El sistema se probó 100 veces y obtuvo una precisión del 89%, con errores atribuidos principalmente a variaciones en el brillo/iluminación ambiental durante la captura.

Khort et al. (2024) desarrollaron métodos sostenibles para la clasificación automatizada de manzanas mediante análisis hiperespectral y algoritmos de aprendizaje automático, con el objetivo de mejorar la calidad del producto y reducir pérdidas. El sistema integró un hiperespectrómetro BaySpec OCI-F, una cámara RGB y un sistema de transporte con rodillos motorizados, controlado mediante Arduino Micro (ATmega32u4) y motores paso a paso, permitiendo la adquisición de información espectral en el rango de 400–1000 nm. El dataset incluyó imágenes hiperespectrales de manzanas Honeycrisp, Gala y Jonagold, con lesiones confirmadas independientemente mediante juicio experto, PCR en tiempo real y fluorimetría 3D, atribuidas a patógenos como *Venturia inaequalis* (sarna), *Botrytis cinerea* y *Penicillium expansum* (pudrición). El estudio evaluó métodos de clasificación supervisada y no supervisada, destacando SVM como el algoritmo con mejor desempeño, alcanzando índices ARI de 0.789 (quemaduras solares), 0.818 (sarna) y 0.854 (pudrición), superando a K-Means. Adicionalmente, se determinó que una frecuencia de rotación de los rodillos inferior a 6 Hz mantiene el daño mecánico a las manzanas por debajo del 5%, demostrando que estas tecnologías son comparables al criterio de expertos humanos y contribuyen a la sostenibilidad y precisión en la clasificación de frutas.

Hou et al. (2024) Desarrollaron un sistema automático de clasificación de tomates que utiliza Raspberry PI 4B como plataforma de control para el brazo robótico, una cámara raspberry Pi, un brazo robótico, Placa CNC Arduino V3.0, hardware de cinta transportadora,

que clasifica a los tomates en función a su madurez, tamaño y grado de defecto y lo hace en 3 niveles: premium, primero y segundo. Emplearon YOLOv8 para analizar las imágenes de los tomates funciones avanzadas como pérdida CIoU y anclaje adaptativo para mejorar la precisión en la detección de contornos y clasificación. Y así mismo se prepararon y entrenaron un conjunto de datos con un total de 470 imágenes, 370 para entrenamiento, 50 para prueba y 50 en validación, ajustadas a un tamaño de 640x640. Los resultados experimentales indican que la precisión de clasificación obtuvo una tasa del 99.1% y una eficiencia de 1350 tomates por hora. Con un 100% en tamaño, 98.37% en madurez, y 99% en detección de áreas defectuosas.

Wang et al. (2024) propusieron un método para la detección y clasificación de fresas maduras vs no maduras empleando un modelo mejorado basado en YOLOv8+ con mecanismos de atención y procesamiento adicional para extraer características discriminativas de las imágenes. El modelo fue entrenado con un dataset de imágenes de fresa**s** capturadas bajo distintas condiciones, y los resultados demostraron que el sistema alcanzó una accuracy de 97.81 %, un recall de 96.36 % y un F1-score de 97.07 %, evidenciando la capacidad del enfoque para clasificar con alta precisión el estado de madurez de las fresas en un entorno controlado de instalación agrícola, lo que es relevante para robots de cosecha selectiva y sistemas automatizados de clasificación de frutos (Wang et al., 2024).

Noordin et al. (2025) desarrollaron un dispositivo de borde (edge device) de bajo costo para la detección de madurez de racimos de fruto de palma (Fresh Fruit Bunch, FFB) en tiempo real, orientado a la agricultura de precisión. El sistema emplea el modelo YOLOv8n, adaptado para ejecutarse sobre una Raspberry Pi 4 mediante técnicas de optimización como cuantización y aceleración por hardware, habilitado además con conectividad IoT para la toma de decisiones a partir de datos en vivo. Se utilizó un dataset de 4,194 imágenes anotadas de racimos de palma, con una división de 3,681 imágenes de entrenamiento, 348 de validación y 165 de prueba. Los resultados mostraron una precisión promedio (average precision) de 0.898 y un mAP de 0.952, demostrando la viabilidad de YOLOv8n en hardware embebido de bajo costo para el monitoreo agrícola en tiempo real.

Oliullah et al. (2025) El estudio propone un sistema innovador llamado FruVeg_MultiNet, que combina tecnologías IoT y un modelo híbrido de aprendizaje profundo para la identificación de frutas y vegetales frescos o en estado de descomposición. El sistema integra modelos de clasificación (EfficientNetB7 y ResNet50), desarrollando un modelo híbrido para optimizar la precisión y eficiencia. Se emplearon 2 conjuntos principales de datos Dataset 1: 14 clases (frescas y podridas) con un total de 16022(11215 entrenamiento, 2403 validación y 2404 prueba) y el Dataset 2: 20 clases con un total de 12000 imágenes distribuidas

uniformemente entre frutas y vegetales frescos y podridos. Se utilizó un modelo híbrido entre EfficientNetB7 y ResNet50 para una clasificación dando 99.92% en el primer conjunto de datos y del 95.93% en el segundo. Evaluaciones indicaron que modelo híbrido superó a otros (VGG16, VGG19, ResNet50) en métricas clave.

Tan et al. (2025) desarrollaron un sistema automatizado de clasificación y separación de manzanas mediante un brazo robótico, utilizando un modelo de segmentación YOLOv8 desplegado sobre una Raspberry Pi 4 Model B para clasificar las manzanas en tiempo real como buenas o defectuosas. Para las manzanas clasificadas como buenas, aplicaron un análisis adicional con OpenCV sobre la relación de color rojo y las dimensiones del fruto (altura y ancho), determinando así su grado como premium, estándar o rechazado. El hardware incluyó una PiCamera para la captura de imágenes y servomotores tanto para la rotación de la manzana como para su clasificación física, soportando dos modos de detección: de un solo lado y de múltiples lados (mediante rotación de la plataforma). Adicionalmente, desarrollaron una interfaz gráfica de usuario (GUI) en Tkinter para controlar los modos de detección, visualizar los resultados de clasificación y monitorear las acciones de separación en tiempo real. El sistema YOLOv8 alcanzó un mAP de 96.56%, y las pruebas de hardware demostraron una mejora en la precisión de clasificación del 97% (detección de un solo lado) al 100% (detección de múltiples lados) sobre 60 muestras de manzanas.

2.2. Bases teóricas

2.2.1. Bases teóricas del Rocoto

2.2.1.1. Descripción botánica

El rocoto (*Capsicum pubescens* R. & P.) es una especie originaria de Perú y Bolivia, considerada uno de los cultivos más representativos de la región andina. Se adapta principalmente a zonas ubicadas entre los 1700 y 2599 m s. n. m., donde las condiciones climáticas favorecen su desarrollo. Esta especie presenta una amplia diversidad de variedades que se distinguen por el color, tamaño y forma de sus frutos. Entre los colores más frecuentes se encuentran el rojo, amarillo, verde y anaranjado, características que influyen en sus propiedades organolépticas y en sus diferentes usos gastronómicos.

En el Perú predominan dos variedades: el rocoto serrano o de huerta y el rocoto de la selva central, siendo este último de mayor tamaño. Esta condición es respaldada por estudios de campo donde el ecotipo selva central registró el mayor peso unitario de

fruto (115.22 g) entre ocho ecotipos evaluados en la región Amazonas (Oliva et al., 2019). Asimismo, estudios experimentales realizados con las variedades Criollo de Oxapampa y Pucallpino reportaron diámetros promedio de fruto entre **7.63 y 8.37 cm**, sin diferencias estadísticamente significativas entre los tratamientos evaluados (Huacachino Zavala & Ripa Huacachino, 2023).

2.2.1.2. Producción y demanda

El Perú es reconocido a nivel mundial como el centro de origen y diversificación de los ajíes, el género capsicum es originario de regiones tropicales y subtropicales, actualmente tiene un valor cultural debido al amplio uso culinario, siendo su fruto empleado en la preparación de diversos platos representativos del Perú.

Según el Ministerio de Desarrollo Agrario y Riego (MIDAGRI, 2024), el Perú cuenta con más de 350 variedades de ajíes, rocotos y pimientos registradas en sus 24 departamentos. Entre las variedades más conocidas destacan el ají amarillo, limo, charapita, rocoto, panca, montaña y pipí de mono. Además, existen más de 11 mil productores agrarios que cultivan estos productos en unas 14 mil hectáreas distribuidas en todo el territorio nacional.Esta riqueza genética no solo posiciona al Perú como líder de biodiversidad, sino que también fortalece su potencial agroindustrial, ya que cada una de sus variedades nos ofrecen características únicas, aprovechadas por diversos mercados y procesos productivos.

Según Agro Perú (2023, septiembre 2) El rocoto al pertenecer al género capsicum, su concentración oscila entre 16.84-41.76 µg/g en la pulpa y 153.45 – 362.53 µg/g en la placenta/semilla. Resaltando que la capsaicina es la que provoca aquella sensación picante muy característica del rocoto mostrando al mismo tiempo importantes usos médicos, como el control del dolor , saturando los sensores y disminuyendo la secreción de ácido estomacales, que provocan úlceras gástricas, así como también disminuye la inflamación adiposa en hígado graso, e inclusive en algunos casos en particular, regula el sistema inmunológico bloqueando los canales de calcio que liberan células de forma irregular evitando así posibles cánceres. Además, cumple un rol fundamental ayudando a combatir la anemia, debido a que contiene hasta cuatro veces más vitamina C que la naranja, la cual favorece la absorción de hierro.

2.2.1.3. Uso en la Agroindustria

Los rocotos no solo se comercializan en fresco necesariamente, sino que también se procesan en diversas formas, como:

- Conservas (rocoto en vinagre o pasta de rocoto).
- Productos deshidratados y en polvo.
- Insumos para la industria gastronómica global.

El rocoto, en particular, tiene gran aceptación en la industria de salsas y encurtidos debido a su sabor, aroma y color. Productos de este tipo impulsan la agroexportación, diversificando los mercados internacionales del Perú.

2.2.1.4. Calidad del producto y aceptación en el mercado

El crecimiento de la demanda del rocoto está directamente vinculado a su calidad, tanto en términos genéticos como en las prácticas agronómicas empleadas. La mejora de la calidad no solo permite satisfacer el mercado interno, impulsado por la rica gastronomía peruana, sino que también fortalece su competitividad en mercados internacionales. El uso de presentaciones innovadoras y el cumplimiento de estándares fitosanitarios han permitido diversificar los destinos de exportación, siendo Estados Unidos un ejemplo destacado.

"En los cinco últimos años, el cultivo de rocoto ha incremento su área sembrada a nivel nacional, registrando 2587 ha en el 2017 e incrementándose a 4217 ha en el 2021" (Yapo & Pacheco, 2023). Esto evidencia una creciente demanda, tanto para consumo nacional como para exportación.

Según Yapo y Pacheco (2023, febrero) Las principales regiones productoras de rocoto y su participación en la producción total son: Pasco: 84.4% (56,310 toneladas), Junín: 6.4% (4273 toneladas), Puno: 2.35% (1571 toneladas), Cusco: 2.14% (1431 toneladas), Huánuco: 2.13% (1421 toneladas).

Según Gestión (2018, septiembre) La gran variedad de ajíes peruanos es muy valorada tanto en el mercado local como en el extranjero. Productos como el rocoto, caracterizado por su picor y sabor distintivo, son ingredientes esenciales en la gastronomía peruana, la cual ha ganado prestigio a nivel internacional. Este reconocimiento ha incrementado la demanda de ajíes peruanos en mercados como Estados Unidos, España y Japón.

La preferencia global por productos naturales y con beneficios nutricionales incrementa su demanda, especialmente por su alto contenido de vitamina C y

capsaicina. Esto refuerza el rol del rocoto como un producto esencial en la agroindustria peruana, integrando aspectos culturales, económicos y comerciales.

Figura 1:

Mapa de distribución geográfica del rocoto en el Perú por región



Nota: Obtenido de Yapo & Pacheco (2023).

2.2.1.5. **Clasificación y limitaciones**

La clasificación de alimentos es un proceso esencial en la industria agroalimentaria, ya que permite garantizar la calidad y homogeneidad de los productos antes de su comercialización. Tradicionalmente, este proceso se realizaba de manera manual, lo que conllevaba limitaciones en eficiencia, consistencia y velocidad. A pesar de la experiencia de los operarios, la inspección manual presenta desventajas como la subjetividad en la evaluación, la baja consistencia entre evaluadores, la dependencia de la mano de obra y la disminución del rendimiento debido a la fatiga humana, lo que afecta la precisión y confiabilidad del proceso (Patel et al., 2011). La identificación de frutos defectuosos, como aquellos con cicatrices, mordeduras o signos de putrefacción, es crucial para evitar su inclusión en la categoría de frutos saludables. Sin embargo, factores como la oclusión, la superposición de los frutos y la iluminación desigual dificultan una clasificación precisa. Para abordar estas limitaciones, la industria incorporó sistemas de clasificación mecánica que permitían separar los frutos según

atributos físicos como el tamaño y el peso. No obstante, estos sistemas no eran capaces de evaluar características de calidad más complejas, como el color, la forma o la presencia de defectos superficiales, limitando la precisión del proceso de clasificación (Patel et al., 2011).

Para superar estas limitaciones, el desarrollo de la visión por computadora permitió implementar sistemas de detección no destructivos y altamente precisos. En la actualidad, los sistemas basados en visión artificial se han implementado en diversas aplicaciones de la industria agroalimentaria, incluyendo la clasificación de frutas y hortalizas como pimientos (Abarquez et al., 2024), tomates (Hou et al., 2024), entre otros.

El uso de visión por computadora en la clasificación de tomates tiene antecedentes desde 1985, cuando Sarkar y Wolfe propusieron un sistema que permitía evaluar los frutos según su tamaño, color, forma y defectos. Sin embargo, este sistema requería que los tomates estuvieran orientados sobre su extremo del tallo o la base floral durante la captura de imágenes para la extracción de características. Años después, Clement et al. (2012) desarrollaron un clasificador de tomates basado en color, tamaño y peso, mejorando significativamente la eficiencia del proceso.

Una de las soluciones más prometedoras en la clasificación de alimentos es el uso de tecnologías avanzadas de visión artificial combinadas con algoritmos de aprendizaje automático. Estas tecnologías permiten una evaluación más precisa y rápida de parámetros como el color, la madurez, la presencia de defectos y la forma de los productos. En la clasificación de alimentos como el tomate, el pimentón y otras hortalizas, se han desarrollado sistemas que emplean cámaras multiespectrales, inteligencia artificial y técnicas de procesamiento de imágenes para lograr resultados más eficientes y automatizados (Patel et al., 2011).

En conclusión, la evolución de los sistemas de clasificación de alimentos ha pasado de métodos manuales a soluciones altamente automatizadas basadas en visión artificial. Estas innovaciones no solo mejoran la calidad del producto final, sino que también optimizan los tiempos de procesamiento y reducen el margen de error humano en la selección de los productos agroalimentarios.

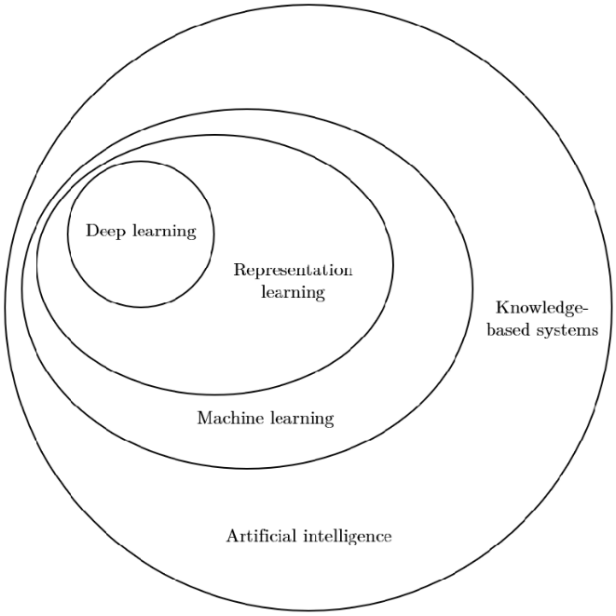
2.2.2. Bases Teóricas de Inteligencia Artificial

2.2.2.1. Inteligencia Artificial

Según Russell y Norvig (2021), la inteligencia artificial comprende un conjunto de subdisciplinas orientadas al desarrollo de sistemas capaces de simular comportamientos inteligentes. Estas subáreas permiten abordar distintos tipos de problemas, desde el aprendizaje automático hasta la percepción visual y la toma de decisiones. Las principales subáreas de la inteligencia artificial se muestran en la Fig.2.

Figura 2:

*Subáreas de la Inteligencia Artificia*l



Nota: Adaptado de Russell y Norvig (2021).

2.2.2.2. Machine Learning

El Machine Learning (aprendizaje automático) es una subárea de la inteligencia artificial que se enfoca en el desarrollo de algoritmos capaces de aprender patrones a partir de datos, sin ser programados explícitamente para cada tarea específica. Estos algoritmos mejoran su desempeño a medida que se exponen a mayor cantidad de datos, permitiendo realizar predicciones o tomar decisiones de manera autónoma (Mitchell, 1997). Según Mitchell (1997), un programa de computadora aprende a partir de la experiencia E, respecto a una tarea T y una medida de desempeño P, si su desempeño en la tarea T, medido por P, mejora con la experiencia.

2.2.2.3. Métricas de Evaluación

La evaluación del rendimiento de los modelos de Deep Learning se realiza mediante métricas cuantitativas que permiten medir objetivamente la capacidad de un modelo para generalizar ante datos nuevos. En el ámbito de la visión artificial, estas métricas se diferencian según la tarea que el modelo resuelve: detección de objetos o clasificación de imágenes. Si bien ambas tareas comparten la estructura base de la matriz de confusión, difieren en la unidad de análisis y en el criterio que determina si una predicción es correcta (Padilla et al., 2021).

a. Matriz de confusión

La matriz de confusión es una herramienta fundamental para evaluar el rendimiento de modelos de aprendizaje profundo, ya que resume de forma estructurada los resultados de las predicciones comparándolas con los valores reales. Está compuesta por cuatro valores que representan los posibles resultados de una predicción binaria (Sathyanarayanan & Roopashri Tantri, 2024):

Tabla 1:

Matriz de confusión

		Valores actuales	
		Positivo	**Negativo**
Valores Predichos	**Positivo**	Verdadero Positivo (VP)	Verdadero Negativo (VN)
	Negativo	Falso Positivo (FP)	Falso Negativo (FN)

Nota: Adaptado de Sathyanarayanan & Roopashri Tantri (2024)

Donde:

- Verdadero Positivo (VP): el modelo predice correctamente la clase positiva.
- Verdadero Negativo (VN): el modelo predice correctamente la clase negativa.
- Falso Positivo (FP): el modelo predice la clase positiva cuando el valor real es negativo (error tipo I).
- Falso Negativo (FN): el modelo predice la clase negativa cuando el valor real es positivo (error tipo II).

Si bien la estructura de la matriz es la misma para ambas tareas, la unidad de análisis y el criterio para determinar si una predicción es correcta difieren según la tarea.

En la clasificación de imágenes, cada fila representa una imagen completa y una predicción es VP cuando la clase predicha coincide exactamente con la clase real. En la detección de objetos, cada fila representa una instancia u objeto individual dentro de una imagen, una sola imagen puede contener múltiples instancias, y una predicción se considera VP únicamente cuando la clase es correcta y además la caja delimitadora predicha supera el umbral de superposición con la caja real, criterio definido por la Intersección sobre la Unión (IoU). Adicionalmente, en detección el Verdadero Negativo (VN) es prácticamente nulo, dado que existen infinitas regiones de la imagen sin objeto que no se contabilizan (Padilla et al., 2021).

b. Métricas de clasificación de imágenes

Las métricas de clasificación se derivan directamente de los valores de la matriz de confusión y permiten cuantificar el desempeño de un modelo que asigna una etiqueta de clase a cada imagen de entrada. Las métricas empleadas en problemas de clasificación binaria son (Sokolova & Lapalme, 2009):

- **Exactitud (Accuracy):** Representa la proporción total de predicciones correctas del modelo respecto al total de instancias evaluadas:

$$Accuracy = \frac{(VP + VN)}{(VP + FP + FN + VN)}$$

- **Precisión (Precision):** Evalúa la fiabilidad de las predicciones positivas, indicando qué proporción de las predicciones positivas son realmente correctas:

$$Precision = \frac{VP}{(VP + FP)}$$

- Sensibilidad (Recall): Mide la capacidad del modelo para identificar correctamente todos los casos positivos reales, minimizando los falsos negativos:

$$Sensibilidad = \frac{VP}{(VP + FN)}$$

- Especificidad: Evalúa la capacidad del modelo para identificar correctamente los casos negativos reales:

$$Especificidad = \frac{VN}{(VN + FP)}$$

- **F1-Score:** Combina precisión y sensibilidad en un único valor armónico, siendo especialmente útil en problemas con clases desbalanceadas:

$$F1 - Score = \frac{2 \times (Precision \times Sensibilidad)}{(Precision + Sensibilidad)}$$

- **Área Bajo la Curva ROC (AUC):** El AUC (*Area Under the ROC Curve*) mide la capacidad discriminativa global del modelo evaluando la relación entre la tasa de verdaderos positivos y la tasa de falsos positivos en todos los umbrales de decisión posibles. Un valor de AUC = 1.0 indica un clasificador perfecto, mientras que AUC = 0.5 equivale a una clasificación aleatoria (Fawcett, 2006).

c. Métricas de detección de objetos

Los modelos de detección de objetos predicen simultáneamente la ubicación espacial y la clase de los objetos presentes en una imagen mediante cajas delimitadoras (*bounding boxes*). Dado que en esta tarea cada predicción incluye tanto una clase como una localización, se requiere un criterio adicional para determinar si una detección es correcta: la Intersección sobre la Unión (IoU). A partir de este criterio se derivan las métricas de evaluación propias de la detección de objetos (Everingham et al., 2010).

- **Intersección sobre la Unión (IoU):** La IoU cuantifica el grado de superposición entre la caja delimitadora predicha y la caja real (*ground truth*). Se calcula como el cociente entre el área de intersección y el área de unión de ambas cajas:

$$IoU = \frac{Área\ de\ intersección}{Área\ de\ unión}$$

Una instancia se clasifica como Verdadero Positivo (VP) cuando IoU ≥ 0.5 y la clase es correcta; como Falso Positivo (FP) cuando IoU < 0.5 o no existe objeto real correspondiente; y como Falso Negativo (FN) cuando un objeto real no es detectado por el modelo (Rezatofighi et al., 2019).

- **Precisión (Precision):** Mide la proporción de instancias detectadas que corresponden a objetos reales sobre el total de detecciones realizadas:

$$Precision = \frac{VP}{(VP + FP)}$$

- **Exhaustividad (Recall):** Mide la proporción de objetos reales que fueron correctamente detectados por el modelo:

$$Recall = \frac{VP}{(VP + FN)}$$

- **Precisión Media Promedio a IoU = 0.5 (mAP@0.5):** El mAP@0.5 corresponde al área bajo la curva Precisión-Recall evaluada con un umbral fijo de IoU = 0.5. Representa el rendimiento global del modelo en condiciones estándar de evaluación (Everingham et al., 2010):

$$mAP@0.5 = \frac{1}{N} \Sigma\, AP_i \quad con\, IoU \geq 0.5$$

Donde N es el número de clases y AP_i es la precisión media de la clase i.

- **Precisión Media Promedio extendida (mAP@0.5:0.95):** El mAP@0.5:0.95 es el promedio del mAP calculado en diez umbrales de IoU entre 0.50 y 0.95 con paso de 0.05. Esta métrica evalúa de forma más exigente la precisión de localización del modelo, penalizando las cajas delimitadoras imprecisas incluso cuando la clase predicha es correcta (Lin et al., 2014):

$$mAP@0.5{:}\,0.95 = \frac{1}{10} \Sigma\, mAP@IoU_t \quad para\, t \in \{0.50, \ldots, 0.95\}$$

2.2.2.4. Deep Learning

Es una rama del Machine Learning basada en redes neuronales profundas que permiten aprender representaciones complejas a partir de grandes volúmenes de datos. Este enfoque ha demostrado un alto desempeño en tareas de visión por computadora y clasificación de imágenes, ya que permite extraer automáticamente características relevantes sin intervención manual (LeCun, Bengio & Hinton, 2015).

2.2.2.5. Visión Artificial

Es un área de la inteligencia artificial que se encarga de permitir que los sistemas computacionales adquieran, procesen y analicen información visual proveniente de imágenes o videos, con el fin de interpretar el entorno y tomar decisiones de manera automática. Esta disciplina combina técnicas de procesamiento de imágenes, aprendizaje automático y modelos de reconocimiento de patrones para imitar la capacidad humana de percepción visual.

La visión artificial es ampliamente utilizada en aplicaciones como la detección y clasificación de objetos, reconocimiento de patrones e inspección visual automatizada, especialmente en entornos industriales y agroindustriales, donde contribuye a mejorar la eficiencia, precisión y consistencia de los procesos de control de calidad (Szeliski, 2010).

2.2.2.6. Red Neuronal Convolucional (CNN)

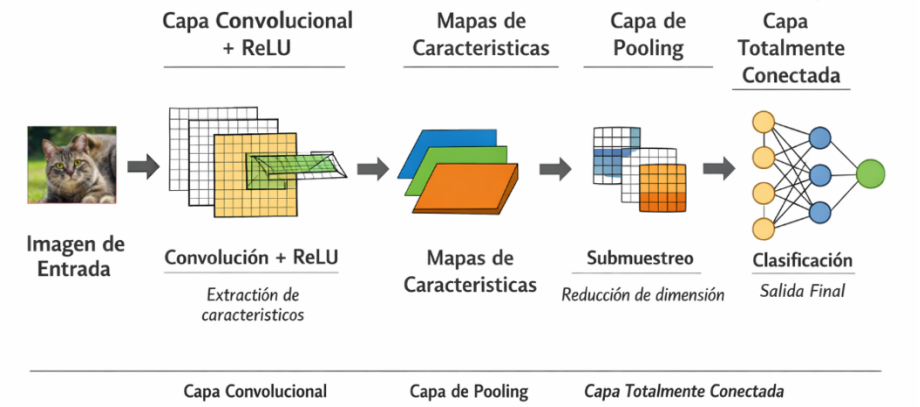
Son un tipo de red neuronal profunda especialmente diseñada para el procesamiento y análisis de imágenes. Estas redes permiten extraer automáticamente características visuales relevantes, como bordes, texturas, formas y patrones, reduciendo la necesidad de una extracción manual de características. Debido a esta capacidad, las CNN son ampliamente utilizadas en tareas de visión por computadora como detección, reconocimiento y clasificación de objetos (Goodfellow, Bengio & Courville, 2016). Una CNN está compuesta principalmente por las siguientes capas:

- Capa convolucional: Aplica filtros o kernels sobre la imagen de entrada para extraer características locales como bordes y contornos. A medida que la red profundiza, estas capas capturan patrones más complejos.
- **Capa de activación:** Introduce no linealidad al modelo, permitiendo aprender relaciones complejas. La función más utilizada es ReLU (Rectified Linear Unit).
- **Capa de pooling (submuestreo):** Reduce la dimensionalidad de los mapas de características, disminuyendo el costo computacional y ayudando a evitar el sobreajuste.
- **Capa totalmente conectada (Fully Connected)**: Integra las características extraídas por las capas anteriores para realizar la clasificación final.

Gracias a esta arquitectura jerárquica, las CNN han demostrado un alto desempeño en aplicaciones que requieren análisis visual preciso y eficiente.

Figura 3:

Arquitectura de Red Neuronal Convolucional (CNN)



Nota: Adaptado de Goodfellow, Bengio & Courville (2016)

2.2.2.7. You Only Look Once (YOLO)

Según Ultralytics (2025), YOLO aborda la detección de objetos como un problema de regresión, en el cual el modelo predice directamente las coordenadas de las cajas delimitadoras (bounding boxes), la probabilidad de pertenencia a cada clase y el nivel de confianza de cada detección. Este enfoque permite alcanzar altos niveles de velocidad y precisión, lo que lo hace ideal para aplicaciones en tiempo real como robótica, visión industrial y control de calidad automatizado. La arquitectura de los modelos YOLO se compone, de manera general, de tres bloques principales:

- **Backbone**: Se encarga de la extracción de características visuales de la imagen de entrada mediante capas convolucionales profundas. Este bloque aprende patrones como bordes, texturas y formas relevantes para la detección de objetos.
- **Neck:** Su función es fusionar características de diferentes escalas, permitiendo detectar objetos de distintos tamaños. Utiliza estructuras como *Feature Pyramid Networks (FPN)* y *Path Aggregation Networks (PAN)* para mejorar la representación espacial.
- **Head**: Realiza la predicción final, generando las coordenadas de las cajas delimitadoras, la probabilidad de clase y el nivel de confianza para cada objeto detectado.

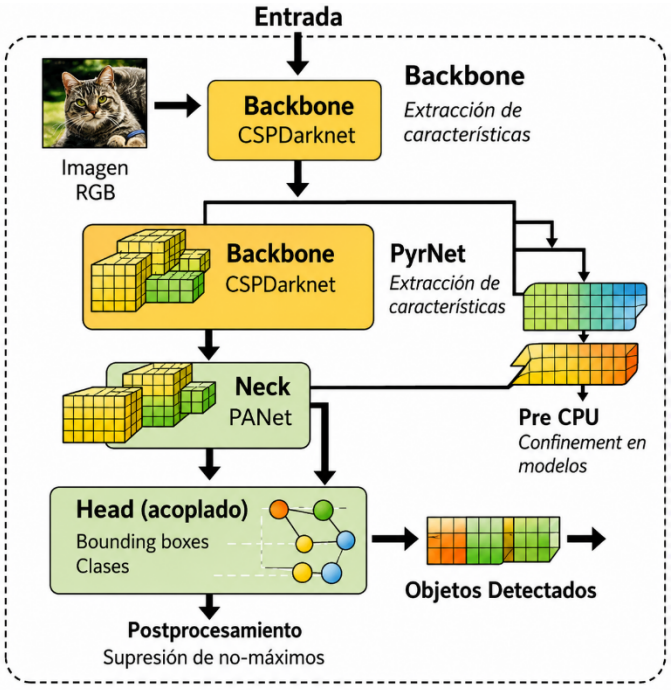
Esta arquitectura unificada permite que YOLO realice detección y clasificación de manera simultánea y eficiente.

a. YOLO V5

Es una de las versiones más utilizadas de la familia YOLO, destacándose por su estabilidad y eficiencia computacional. Su arquitectura está compuesta por un backbone CSPDarknet, un neck basado en PANet y un head acoplado; la predicción de clases y la localización de objetos se realizan de forma conjunta.

Figura 4:

Arquitecturas YOLOv5



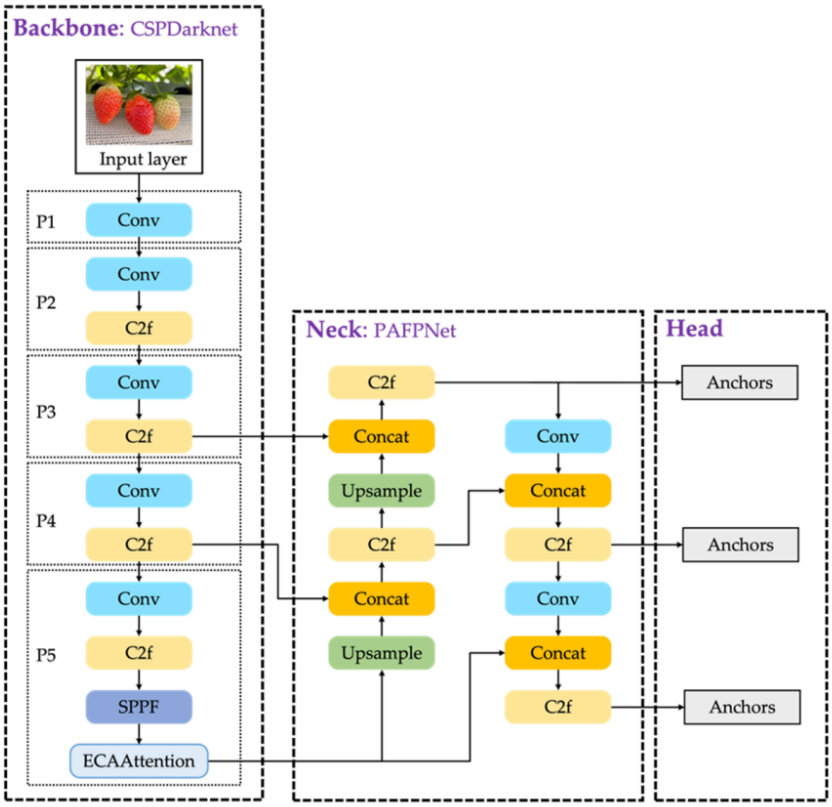
Nota: Adoptado de Ultralytics. (2025)

b. YOLO V8

Representa una evolución de versiones anteriores, incorporando mejoras significativas en su arquitectura interna. Entre sus principales innovaciones se encuentra el uso de un head desacoplado, donde la predicción de clases y la regresión de cajas delimitadoras se realizan de manera separada, lo que mejora la precisión y estabilidad del modelo durante el entrenamiento. Además, YOLOv8 presenta optimizaciones en el backbone y el neck, lo que permite una mejor detección de objetos pequeños y un rendimiento superior en aplicaciones en tiempo real.

Figura 5:

Arquitecturas YOLOv8



Nota: Obtenido de Wang, et al. (2024)

c. Yolo v11

Es la versión más reciente de la familia YOLO, introducida por Ultralytics (2025), que incorpora el módulo C3k2 para bloques de convolución más eficientes y el módulo de atención espacial paralela PSA (*Parallel Spatial Attention*), orientado a mejorar la captura de contexto global en la imagen sin incrementar significativamente el costo computacional respecto a versiones anteriores (Ultralytics, 2025).

2.2.2.8. Efficient Net

Es una familia de modelos de redes neuronales convolucionales profundas (CNN) diseñada para lograr un alto rendimiento en tareas de clasificación de imágenes, optimizando de manera equilibrada la precisión, el costo computacional y el uso de memoria.

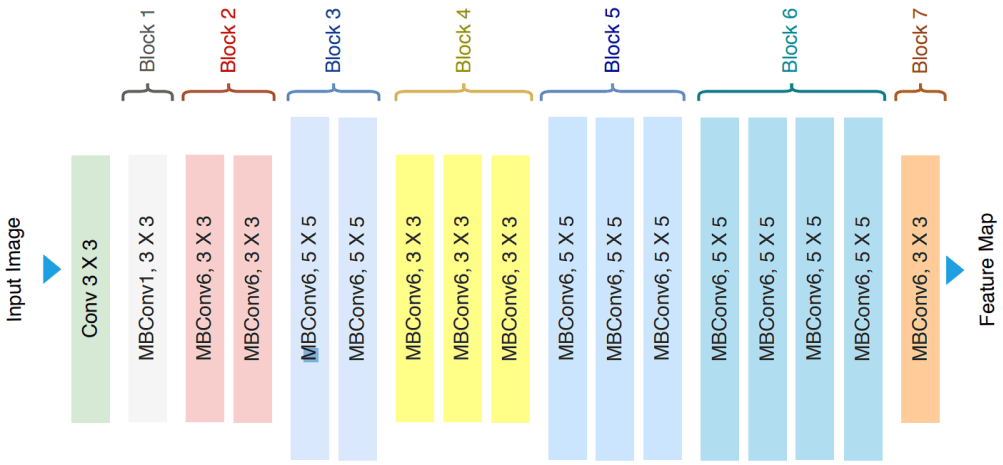
EfficientNet fue propuesto por Tan y Le (2019), quienes introdujeron un nuevo enfoque denominado escalado compuesto (*compound scaling*), el cual ajusta de forma conjunta y sistemática tres dimensiones fundamentales de una red neuronal:

profundidad, ancho y resolución de entrada. La arquitectura de EfficientNet se basa en bloques convolucionales eficientes llamados MBConv (Mobile Inverted Bottleneck Convolution), que incluyen:

- Convoluciones separables en profundidad (depthwise separable convolutions).
- Capas de normalización y activación.
- Mecanismo de atención Squeeze-and-Excitation (SE), que mejora la representación de características relevantes.
- A partir del modelo base EfficientNet-B0, se derivan versiones más profundas y complejas (B1–B7), manteniendo un equilibrio óptimo entre rendimiento y eficiencia.

Figura 6:

Arquitectura de EfficientNet



Nota: Obtenido de Tan y Le (2019).

EfficientNet ha demostrado un excelente desempeño en tareas de clasificación de imágenes, siendo especialmente adecuado para aplicaciones donde se requiere: Alta precisión en la clasificación, uso eficiente de recursos computacionales e implementación en sistemas embebidos o prototipos inteligentes.

En el contexto de esta investigación, EfficientNet se emplea para la clasificación del estado del rocoto, diferenciando entre frutos saludables y defectuosos a partir de imágenes capturadas por el sistema de visión artificial.

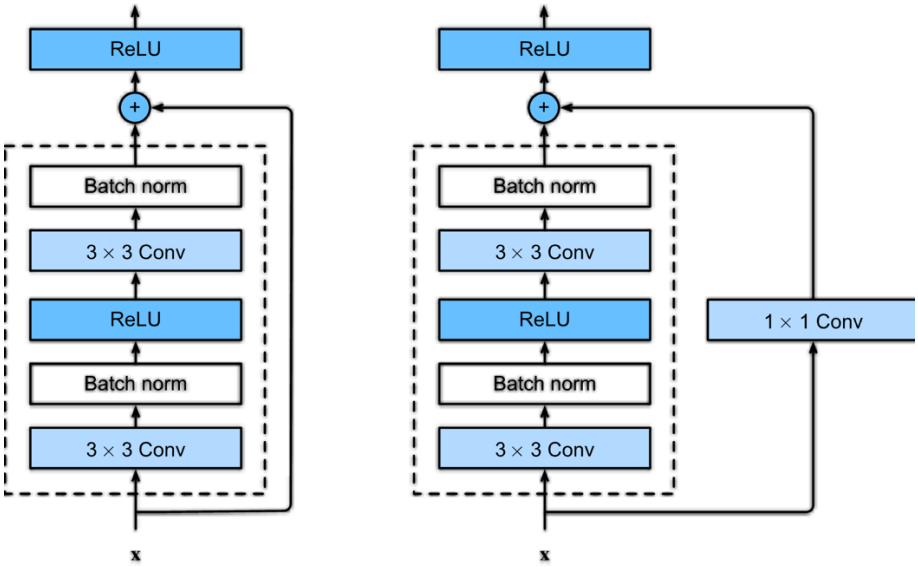
2.2.2.9. ResNet (Residual Network)

Las Redes Neuronales Residuales (ResNet) son una arquitectura de redes neuronales convolucionales profundas propuesta por Kaiming He et al. (2016), cuyo objetivo principal es facilitar el entrenamiento de redes muy profundas y mejorar el rendimiento en tareas de clasificación de imágenes.

Este permite que las capas de la red aprendan funciones residuales en lugar de transformaciones directas. Este enfoque resuelve el problema del desvanecimiento del gradiente, común en redes profundas, permitiendo entrenar modelos con decenas o cientos de capas sin degradar su desempeño (He et al., 2016).

Figura 7:

Bloque residual de la arquitectura ResNet



Nota: Adaptado de He et al., (2016)

2.2.2.10.OpenCV (Open-Source Computer Vision)

OpenCV es una herramienta de código abierto que permite que las computadoras puedan "ver" e interpretar imágenes y videos. Con esta biblioteca, podemos hacer cosas como detectar rostros, seguir objetos en movimiento, inclusive mejorar la calidad de una imagen. Es como darle a una computadora la capacidad de ver y entender lo que está mirando.

2.2.2.11.Tensor Flow

TensorFlow es una biblioteca de código abierto desarrollada por Google Brain para la creación y entrenamiento de modelos de aprendizaje automático, con un enfoque especial en el aprendizaje profundo. Su arquitectura basada en grafos computacionales permite la ejecución eficiente en CPU, GPU y TPU, optimizando el rendimiento en diversas aplicaciones, como visión artificial, procesamiento de lenguaje natural y análisis de datos (Tang et al., 2022).

Desde su lanzamiento, TensorFlow ha evolucionado con mejoras en rendimiento, compatibilidad y facilidad de uso. Actualmente, su ecosistema incluye TensorFlow Lite, diseñado para embebidos, y TensorFlow.js, que permite la ejecución de modelos en navegadores web (Biswas & Simon, 2023).

2.2.2.12.Keras

Keras es una biblioteca de aprendizaje profundo de alto nivel, diseñada para facilitar la construcción, entrenamiento y validación de modelos de redes neuronales. Proporciona una interfaz simple y modular que permite desarrollar modelos de manera rápida y eficiente, siendo ampliamente utilizada en aplicaciones de visión por computadora y aprendizaje automático. Actualmente, Keras se integra de forma nativa con TensorFlow, lo que permite aprovechar su capacidad de cómputo y escalabilidad (Chollet, 2015).

2.2.2.13.Lenguaje de Programación Python

Es un lenguaje de programación muy popular por su simplicidad y versatilidad. Actualmente se ha convertido en uno de los más demandados, se utiliza en una amplia gama de aplicaciones, desde el desarrollo web hasta la automatización de tareas y, especialmente, en el campo de la inteligencia artificial y el análisis de datos. Debido a su ecosistema de bibliotecas como NumPy, pandas, TensorFlow, y OpenCV que lo hacen ideal para trabajar en proyectos complejos, como el procesamiento de imágenes o la creación de modelos de aprendizaje automático.

2.2.3. Bases Teórica de robótica

2.2.3.1. Unidad de procesamiento

a. Raspberry Pi

La Raspberry Pi 4 Model B es una computadora de placa única desarrollada por la Raspberry Pi Foundation, diseñada para aplicaciones de computación embebida, automatización, robótica y sistemas inteligentes. Este dispositivo integra un procesador ARM de cuatro núcleos, memoria RAM configurable y múltiples interfaces de entrada y salida, lo que lo convierte en una plataforma adecuada para proyectos de visión artificial y control robótico.

Según la Raspberry Pi Foundation (2023), la Raspberry Pi 4 Model B incorpora un procesador Broadcom BCM2711 Quad-Core Cortex-A72 de 64 bits, puertos USB 3.0, HDMI dual con soporte para resolución 4K, conectividad inalámbrica (Wi-Fi y Bluetooth) y pines GPIO programables, permitiendo su integración con sensores, cámaras y actuadores en sistemas automatizados.

2.2.3.2. Sensores

a. Cámara Raspberry Pi

La Cámara Raspberry Pi es un módulo de captura de imágenes diseñado específicamente para integrarse con las plataformas Raspberry Pi a través del puerto CSI (Camera Serial Interface). A diferencia de las cámaras USB convencionales, este módulo ofrece mayor estabilidad de transferencia de datos y menor latencia, características relevantes para aplicaciones de visión artificial en tiempo real. Según Bradski y Kaehler (2008), la adquisición de imágenes mediante interfaces dedicadas constituye una solución práctica y efectiva para aplicaciones de visión por computadora en tiempo real.

b. Sensor infrarrojo FC-51

El sensor infrarrojo FC-51 es un sensor de detección de obstáculos que funciona mediante la emisión y recepción de luz infrarroja. En el sistema propuesto, este sensor permite detectar la presencia del rocoto en puntos estratégicos del proceso, facilitando la sincronización del transporte y la clasificación. De acuerdo con la documentación técnica del módulo, el FC-51 proporciona señales digitales confiables para aplicaciones de detección de objetos (ElectroPeak, 2019).

2.2.3.3. Actuadores

a. Motoreductor DC JGB37-520

El motoreductor JGB37-520 es un actuador electromecánico de corriente continua con caja reductora integrada, que combina un motor DC con un sistema de engranajes para aumentar el torque de salida y reducir la velocidad de rotación. Esta característica lo hace adecuado para aplicaciones donde se requiere un movimiento controlado y sostenido, como el accionamiento de fajas transportadoras en sistemas automatizados. Según Bolton (2015), los motoreductores DC son ampliamente utilizados en sistemas mecatrónicos debido a su simplicidad y eficiencia.

b. Servomotor MG995

El servomotor MG995 es un actuador de posicionamiento angular de alto torque que integra un motor DC, un sistema de engranajes metálicos y un circuito de control con realimentación de posición. Su capacidad de posicionamiento preciso mediante señales PWM lo convierte en una solución adecuada para mecanismos de clasificación y control de compuertas en sistemas automatizados. Según el fabricante TowerPro (2014), el MG995 está diseñado para aplicaciones que requieren movimientos repetitivos y alta fuerza mecánica.

c. Driver L298N

El módulo L298N es un controlador de motores de doble puente H utilizado para regular la velocidad y el sentido de giro de motores DC. En este proyecto, el L298N permite controlar el motoreductor de la faja corredora a partir de señales de bajo nivel generadas por la Raspberry Pi. De acuerdo con STMicroelectronics (2000), este módulo es adecuado para aplicaciones de control de motores en sistemas automatizados.

d. Convertidor DC-DC LM2596

El LM2596 es un convertidor DC-DC tipo reductor (buck) que regula una tensión de entrada variable a una tensión de salida estable y configurable. Su función es proporcionar una alimentación estable a los componentes electrónicos del sistema, protegiéndolos de variaciones en la tensión de la fuente principal. Según Texas Instruments (2016), el LM2596 es ampliamente utilizado en sistemas embebidos por su eficiencia energética y confiabilidad.

2.2.3.4. Componente Mecánicos e iluminación

a. Faja transportadora con rodillos

La faja transportadora con rodillos es un sistema mecánico de transporte continuo que emplea una banda flexible soportada por rodillos para desplazar objetos de forma controlada a lo largo de una trayectoria lineal. Su estructura ligera, fabricada en MDF de 4 mm, ofrece rigidez, facilidad de mecanizado y bajo costo, características adecuadas para el desarrollo de prototipos de automatización. De acuerdo con Kalpakjian y Schmid (2014), el MDF es un material ampliamente utilizado en la fabricación de prototipos mecánicos y estructuras ligeras.

b. Caja de cartón espuma

La caja de cartón espuma es una estructura de soporte ligera utilizada para alojar la cámara y el sistema de iluminación, aislando la zona de captura de imágenes de la luz ambiental y garantizando condiciones de iluminación controladas y reproducibles. Según Ulrich y Eppinger (2016), este tipo de material es adecuado para prototipos experimentales debido a su ligereza y facilidad de manipulación.

c. Tira de Luz

La tira de luz LED es un dispositivo de iluminación flexible compuesto por diodos emisores de luz (LED) distribuidos a lo largo de una banda continua, que proporciona una iluminación uniforme y dirigida sobre el objeto a inspeccionar. Su flexibilidad permite adaptarla a diferentes geometrías y posiciones dentro del sistema, reduciendo sombras y variaciones lumínicas que pueden afectar el desempeño de los algoritmos de visión artificial. Según Gonzalez y Woods (2018), una iluminación controlada es un factor determinante para mejorar la calidad de las imágenes y la precisión del procesamiento visual en sistemas de inspección automatizada.

2.2.4. Bases teóricas de desarrollo de software

2.2.4.1. Arquitectura de Software

La arquitectura de software define la estructura organizativa de un sistema, estableciendo cómo sus componentes se relacionan entre sí y con el entorno (Bass et al., 2021). Una arquitectura bien definida facilita la mantenibilidad, la escalabilidad y la testabilidad del sistema, reduciendo el costo de los cambios a lo largo del ciclo de vida del proyecto.

2.2.4.2. Clean Architecture

Clean Architecture es un patrón de arquitectura de software propuesto por Robert C. Martin (2017) que organiza el sistema en capas concéntricas con dependencias que apuntan únicamente hacia el interior, es decir, hacia las reglas de negocio. Las capas externas (frameworks, bases de datos, interfaces de usuario) son detalles de implementación intercambiables que no deben afectar la lógica central del sistema. Este patrón garantiza que el núcleo de negocio pueda probarse de forma independiente sin requerir la presencia de bases de datos, servidores web ni interfaces gráficas (Martin, 2017).

2.2.4.3. Monolito Modular

Un monolito modular es una arquitectura de software en la que todo el sistema se despliega como una única unidad ejecutable, pero su código interno se organiza en módulos con responsabilidades claramente delimitadas y bajo acoplamiento entre ellos. A diferencia de los microservicios, el monolito modular no requiere infraestructura de comunicación entre servicios, lo que reduce la complejidad operativa sin sacrificar la separación de responsabilidades; es especialmente adecuado para proyectos de mediana escala donde el equipo de desarrollo es pequeño y el volumen de tráfico no justifica la distribución (Fowler, 2015).

2.2.4.4. API REST

Una API REST (Representational State Transfer) es un estilo de arquitectura para el diseño de servicios web que utiliza el protocolo HTTP para la comunicación entre clientes y servidores mediante operaciones estándar (GET, POST, PUT, DELETE). Sus principios fundamentales incluyen la ausencia de estado en el servidor (stateless), la identificación de recursos mediante URLs y la transferencia de representaciones de datos en formato JSON (Fielding, 2000). Las APIs REST permiten que distintos clientes (navegadores web, aplicaciones móviles, dispositivos embebidos consuman los mismos servicios de forma independiente, favoreciendo la interoperabilidad y el desacoplamiento entre componentes.

2.2.4.5. Control de Acceso Basado en Roles (RBAC)

El modelo de control de acceso basado en roles (RBAC, Role-Based Access Control) es un mecanismo de seguridad en el cual los permisos no se asignan directamente a los usuarios individuales, sino a roles predefinidos que luego se asocian

a las cuentas. De este modo, la gestión de permisos se simplifica: modificar los privilegios de un rol afecta automáticamente a todos los usuarios que lo poseen, sin necesidad de reconfigurar cada cuenta de forma individual (Sandhu et al., 1996). RBAC es ampliamente adoptado en sistemas empresariales e industriales porque reduce el riesgo de errores de configuración y facilita el cumplimiento de políticas de seguridad.

2.2.4.6. FastAPI

FastAPI es un framework de Python de alto rendimiento para el desarrollo de APIs REST, diseñado sobre los estándares ASGI (Asynchronous Server Gateway Interface) y OpenAPI. Sus características principales incluyen soporte nativo de programación asíncrona mediante async/await, validación automática de datos mediante Pydantic y generación automática de documentación interactiva (Swagger UI y ReDoc). Según Ramírez (2021), FastAPI permite desarrollar APIs con una productividad comparable a frameworks de alto nivel como Django, manteniendo un rendimiento cercano a frameworks de bajo nivel como Node.js.

2.2.4.7. React

React es una biblioteca de JavaScript de código abierto utilizada para el desarrollo de interfaces de usuario mediante componentes reutilizables. Su arquitectura basada en componentes favorece la modularidad, la reutilización del código y el mantenimiento de aplicaciones web. Asimismo, emplea un DOM virtual (Virtual DOM), el cual optimiza el proceso de actualización de la interfaz al comparar los cambios realizados y modificar únicamente los elementos necesarios en el DOM real, mejorando así el rendimiento de aplicaciones con actualizaciones frecuentes. Además, React facilita el desarrollo de aplicaciones de tipo *Single Page Application* (SPA), en las que una única página HTML se actualiza dinámicamente mediante JavaScript sin necesidad de recargar completamente la página, proporcionando una experiencia de usuario más rápida y fluida (React, 2024).

2.2.4.8. PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacional de código abierto que implementa el estándar SQL con extensiones adicionales para manejo de tipos de datos complejos, integridad referencial, transacciones ACID y soporte para esquemas múltiples dentro de una misma base de datos (PostgreSQL Global Development Group, 2023). Su motor de consultas optimizado lo hace especialmente

eficiente para operaciones de agregación, reportes y consultas analíticas sobre grandes volúmenes de datos relacionales.

2.2.4.9. Redis

Redis (Remote Dictionary Server) es un almacén de datos en memoria de código abierto que funciona como base de datos, caché y broker de mensajes. Al mantener los datos en RAM, Redis ofrece tiempos de respuesta en el orden de los microsegundos, lo que lo hace ideal para casos de uso que requieren lecturas y escrituras de alta frecuencia y baja latencia, como sesiones de usuario, resultados temporales en tiempo real y contadores de eventos (Redis, 2023).

2.2.4.10. Integración y Despliegue Continuos (CI/CD)

La integración continua (CI) y el despliegue continuo (CD) constituyen un conjunto de prácticas de ingeniería de software orientadas a automatizar la compilación, las pruebas y el despliegue de nuevas versiones del sistema. En un pipeline CI/CD, cada cambio en el código fuente activa automáticamente la ejecución de pruebas unitarias, análisis estático del código y, si todos los controles pasan, el despliegue de la nueva versión en producción sin intervención manual (Humble y Farley, 2010). Esta práctica reduce el riesgo de errores en producción, acorta el ciclo de entrega y garantiza trazabilidad completa de cada versión desplegada.

2.2.4.11. Vercel

Vercel es una plataforma de despliegue en la nube especializada en aplicaciones frontend, especialmente aquellas basadas en frameworks JavaScript como React y Next.js. Ofrece una red de distribución de contenido (CDN) global que sirve los recursos estáticos desde el nodo geográfico más cercano al usuario, reduciendo los tiempos de carga. Se integra de forma nativa con repositorios de GitHub para activar despliegues automáticos ante cada cambio en el código fuente (Vercel, 2023).

2.2.4.12. Railway

Railway es una plataforma de infraestructura en la nube que permite desplegar aplicaciones backend, bases de datos y servicios de caché en un entorno unificado con comunicación por red privada interna entre componentes. Su modelo de precios basado en el consumo real de recursos lo hace especialmente adecuado para proyectos

académicos y de prototipado, donde el tráfico no es constante y mantener un servidor dedicado representaría un costo innecesario (Railway, 2023).

2.2.4.13.Esquemas de Base de Datos

Un esquema en PostgreSQL es un espacio de nombres lógico dentro de una base de datos que agrupa tablas, vistas, funciones y otros objetos relacionados bajo un mismo dominio funcional. El uso de múltiples esquemas dentro de una misma base de datos permite separar los módulos del sistema como autenticación, navegación y producción sin necesidad de crear bases de datos independientes, facilitando la gestión de permisos por dominio, mejorando la legibilidad del modelo de datos y reduciendo la complejidad de la infraestructura (PostgreSQL Global Development Group, 2023).

El diseño de esquemas sigue los principios del modelo entidad-relación (ER) y la normalización hasta la tercera forma normal (3FN), que establecen que cada tabla debe representar una única entidad, cada atributo debe depender directamente de la clave primaria y no deben existir dependencias transitivas entre atributos no clave (Codd, 1970). La aplicación de estos principios garantiza la integridad referencial entre las tablas, evita la redundancia de datos y facilita la consistencia del sistema ante operaciones de inserción, actualización y eliminación.

2.3. Operacionalización de variable

2.3.1. Definición conceptual

La variable de estudio es el sistema robótico inteligente para la detección de defectos y clasificación de calidad en rocotos (*Capsicum pubescens*).

2.3.2. Definición operativa

Esta variable se mide en tres dimensiones: rendimiento del modelo de detección, rendimiento del modelo de clasificación y rendimiento operativo del prototipo físico, detalladas en la Tabla 2.

Tabla 2:

Operacional de variables del proyecto

Variable	Dimensión	Indicadores	Medida
Sistema robótico ROCOBOT	Rendimiento del modelo de detección	Precisión	Porcentaje
		Recall	Porcentaje
		mAP@0.5	Porcentaje
		mAP@0.5:0.95	Porcentaje
	Rendimiento del modelo de clasificación	Exactitud	Porcentaje
		Precisión	Porcentaje
		Sensibilidad	Porcentaje
		Especificidad	Porcentaje
	Rendimiento operativo del prototipo	Tasa de clasificación correcta	Porcentaje
		Confianza promedio del modelo	Porcentaje
		Tiempo de procesamiento por rocoto	Milisegundos

Nota: Adaptado de Sathyanarayanan & Roopashri Tantri (2024).

3. DISEÑO METODOLÓGICO

3.1. Tipo de investigación

La presente investigación se clasifica, según su finalidad, como una investigación de tipo aplicada, dado que orienta el conocimiento científico y tecnológico existente en Deep Learning, visión artificial y robótica hacia la solución de un problema concreto del sector agroindustrial: la clasificación ineficiente y subjetiva de rocotos (*Capsicum pubescens*). Según su enfoque, corresponde a una investigación cuantitativa, en tanto los resultados se expresan mediante métricas numéricas de rendimiento (precisión, recall, mAP, exactitud, entre otras) que permiten comparar objetivamente el desempeño de los distintos modelos evaluados. Según su diseño, se trata de una investigación de tipo tecnológico-experimental, ya que involucra el diseño, desarrollo y evaluación controlada de un prototipo (ROCOBOT), así como el entrenamiento y comparación sistemática de múltiples arquitecturas de redes neuronales bajo las mismas condiciones de entrenamiento y evaluación, con el fin de determinar cuál ofrece el mejor desempeño para cada etapa del sistema. Esta clasificación es concordante con la pregunta de investigación planteada, la cual no busca establecer relaciones causales entre variables independientes y dependientes en un sentido experimental clásico, sino determinar la viabilidad técnica de una solución tecnológica ante un problema identificado.

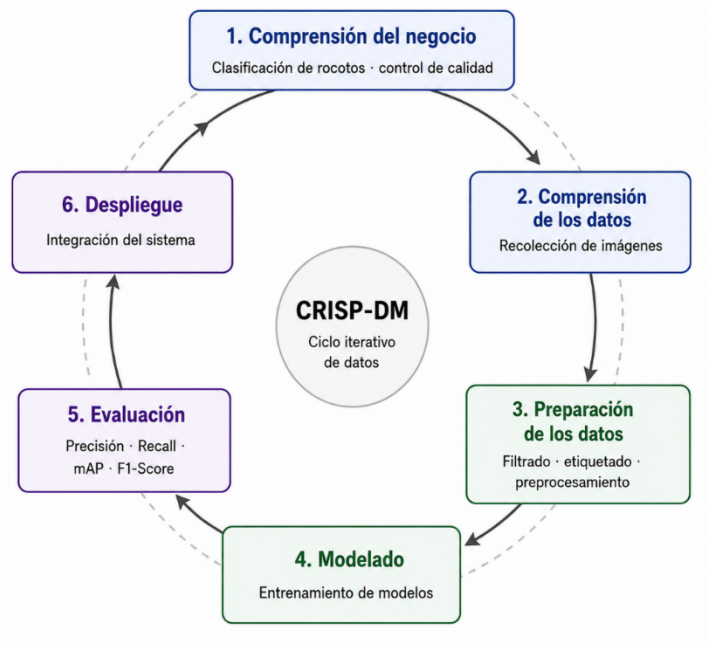
3.2. Métodología empleada: CRISP-DM

La metodología empleada en el desarrollo del presente proyecto fue CRISP-DM (Cross-Industry Standard Process for Data Mining), un proceso estándar de ciclo de vida ampliamente adoptado en proyectos de minería de datos, ciencia de datos e inteligencia artificial. De acuerdo con Chapman et al. (2000), CRISP-DM define un marco metodológico compuesto por seis fases interdependientes organizadas en un ciclo iterativo: (1) comprensión del negocio, (2) comprensión de los datos, (3) preparación de los datos, (4) modelado, (5) evaluación y (6) despliegue. Su carácter no lineal permitió retroceder a fases anteriores según los hallazgos obtenidos durante el proceso, lo que garantizó la mejora continua del modelo y la adaptación ante resultados inesperados. La selección de CRISP-DM como metodología del presente proyecto se fundamentó en que ROCOBOT es, en esencia, un sistema cuyo núcleo consistió en el entrenamiento, la evaluación y el despliegue de modelos de aprendizaje profundo orientados a la clasificación automática de rocotos. A diferencia de metodologías orientadas al desarrollo de software convencional como RUP o Scrum, CRISP-DM fue concebida específicamente para proyectos donde el artefacto central es un modelo de datos entrenado y

validado, característica que se correspondió directamente con los objetivos del presente trabajo de investigación.

Figura 8:

Metodología CRISP-DM aplicada al desarrollo del sistema ROCOBOT



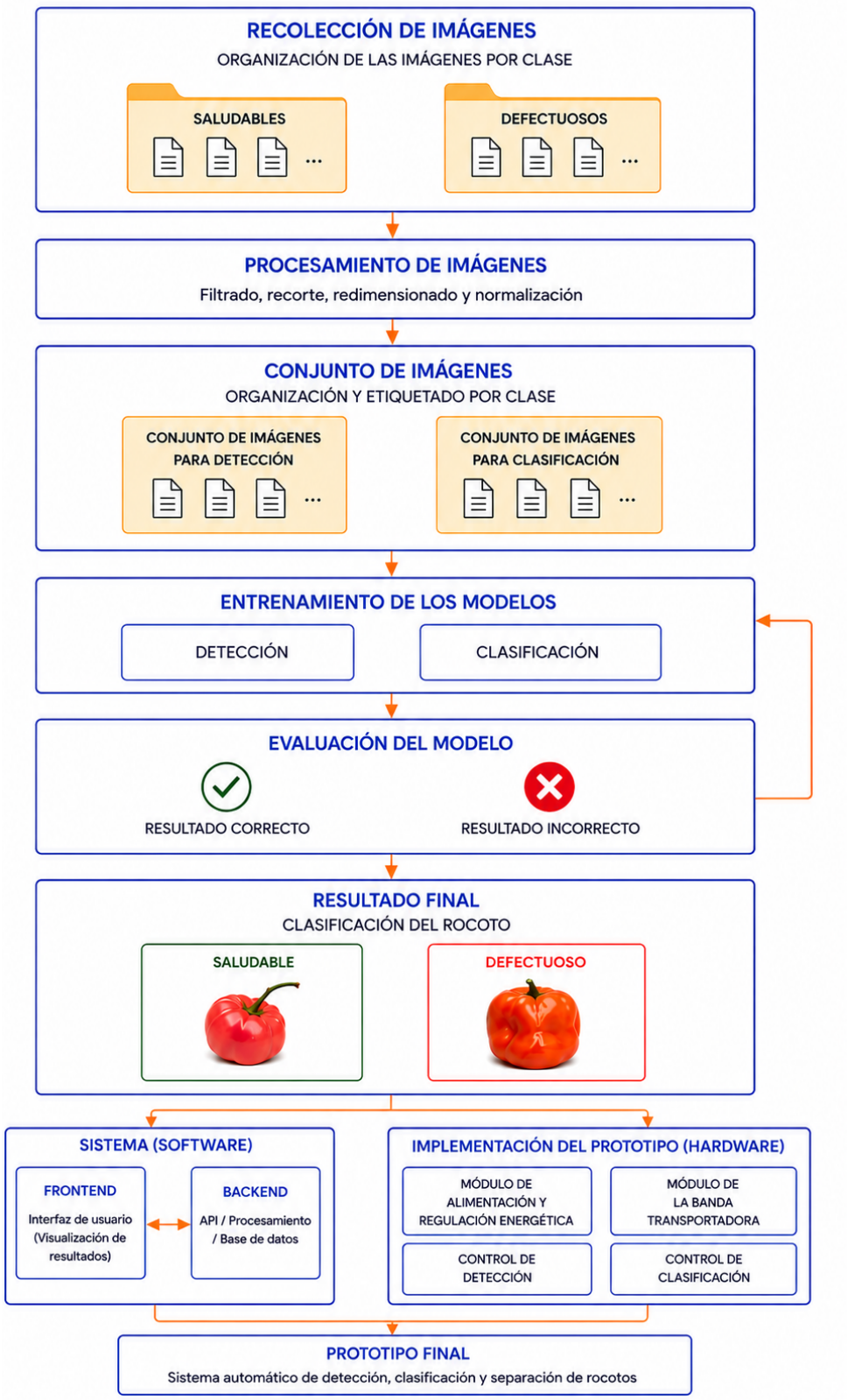
: Adaptado de Chapman et al. (2000). Elaboración propia (2025).

3.3. Flujo de Trabajo del Sistema

El flujo sintetiza de manera gráfica la aplicación de las seis fases de CRISP-DM al presente proyecto, desde la recolección y procesamiento de imágenes hasta la obtención del sistema ROCOBOT.

Figura 9:

Flujo de desarrollo del sistema ROCOBOT



Nota: Elaboración propia (2025).

Fase 1: Comprensión del negocio. Se identificó y delimitó el problema central del proyecto: la clasificación ineficiente y subjetiva de rocotos (*Capsicum pubescens*) en la industria agroalimentaria, que genera pérdidas económicas por el rechazo de productos en buen estado y el aumento del desperdicio. El objetivo de negocio se tradujo en un objetivo técnico concreto: desarrollar un sistema robótico denominado ROCOBOT capaz de detectar y clasificar automáticamente rocotos en dos categorías (saludable y defectuoso) mediante modelos de aprendizaje profundo.

Fase 2: Comprensión de los datos. Se procedió a la recolección de imágenes de rocotos empleando dispositivos móviles en distintas condiciones de iluminación, ángulos y fondos, con el propósito de obtener un conjunto de datos representativo del entorno real de operación. Las imágenes se organizaron en dos categorías: rocotos saludables y rocotos defectuosos, constituyendo la base del aprendizaje supervisado del sistema.

Fase 3: Preparación de los datos**.** Las imágenes recolectadas fueron sometidas a procesos de filtrado, recorte, redimensionamiento y normalización para estandarizar las entradas del sistema. Posteriormente, se realizó el etiquetado manual de cada imagen utilizando herramientas especializadas. El dataset final se dividió en tres subconjuntos: entrenamiento, validación y prueba, garantizando una evaluación objetiva e independiente del desempeño de los modelos.

Fase 4: Modelado. Se planificó el entrenamiento y la comparación de múltiples arquitecturas de aprendizaje profundo para dos etapas independientes: detección de objetos y clasificación binaria del estado del rocoto. El entrenamiento se realizó en la plataforma Google Colab utilizando GPU, lo que permitió reducir significativamente los tiempos de cómputo. Asimismo, se aplicaron técnicas de aumento de datos (*data augmentation*) para mejorar la capacidad de generalización de los modelos ante variaciones en las condiciones de captura.

Fase 5: Evaluación**.** Los modelos entrenados fueron evaluados mediante métricas estándar de rendimiento en visión artificial: precisión (*precision*), exhaustividad (*recall*), media de precisión promedio (*mAP@0.5*) y puntuación F1 (*F1-score*). Los resultados obtenidos permitieron seleccionar los modelos con mejor desempeño para su integración en el sistema final. Además, cuando fue necesario, se retornó a la fase de modelado para ajustar hiperparámetros, evidenciando el carácter iterativo de la metodología.

Fase 6: Despliegue. Los modelos seleccionados fueron integrados en el sistema ROCOBOT, compuesto por un prototipo físico con banda transportadora controlada mediante una Raspberry Pi 4B, un backend desarrollado en FastAPI para el procesamiento de imágenes y la ejecución de inferencias, y un frontend desarrollado en React para la visualización de

resultados en tiempo real. Finalmente, se realizó la validación funcional del sistema completo en condiciones controladas de operación.

3.4. Metodología de Trabajo

La metodología de trabajo que se adoptó en el desarrollo del proyecto ROCOBOT fue Kanban, perteneciente al enfoque de las metodologías ágiles de gestión de proyectos. De acuerdo con Anderson y Carmichael (2016), Kanban se basa en un sistema de tracción (*pull system*), mediante el cual las tareas avanzan de una etapa a la siguiente únicamente cuando existe capacidad disponible en el equipo, evitando la sobrecarga de trabajo y favoreciendo un ritmo de desarrollo sostenible y predecible a lo largo del ciclo de vida del proyecto.

Este enfoque resultó especialmente adecuado para el presente proyecto debido a la naturaleza iterativa e incremental de sus componentes: el desarrollo de modelos de inteligencia artificial requirió ciclos continuos de experimentación, ajuste y validación, mientras que la construcción del prototipo físico demandó la coordinación simultánea de actividades de ingeniería de software, electrónica y mecánica. Kanban proporcionó la flexibilidad necesaria para gestionar estas interdependencias sin imponer cronogramas rígidos que pudieran comprometer la calidad de los entregables.

3.4.1. Principios de Kanban aplicados al proyecto

La implementación de Kanban en ROCOBOT se rigió por los siguientes principios fundamentales establecidos por Anderson y Carmichael (2016):

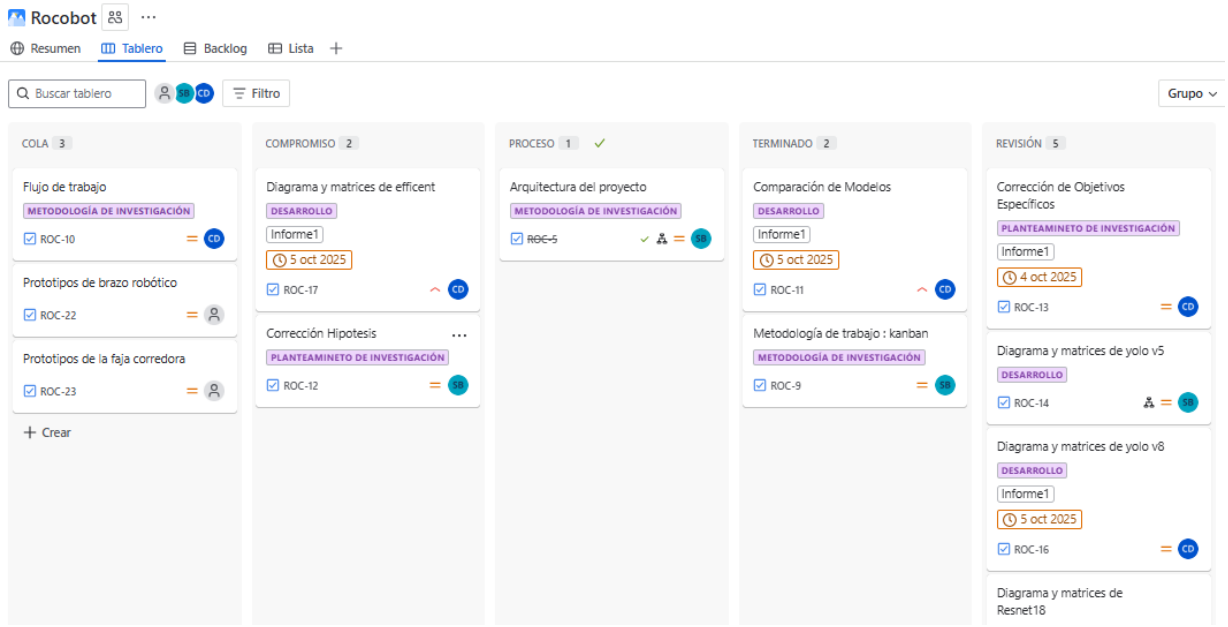
- Visualizar el trabajo: todas las tareas del proyecto se representaron mediante tarjetas distribuidas en el tablero Kanban, permitiendo identificar el estado de cada actividad en tiempo real.
- Limitar el trabajo en curso (WIP): se establecieron límites máximos por columna para evitar la dispersión del equipo y garantizar la conclusión efectiva de cada tarea antes de iniciar una nueva.
- Gestionar el flujo: se monitoreó el lead time de cada tarea para identificar cuellos de botella y optimizar la velocidad de entrega.
- Definir políticas explícitas: cada columna del tablero tuvo criterios de entrada y salida claramente definidos, evitando ambigüedades en la transición de tareas.
- Fomentar la mejora continua: mediante revisiones semanales se identificaron oportunidades de optimización del proceso de desarrollo.

3.4.2. Estructura del tablero Kanban

El tablero Kanban que se diseñó para la gestión del proyecto ROCOBOT se organizó en cinco columnas o canalizaciones que representaron los estados posibles de cada tarea a lo largo del ciclo de desarrollo, tal como se muestra en la Figura 10.

Figura 10

Tablero Kanban aplicado al desarrollo del proyecto ROCOBOT



Nota: Adaptado de Anderson y Carmichael (2016). Elaboración propia (2025).

3.5. Población y muestra

3.5.1. Población

La población de estudio estuvo conformada por todas las unidades de rocoto (*Capsicum pubescens*) producidas en cultivos agrícolas de la región Amazonas, Perú. Los rocotos se clasificaron en dos estados: saludables y defectuosos, que representaron las categorías esenciales para el control de calidad postcosecha mediante el sistema robótico ROCOBOT.

3.5.2. Muestra

El muestreo empleado para la construcción de ambos datasets fue de tipo no probabilístico por conveniencia, dado que las imágenes se obtuvieron directamente del establecimiento del proveedor distribuidor de rocotos ubicado en el distrito de José Leonardo Ortiz, Chiclayo, en función de la disponibilidad de acceso y de la accesibilidad logística para

la recolección periódica de imágenes durante el periodo de estudio. Si bien este tipo de muestreo no garantiza representatividad estadística respecto a la totalidad de la producción nacional de rocoto, resultó adecuado para los fines de la presente investigación tecnológica, orientada a validar la viabilidad técnica del sistema propuesto bajo condiciones reales de un proveedor representativo de la región Amazonas.

La muestra estuvo compuesta por imágenes de rocotos capturadas en la región Amazonas, organizadas en dos datasets independientes para las etapas de detección y clasificación.

Para la etapa de detección, se utilizó un dataset de 5,421 imágenes obtenido mediante filtro de similitud y técnicas de *data augmentation* sobre un conjunto inicial de 2,725 imágenes. Las imágenes fueron organizadas en los subconjuntos *train*, *val* y *test*, incluyendo tanto imágenes con rocotos anotados como imágenes negativas sin anotaciones.

Para la etapa de clasificación, se empleó un dataset de 2,732 imágenes distribuidas en dos clases: defectuoso (clase 0) y saludable (clase 1), con 1,366 imágenes por clase.

3.6. Técnicas, instrumentos, equipos y materiales

3.6.1. Técnicas

Para garantizar la calidad y precisión de los datos, se emplearon dos técnicas principales de recolección: análisis documental y adquisición de datos mediante dispositivos.

- **Análisis documental:** Consiste en la revisión sistemática de documentos, como artículos científicos, libros, revistas especializadas y otras fuentes de información, con el propósito de fundamentar teóricamente una investigación y obtener información relevante para el desarrollo del estudio (Hernández Sampieri & Mendoza, 2018). En este estudio, se empleó para identificar arquitecturas de redes neuronales convolucionales (CNN) y algoritmos de *Deep Learning* aplicables a la detección y clasificación de rocotos.
- **Adquisición de datos mediante dispositivos:** Consiste en la recopilación de información mediante cámaras, sensores y otros dispositivos electrónicos para obtener datos que puedan ser procesados por sistemas computacionales (Szeliski, 2022). En esta investigación, se capturaron imágenes de rocotos para construir los conjuntos de entrenamiento, validación y prueba del dataset, utilizados en el desarrollo y evaluación de los modelos de *Deep Learning*.

3.6.2. Instrumentos

Para evaluar el desempeño del sistema robótico inteligente en la clasificación de rocotos, se utilizó un instrumento de medición que permitió recopilar, analizar y validar los resultados obtenidos. Este instrumento ayudó a medir la precisión (*accuracy*), la métrica F1-Score y otros indicadores de desempeño del modelo de deep learning implementado.

- **Fichas de Observación:** Se emplearon estos instrumentos para registrar el desempeño del sistema robótico en tiempo real, incluyendo su comportamiento, fallos y eficiencia en la clasificación del rocoto.

3.6.3. Equipos

Los dispositivos que se utilizaron para la adquisición de imágenes fueron un iPhone 15 (iOS), con cámara principal con estabilización óptica de imagen (OIS), y un Samsung Galaxy (Android 15). La incorporación de ambos dispositivos con diferentes sensores, temperatura de color y rango dinámico permitió robustecer los modelos ante variaciones propias de distintos equipos de captura.

El procesamiento de datos y el entrenamiento de los modelos se llevaron a cabo en la plataforma Google Colab, haciendo uso de GPU en la nube, lo que permitió reducir significativamente los tiempos de cómputo sin depender de hardware local de alto rendimiento.

3.6.4. Materiales

Los instrumentos que se utilizaron para la construcción del sistema robótico incluyeron herramientas especializadas en diseño, fabricación y ensamblaje. El diseño estructural en 2D se desarrolló en AutoCAD, permitiendo una planificación precisa de los componentes, mientras que el modelo 3D se elaboró en Fusion 360, lo que facilitó la visualización y el ajuste de las dimensiones antes de la fabricación.

Las piezas del robot fueron fabricadas mediante impresión 3D con material PLA, seleccionado por su resistencia mecánica y durabilidad. Además, se empleó corte láser para la elaboración de piezas específicas que requerían mayor precisión y rigidez.

3.6.5. Aspectos éticos de la investigación

El desarrollo de la presente investigación se realizó respetando los principios éticos fundamentales aplicables a la investigación tecnológica. Se contó con la autorización verbal del proveedor distribuidor de rocotos para el acceso a sus instalaciones y la captura de imágenes del producto con fines exclusivamente académicos, sin que ello implicara la divulgación de

información comercial sensible de la empresa. Las imágenes recolectadas fueron empleadas únicamente para la construcción de los datasets de entrenamiento, validación y prueba de los modelos de Deep Learning, y se almacenaron en repositorios de acceso controlado, evitando su uso para fines distintos a los declarados en el presente estudio. Asimismo, se garantizó la originalidad del desarrollo mediante la citación adecuada de todas las Notas teóricas y antecedentes consultados, conforme a las normas de citación vigentes, y se emplearon únicamente herramientas, bibliotecas y plataformas de software con licencias abiertas o de uso académico (Google Colab, PyTorch, FastAPI, React, entre otras), respetando los términos de uso establecidos por sus respectivos desarrolladores. Finalmente, los resultados obtenidos durante el entrenamiento y evaluación de los modelos se reportan de manera íntegra y sin manipulación, incluyendo tanto los desempeños favorables como las limitaciones identificadas en el sistema, en concordancia con los principios de honestidad e integridad científica.

4. RESULTADOS

4.1. Construcción de los datasets de imágenes de rocotos

4.1.1. Proceso de adquisición de imágenes

En la literatura revisada no se encontró ningún dataset público de imágenes de rocotos (*Capsicum pubescens*), por lo que se procedió a construir un dataset propio. Los trabajos más cercanos emplean datasets de frutos de la misma familia botánica *Capsicum*, como pimientos morrones (Mohi-Alden et al., 2023; Abarquez et al., 2024), chiles (Ibrahim et al., 2023; Hespeler et al., 2021) y pimientos picantes (Hyeon et al., 2021), o de otras hortalizas como tomate (Ouyang et al., 2022; Hou et al., 2024; Moya et al., 2025) y frutas diversas (Ismail & Malik, 2022; Dairath et al., 2023). Las metodologías de recolección, etiquetado y augmentation reportadas en dichos antecedentes fueron tomadas como referencia y adaptadas a las características morfológicas y condiciones de distribución agroindustrial del rocoto peruano.

Los dispositivos utilizados para la captura de imágenes fueron un iPhone 15 (iOS, cámara principal de 48 MP con OIS) y un Samsung Galaxy (Android 15), incorporando deliberadamente variabilidad de sensor, temperatura de color y rango dinámico. El uso de dos dispositivos de distintos sistemas operativos es consistente con la práctica reportada por Ouyang et al. (2022), quienes capturaron imágenes en diferentes condiciones de iluminación y dispositivos para robustecer sus datasets de detección en tomate, y con Abarquez et al. (2024), cuyo dataset de 5,000 imágenes de pimientos incorporó variabilidad de captura para mejorar la representatividad del conjunto de datos.

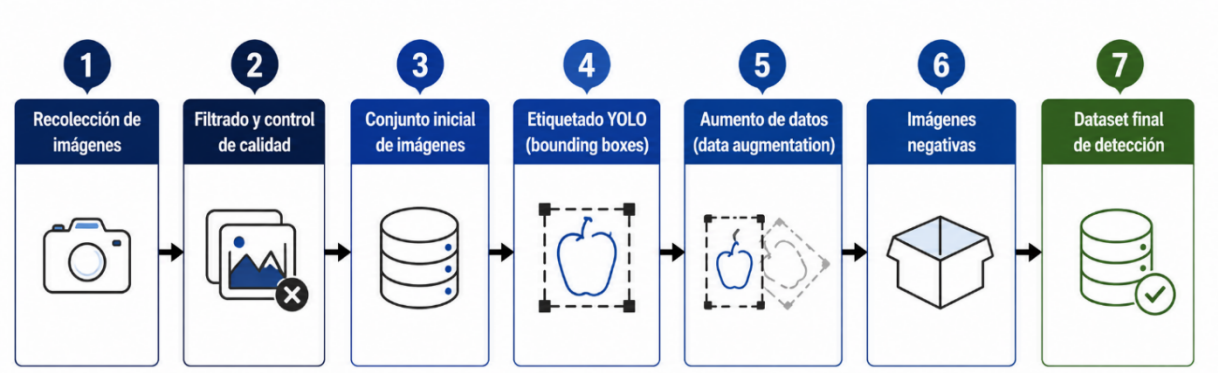
Para la construcción del dataset se realizó un proceso de recolección periódica durante dos años (agosto 2023 – diciembre 2025), con una frecuencia aproximada mensual, condicionada a la disponibilidad de lotes del proveedor distribuidor. Los rocotos son provenientes de la región Amazonas, provincia de Bongará, y eran trasladados por el proveedor distribuidor hacia sus instalaciones ubicadas en el distrito de José Leonardo Ortiz, Chiclayo, donde se realizaron las sesiones de captura sobre las cajas de rocotos recién llegadas. Esta modalidad garantizó la disponibilidad continua de especímenes frescos en distintos estados (saludables y defectuosos) a lo largo de las dos temporadas de cosecha. La decisión de extender la recolección durante dos años responde a la necesidad de capturar variabilidad temporal en las características morfológicas del rocoto (variaciones de color, tamaño, tipo de defecto y grado de madurez) propias de distintas temporadas de cosecha, siguiendo la recomendación de Khalifa et al. (2022) de incorporar diversidad temporal en los datasets para mejorar la robustez

ante condiciones cambiantes de producción. Se obtuvieron dos datasets independientes: uno para la etapa de detección y otro para la etapa de clasificación binaria.

El proceso de construcción de cada dataset siguió flujos independientes adaptados a los requisitos de cada etapa. Para el dataset de detección, el flujo comprende siete etapas basadas en las especificaciones de Ultralytics (2025) para la construcción de datasets con arquitecturas YOLO. La primera etapa corresponde a la recolección de imágenes en condiciones naturales. La segunda aplica filtrado y control de calidad, eliminando imágenes duplicadas, borrosas, sobreexpuestas o con oclusión severa. La tercera constituye la obtención del conjunto inicial listo para etiquetado. La cuarta realiza el etiquetado manual de los rocotos mediante bounding boxes en formato YOLO. La quinta aplica data augmentation (rotaciones, volteo horizontal y ajuste de brillo) para incrementar la diversidad del conjunto. La sexta incorpora imágenes negativas sin instancias de rocoto para mejorar la capacidad discriminativa del modelo ante fondos complejos. La séptima conforma el dataset final listo para entrenamiento, validación y prueba de los modelos YOLO.

Figura 11:

Flujo de construcción del dataset de detección ROCOBOT



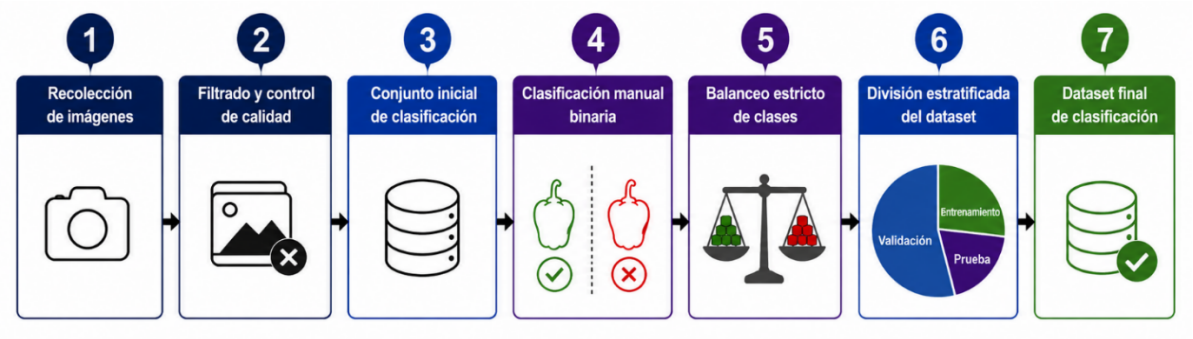
Nota. Elaboración propia (2026). Flujo basado en Ultralytics (2025).

Para el dataset de clasificación, el flujo comprende siete etapas basadas en la metodología de construcción de datasets de clasificación binaria de frutos reportada por Ismail y Malik (2022). La primera etapa corresponde a la recolección de imágenes en condiciones naturales. La segunda aplica filtrado y control de calidad, eliminando imágenes duplicadas, borrosas, sobreexpuestas o con oclusión severa. La tercera constituye la obtención del conjunto inicial para clasificación. La cuarta realiza la clasificación manual binaria de cada imagen en dos clases: saludable o defectuoso. La quinta aplica balanceo estricto del número de imágenes por clase para evitar sesgos en el entrenamiento, siguiendo el principio establecido por

Goodfellow et al. (2016) de que el desbalance entre clases infla artificialmente las métricas de accuracy. La sexta realiza la división estratificada en subconjuntos de entrenamiento, validación y prueba. La séptima conforma el dataset final listo para entrenamiento, validación y prueba del modelo de clasificación..

Figura 12:

Flujo de construcción del dataset de clasificación ROCOBOT



Nota. Elaboración propia (2026). Flujo basado en Ismail y Malik (2022).

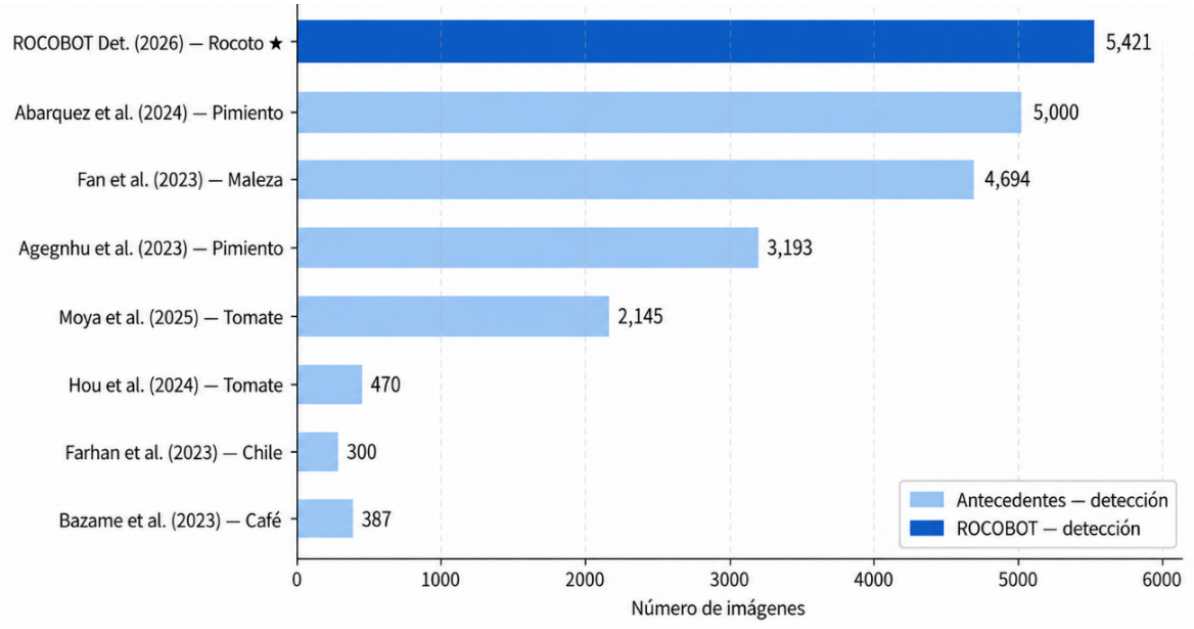
4.1.2. Dataset de detección

El dataset de detección parte de 2,725 imágenes base, de las cuales 1,925 fueron etiquetadas manualmente mediante bounding boxes en formato YOLO con una única clase anotada (rocoto), y 800 corresponden a imágenes negativas sin anotación. La decisión de anotar una única clase responde a que el objetivo de esta etapa es exclusivamente localizar el rocoto dentro de la imagen, siendo el fondo tratado implícitamente por el modelo como región sin objeto de interés, práctica consistente con la empleada por Ibrahim et al. (2023) y Abarquez et al. (2024) para la detección de frutos de la familia *Capsicum* con arquitecturas YOLO.

La Figura 13 compara el tamaño del dataset de detección ROCOBOT con los empleados en los antecedentes de la literatura para sistemas de detección de frutos mediante arquitecturas YOLO.

Figura 13:

Comparación del tamaño del dataset de detección ROCOBOT vs. Antecedentes



Nota. Elaboración propia (2026). El dataset ROCOBOT de detección (5,421 imágenes con augmentation)

El dataset de detección ROCOBOT (5,421 imágenes, incluyendo data augmentation aplicado exclusivamente al subconjunto de entrenamiento) supera en volumen a la mayoría de los antecedentes revisados que emplean arquitecturas YOLO. El menor dataset reportado corresponde a Ibrahim et al. (2023) con 300 imágenes de chile, seguido de Bazame et al. (2023) con 387 imágenes de café y Hou et al. (2024) con 470 imágenes de tomate, mientras que Moya et al. (2025) emplearon 2,145 imágenes de tomate y Abarquez et al. (2024) emplearon 5,000 imágenes de pimientos, fruto de la misma familia botánica *Capsicum*. Cabe precisar que Ibrahim et al. (2023), Bazame et al. (2023), Hou et al. (2024) y Moya et al. (2025) no reportan el uso de técnicas de aumento de datos, siendo el caso de Moya et al. (2025) explícito al respecto, al indicar textualmente que su modelo fue entrenado *"without dataset augmentation"*, por lo que sus cifras corresponden a conteos de imágenes originales. En este sentido, incluso considerando el conjunto base de ROCOBOT previo al aumento de datos (2,725 imágenes), este supera a los cuatro antecedentes mencionados, evidenciando que la ventaja en volumen del dataset ROCOBOT no depende exclusivamente del proceso de *augmentation* aplicado. Considerando el dataset final con *augmentation* (5,421 imágenes), ROCOBOT supera adicionalmente a Abarquez et al. (2024) en volumen total de imágenes. Adicionalmente, la incorporación de 800 imágenes negativas es una práctica no reportada en ninguno de los

antecedentes revisados, que fortalece la capacidad discriminativa del modelo en condiciones reales de operación con fondos variables.

Las 2,725 imágenes base se dividieron en tres subconjuntos independientes (entrenamiento, validación y prueba) mediante la proporción estándar de 70%, 20% y 10% respectivamente, siguiendo los principios del aprendizaje supervisado, donde la separación estricta garantiza que las métricas reportadas reflejen la capacidad real de generalización del modelo ante datos no vistos (Goodfellow, Bengio & Courville, 2016). Esta partición 70/20/10, criterio metodológico central de la presente investigación, se detalla en la Tabla 3.

Tabla 3:

Distribución del dataset de detección por subconjunto y tipo de imagen

Subconjunto	Total imgs	Etiquetadas	Negativas	% del total
train	1,908	1,348	560	70%
val	545	385	160	20%
test	272	192	80	10%
Total	**2,725**	**1,925**	**800**	**100%**

Nota. Partición del conjunto base de imágenes, antes de aplicar data augmentation. Elaboración propia (2026).

Sobre esta partición 70/20/10 se aplicaron técnicas de *data augmentation* (rotaciones, cambios de brillo y volteo horizontal) exclusivamente sobre el subconjunto de entrenamiento, siguiendo la estrategia reportada por Khalifa et al. (2022), quienes demuestran que el *augmentation* mejora la capacidad de generalización de los modelos ante variaciones de iluminación y orientación del objeto. Los subconjuntos de validación (545 imágenes) y prueba (272 imágenes) se mantuvieron intactos, sin ninguna imagen aumentada, evitando así una estimación optimista y no representativa del desempeño real ante datos no vistos (*data leakage*). Adicionalmente, se incorporaron 800 imágenes negativas, dado que en modelos de detección con una única clase, la ausencia de negativas incrementa la tasa de falsos positivos durante la inferencia en entornos con fondos complejos (Ultralytics, 2025).

Como consecuencia del data augmentation aplicado solo a train (que pasó de 1,348 a 4,044 imágenes etiquetadas), la proporción se desplaza a 85%/10%/5% (Tabla 4), sin que esto redefina la partición 70/20/10 original: val y test se mantienen en 545 y 272 imágenes, iguales a la Tabla 3.

Tabla 4:

Distribución final del dataset de detección (tras data augmentation en entrenamiento)

Subconjunto	Total imgs	Etiquetadas	Negativas	% del total
train	4,604	4,044	560	85%
val	545	385	160	10%
test	272	192	80	5%
Total	**5,421**	**4,621 (85%)**	**800 (15%)**	**100%**

Nota. Distribución de imágenes aplicando data augmentation en las etiquetadas. Elaboración propia (2026).

Como resultado de este proceso, el dataset final quedó compuesto por 5,421 imágenes: 4,621 (85%) con presencia de rocoto y 800 (15%) negativas sin anotación. Del total de imágenes con rocoto, el subconjunto de entrenamiento (train) pasó de 1,348 imágenes etiquetadas manualmente a 4,044, incorporando 2,696 imágenes adicionales cuya anotación se generó de forma automática mediante la propagación del *bounding box* durante el *data augmentation*. Los subconjuntos de validación (385) y prueba (192) no fueron modificados: conservan el mismo número de imágenes y la misma anotación manual original de la Tabla 3. Este balance entre imágenes positivas y negativas es necesario para que el modelo cuente con suficientes ejemplos del objeto de interés sin perder capacidad discriminativa frente a fondos sin rocoto.

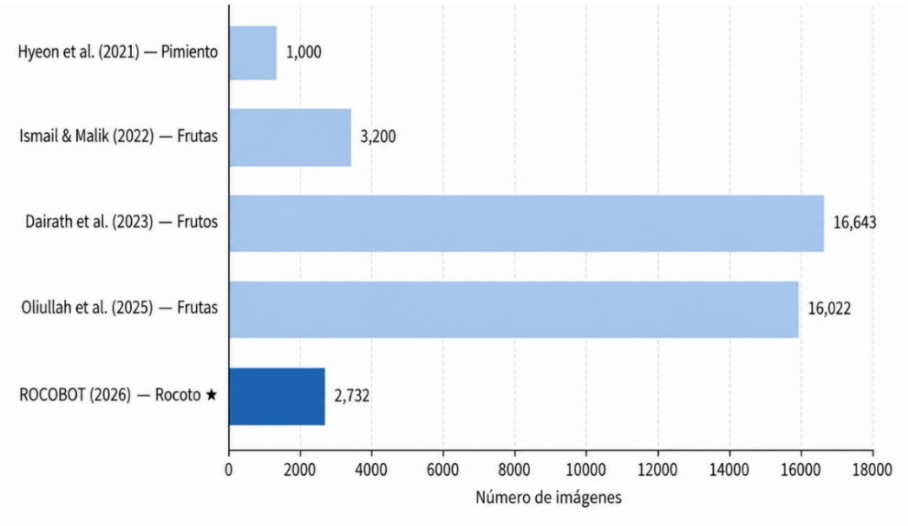
4.1.3. Dataset de clasificación

El dataset de clasificación se conformó con 2,732 imágenes distribuidas en dos clases: defectuoso (clase 0) y saludable (clase 1). A diferencia del dataset de detección, no se aplicaron técnicas de data augmentation en esta etapa, dado que el objetivo es evaluar la capacidad del modelo para discriminar entre ambas clases a partir de imágenes representativas del entorno real de operación. La decisión de conformar un dataset perfectamente balanceado con 1,366 imágenes por clase responde a que el desbalance entre clases en problemas de clasificación binaria introduce sesgo en el entrenamiento, favoreciendo la predicción de la clase mayoritaria e inflando artificialmente las métricas de accuracy (Goodfellow et al., 2016). El balanceo estricto garantiza que las métricas reportadas (accuracy, precisión, recall y F1-score) reflejen el rendimiento real del modelo ante ambas categorías por igual, condición crítica en un sistema de control de calidad donde tanto los falsos positivos como los falsos negativos tienen

consecuencias económicas directas para el productor. La Figura 14 compara el tamaño del dataset de clasificación ROCOBOT con los empleados en los antecedentes de la literatura para sistemas de clasificación de frutos.

Figura 14:

Comparación del dataset de clasificación ROCOBOT vs. antecedentes de la literatura



Nota. Elaboración propia (2026).

El dataset de clasificación ROCOBOT (2,732 imágenes) es de tamaño moderado en comparación con los antecedentes revisados. Dairath et al. (2023) emplearon el dataset Food-11 con 16,643 imágenes para evaluar modelos de clasificación de frutos en un prototipo robótico con Raspberry Pi; Oliullah et al. (2025) utilizaron 16,022 imágenes distribuidas en 14 clases para su sistema híbrido FruVeg_MultiNet de identificación de frutas y vegetales frescos o en descomposición; e Ismail y Malik (2022) emplearon 8,791 imágenes de manzanas (4,693 defectuosas y 3,946 saludables) para clasificación binaria mediante Raspberry Pi. En contraste, Hyeon et al. (2021) emplearon un dataset de menor tamaño, con aproximadamente 1,000 imágenes de pimiento, siendo ROCOBOT superior en volumen a este último antecedente. Sin embargo, un mayor volumen de imágenes no garantiza por sí solo un dataset de calidad, siendo el balanceo estricto entre clases y la estrategia de construcción los factores que garantizan la representatividad del conjunto de datos.

Es importante precisar que la comparación directa del volumen total de imágenes no refleja completamente la complejidad relativa de cada problema de clasificación, dado que los antecedentes revisados abordan un número considerablemente mayor de clases. Dairath et al. (2023) clasificaron 11 categorías de frutos y Oliullah et al. (2025) trabajaron con 14 clases de

frutas y vegetales, lo que resulta en un promedio aproximado de 1,513 y 1,144 imágenes por clase, respectivamente. Ismail y Malik (2022), al abordar igualmente un problema de clasificación binaria (2 clases) con su dataset de manzanas, disponen de un promedio de 4,395 imágenes por clase. En contraste, el dataset de clasificación ROCOBOT dispone de 1,366 imágenes por clase, un volumen inferior al de Ismail y Malik (2022) en este mismo tipo de problema binario, pero superior al promedio por clase de Dairath et al. (2023) y Oliullah et al. (2025), pese a que estos últimos manejan datasets totales considerablemente mayores. Esto sugiere que, en términos de densidad de información disponible por categoría a discriminar, el dataset ROCOBOT se posiciona de forma competitiva frente a antecedentes que abordan problemas de clasificación multiclase, aunque con margen de mejora frente a antecedentes binarios de mayor escala.

La división 70/20/10 adoptada responde a que en datasets de tamaño moderado con clases perfectamente balanceadas, destinar el 10% al subconjunto de prueba es estadísticamente suficiente para estimar el error de generalización de forma confiable (Goodfellow et al., 2016). El 10% equivale a 278 imágenes (139 por clase). El 20% destinado a validación (544 imágenes) constituye un subconjunto de tamaño suficiente para monitorear el desempeño del modelo de forma confiable durante el proceso de entrenamiento. La proporción de entrenamiento del 70%, con 1,910 imágenes, constituye el subconjunto de mayor volumen dentro de la partición, destinado a que el modelo disponga de suficientes ejemplos para aprender las características visuales específicas del fruto.

Tabla 5:

Distribución del dataset de clasificación por subconjunto y clase

Conjunto	**N° imágenes**	**Clase 0 (defectuoso)**	**Clase 1 (saludable)**
Entrenamiento	1,910 (70%)	955	955
Validación	544 (20%)	272	272
Prueba	278 (10%)	139	139
Total	**2,732 (100%)**	**1,366**	**1,366**

Nota. Distribución del conjunto de imágenes para clasificación. Elaboración propia (2026).

El dataset completo empleado en esta investigación se encuentra disponible para su consulta en el Anexo E.

4.2. Entrenamiento y comparación de modelos de detección y clasificación

4.2.1. Condiciones de entrenamiento

El entrenamiento de todos los modelos de inteligencia artificial del sistema ROCOBOT, tanto para la etapa de detección como para la etapa de clasificación, se realizó en la plataforma Google Colab, haciendo uso de una GPU Tesla T4 proporcionada por el entorno en la nube. Esta elección permitió acelerar significativamente el proceso de entrenamiento respecto a un procesamiento basado únicamente en CPU, reduciendo los tiempos de cómputo sin requerir hardware local de alto rendimiento. Las particularidades de cada etapa, en cuanto a número de épocas, tamaño de lote e hiperparámetros específicos, se detallan en los apartados correspondientes a la detección y a la clasificación. Los pesos entrenados de los seis modelos de detección y los dos modelos de clasificación, junto con los notebooks y scripts de evaluación empleados, se presentan en el Anexo A.1.

4.2.2. Selección de modelo de detección

En la revisión de los antecedentes de la literatura se encontraron diversos estudios que emplean modelos de detección de objetos para la identificación de frutos y hortalizas en sistemas de visión artificial. La Tabla 6 presenta los antecedentes identificados, ordenados de mayor a menor año de publicación, indicando el producto evaluado y el modelo de inteligencia artificial empleado en cada caso.

Tabla 6:

Modelos de detección de objetos empleados en antecedentes de productos agrícolas

N°	Autor/Año	Producto	Modelo IA
1	Moya et al. (2025)	Tomate	YOLOv8
2	Abarquez et al. (2024)	Pimiento	YOLOv8
3	Hou et al. (2024)	Tomate	YOLOv8
4	Wang et al. (2024)	Fresa	YOLOv8+
5	Bazame et al. (2023)	Café	YOLOv4
6	Dhanush et al. (2023)	Múltiples	YOLO/MASK-RCNN
7	Ibrahim et al. (2023)	Chile	YOLOv5
8	Wang et al. (2023)	Melocotón	YOLOv5
9	Li et al. (2022)	Fresa	YOLOv4+Deep SORT
10	Ouyang et al. (2022)	Tomate	YOLOv5+MobileNetV2
11	Wang et al. (2022)	Tomate	Faster R-CNN+ResNet50

12	Hespeler et al. (2021)	Chile	YOLOv3/Mask-RCNN

Nota. Elaboración propia (2025), basada en estudios revisados.

En base a los antecedentes presentados, se encontró que las arquitecturas de la familia YOLO son las más empleadas y con mejores resultados reportados para la detección de frutos de la familia *Capsicum* y hortalizas similares. Si bien algunos antecedentes emplean versiones anteriores como YOLOv3 (Hespeler et al., 2021) y YOLOv4 (Bazame et al., 2023; Li et al., 2022), estas arquitecturas han sido progresivamente superadas en precisión y eficiencia computacional por las versiones posteriores: YOLOv5 introduce un backbone CSPDarknet53 más eficiente y mejoras en el proceso de entrenamiento, mientras que YOLOv8 incorpora un head desacoplado y la función de pérdida DFL, mejorando la precisión de localización respecto a sus predecesoras. Por esta razón, se descartaron las versiones YOLOv3 y YOLOv4 como modelos candidatos, priorizando las arquitecturas más recientes y con mejor desempeño reportado en los antecedentes: YOLOv5 (Ibrahim et al., 2023; Wang et al., 2023; Ouyang et al., 2022) y YOLOv8 (Moya et al., 2025; Abarquez et al., 2024; Hou et al., 2024; Wang et al., 2024).

Cada una de estas arquitecturas ofrece, además, distintas variantes según su tamaño y número de parámetros, las cuales representan un compromiso entre capacidad representacional y velocidad de inferencia. Dado que el sistema ROCOBOT requiere procesamiento en tiempo real sobre hardware embebido (Raspberry Pi 4B), se decidió no limitar la comparación a una sola variante por arquitectura, sino evaluar múltiples tamaños de cada familia para determinar empíricamente el balance óptimo entre precisión y tiempo de inferencia: las variantes small, medium y large de YOLOv5 (s, m, l), y las variantes nano y small de YOLOv8 (n, s), siendo la variante nano la de mayor interés por su menor costo computacional para despliegue en hardware limitado.

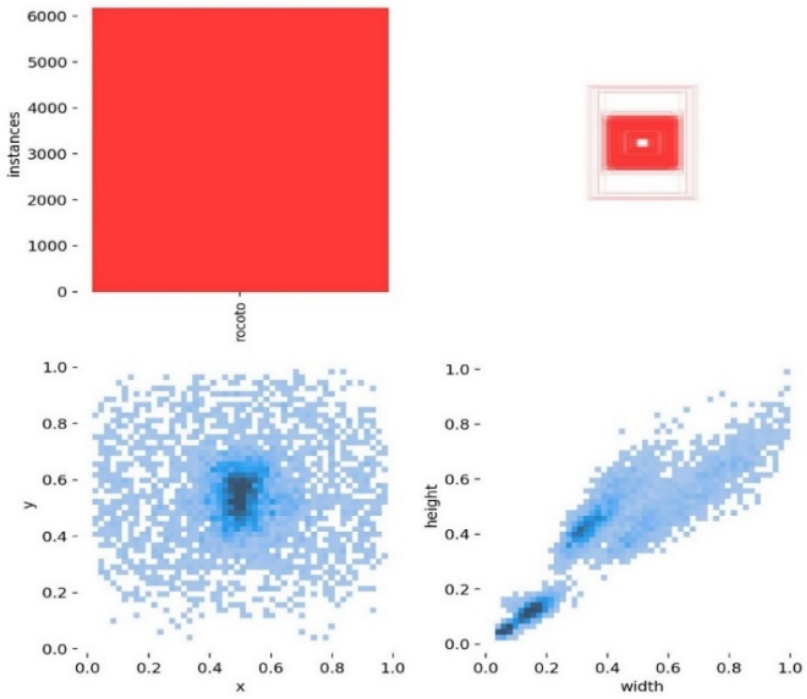
Adicionalmente, la reciente publicación de YOLOv11 por Ultralytics (2025) introduce mejoras arquitectónicas (el módulo C3k2 y el módulo de atención espacial paralela PSA) que justifican su inclusión como modelo candidato, evaluando su variante nano (YOLOv11n) a fin de comparar directamente con YOLOv8n bajo el mismo criterio de eficiencia computacional. En base a este análisis, se ha decidido entrenar y comparar seis variantes de la familia YOLO, cubriendo tres generaciones de la arquitectura: YOLOv5s, YOLOv5m e YOLOv5l, y YOLOv8n, YOLOv8s e YOLOv11n.

4.2.3. Entrenamiento de los modelos de detección

Todos los modelos emplearon el mismo dataset de 5,421 imágenes anotadas. A continuación se muestran la distribución de instancias y el correlograma.

Figura 15:

Distribución de instancias y posición de cajas delimitadoras del dataset de detección

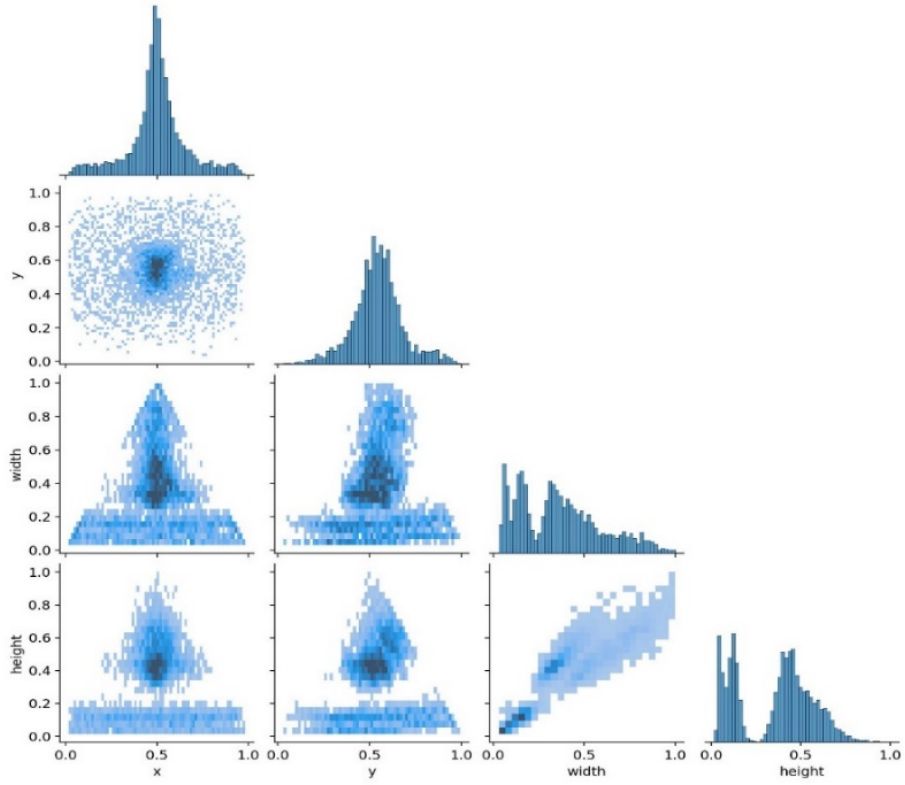


Nota. Elaboración propia (2025).

La Figura 15 presenta el histograma de instancias por clase (izquierda) y el mapa de densidad de la posición (x, y) de las cajas delimitadoras (derecha). La clase rocoto registra ~6,000 instancias, concentradas mayormente en el centro de la imagen.

Figura 16:

Correlograma de etiquetas del dataset de detección



Nota. Elaboración propia (2025).

La Figura 16 presenta la matriz de dispersión y las distribuciones de las variables x, y, width y height de las cajas delimitadoras. La fuerte correlación entre width y height confirma cajas aproximadamente cuadradas, y la concentración de x e y en torno a 0.5 indica ausencia de sesgo posicional en el dataset.

4.2.3.1. Resultados del modelo YOLOv5

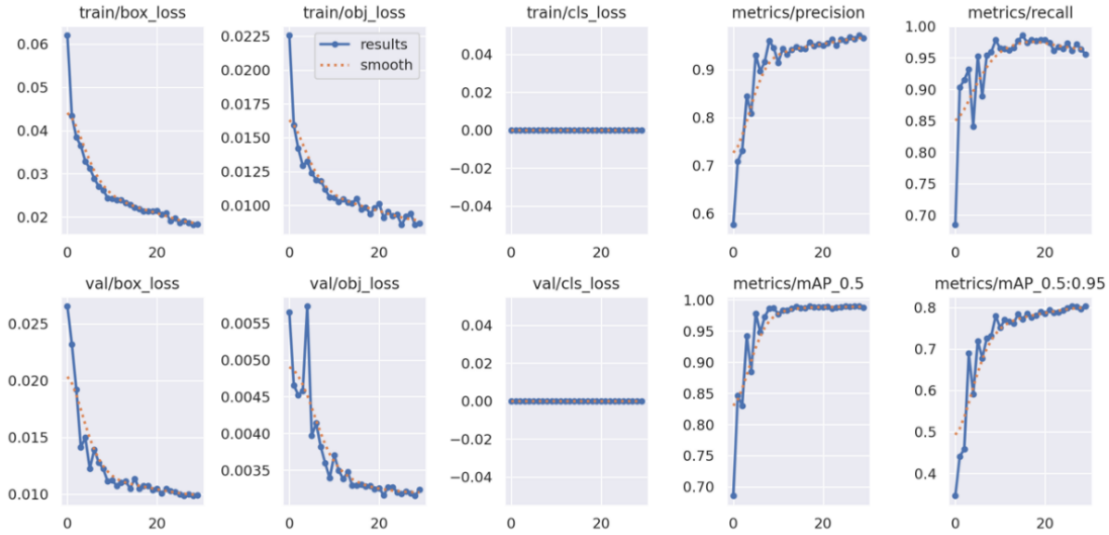
YOLOv5 es una arquitectura de detección de objetos de la familia YOLO desarrollada por Ultralytics (2025), con backbone CSPDarknet53, cuello PANet y tres cabezas de detección para objetos de diferentes escalas, emplea un head acoplado y una función de pérdida compuesta por tres componentes: box_loss (error de localización de caja delimitadora), obj_loss (confianza de presencia del objeto) y cls_loss (error de clasificación de clase). Se entrenaron tres variantes que difieren en profundidad y número de parámetros: YOLOv5s (~7.2 M), YOLOv5m (~21.2 M) y YOLOv5l (~46.5 M), cada una durante 30 épocas.

a. YOLOv5s (small)

Es la variante más ligera en base a los entrenados de YOLOv5, con el menor número de parámetros (~7.2 M) y el menor tiempo de inferencia. Su diseño prioriza la velocidad sobre la capacidad representacional, siendo adecuada para dispositivos embebidos con recursos limitados como la Raspberry Pi 4B.

Figura 17

Curvas de entrenamiento y validación del modelo YOLOv5s

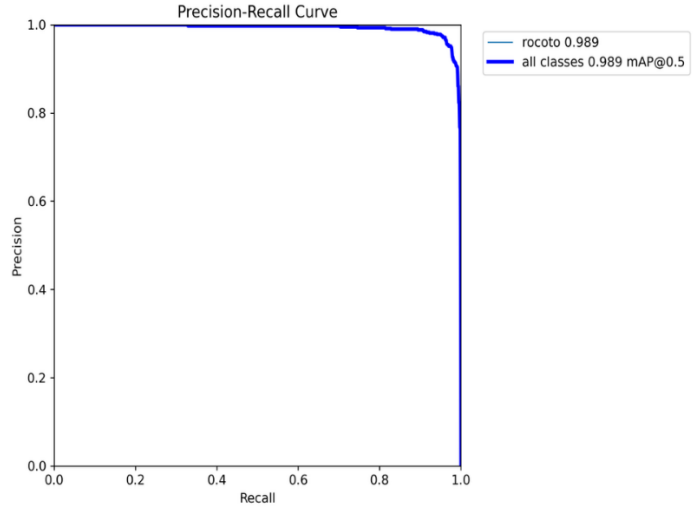


Nota. Elaboración propia (2025). Entrenamiento: 30 épocas, batch=16, imgsz=640, GPU.

La Figura 17 muestra la evolución de las métricas durante el entrenamiento de YOLOv5s. El box_loss de entrenamiento desciende de 0.062 a 0.018 y la obj_loss de 0.022 a 0.009, evidenciando un aprendizaje progresivo de la localización y detección de rocotos. La cls_loss permanece en cero en todas las épocas, comportamiento esperado en modelos de detección con una única clase donde no existe ambigüedad entre categorías. La val/box_loss converge de 0.027 a 0.010 y la val/obj_loss de 0.0055 a 0.0033, sin divergencia entre curvas de entrenamiento y validación, lo que indica ausencia de sobreajuste. La precisión sube de 0.57 en la época 1 hasta estabilizarse entre 0.96 y 0.99 a partir de la época 15. El mAP@0.5 alcanza su mejor valor de 0.9898 en la época 27 y el mAP@0.5:0.95 sube de 0.40 hasta 0.80.

Figura 18:

Curva precisión-recall del modelo YOLOv5s

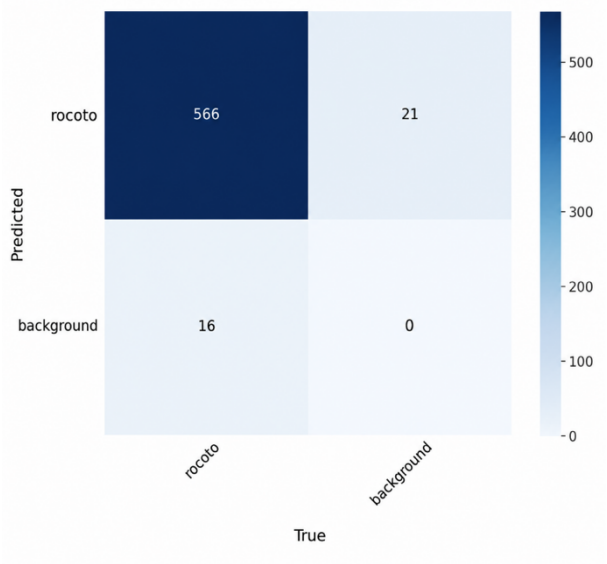


Nota. Elaboración propia (2025). mAP@0.5 = 0.989 para la clase rocoto.

La curva precisión-recall de YOLOv5s (Figura 18) presenta un perfil casi rectangular, con el área bajo la curva (mAP@0.5) de 0.989. La precisión se mantiene en 1.0 para valores de recall entre 0.0 y 0.40, con una caída gradual hasta 0.96 al acercarse a recall = 0.97. Esta forma indica que el modelo detecta la gran mayoría de rocotos con mínima generación de falsos positivos en los umbrales de confianza medios y altos.

Figura 19:

Matriz de confusión normalizada del modelo YOLOv5s



Nota. Elaboración propia (2025). Valores normalizados sobre el conjunto de validación.

La matriz de confusión normalizada (Figura 19) confirma el alto desempeño de YOLOv5s: el 99% de las instancias reales de rocoto son predichas correctamente como rocoto, con solo el 1% clasificado erróneamente como fondo (falsos negativos). La tasa de falsos positivos es nula en términos normalizados. La Tabla 7 presenta las métricas numéricas correspondientes al mejor epoch del modelo.

Tabla 7:

Métricas de evaluación del modelo YOLOv5s — mejor epoch (época 27 de 30)

Métrica / Valor	Precision (P)	Recall (R)	mAP@0.5	mAP@0.5:0.95	F1-Score
	96.37%	97.17%	98.98%	80.16%	96.77%

Nota. Elaboración propia (2025). Métricas extraídas del archivo results.csv del entrenamiento.

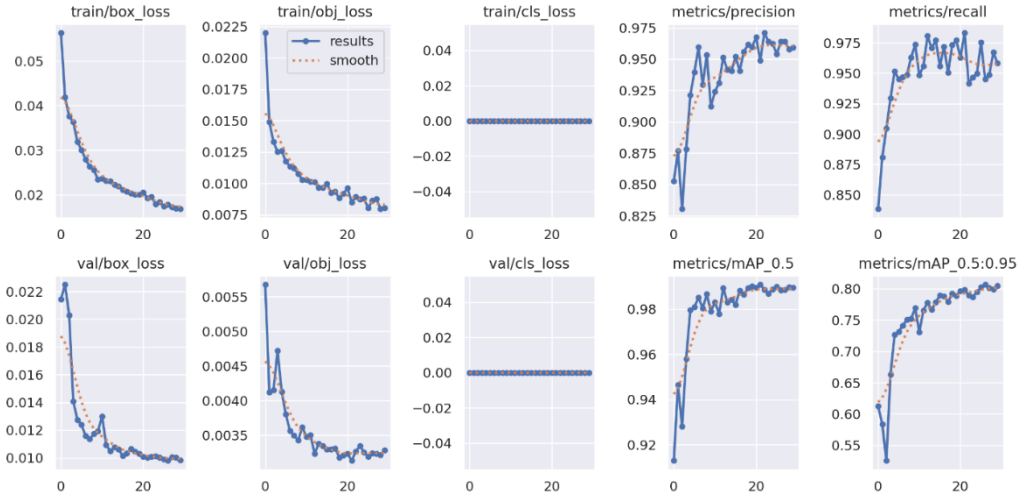
El modelo YOLOv5 alcanzó un desempeño sobresaliente en la detección de frutos de rocoto, según los indicadores de evaluación obtenidos durante la validación. Los resultados muestran una precisión (P) de 0.965 y un recall (R) de 0.965, lo que refleja un equilibrio adecuado entre la detección correcta de los objetos y la minimización de falsos positivos. El valor de mAP@0.5 = 0.989 indica un rendimiento casi perfecto en la identificación de las instancias dentro de la clase analizada, mientras que el mAP@0.5–0.95 = 0.808 demuestra una alta capacidad del modelo para mantener la precisión en diferentes umbrales de intersección sobre unión (IoU).

b. YOLOv5m (medium)

YOLOv5m es la variante de escala media con ~21.2 M de parámetros, que ofrece un mejor balance entre capacidad representacional y eficiencia respecto a YOLOv5s. Su mayor profundidad le permite capturar características más complejas de los rocotos, lo que se refleja especialmente en una mejora del recall.

Figura 20:

Curvas de entrenamiento y validación del modelo YOLOv5m

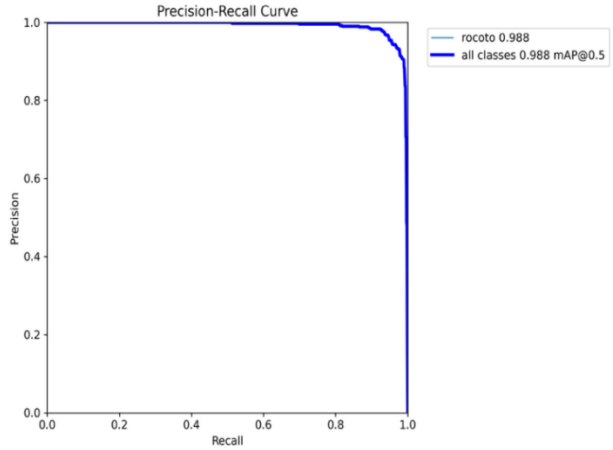


Nota. Elaboración propia (2025). Entrenamiento: 30 épocas, batch=16, imgsz=640, GPU-

Las curvas de YOLOv5m (Figura 20) siguen la misma estructura de tres pérdidas que YOLOv5s, con cls_loss constante en cero. La box_loss desciende de 0.057 a 0.017 y la obj_loss de 0.022 a 0.008, mostrando un ajuste más preciso gracias al mayor número de parámetros. La precisión de validación oscila entre 0.95 y 0.975 en las épocas finales, con mayor variabilidad que YOLOv5s atribuible a la mayor sensibilidad del modelo profundo ante variaciones en los lotes de validación. El mAP@0.5 alcanza su máximo de 0.9909 en la época 22, siendo el mayor mAP@0.5 puntual entre todas las variantes YOLOv5. El mAP@0.5:0.95 converge a 0.805.

Figura 21:

Curva precisión-recall del modelo YOLOv5m

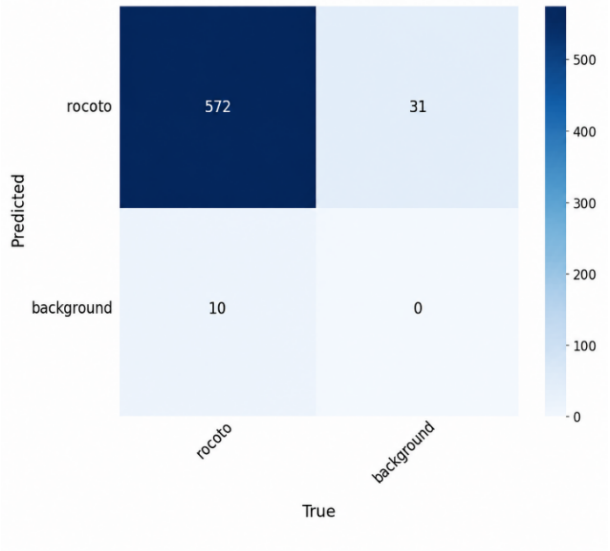


Nota. Elaboración propia (2025). mAP@0.5 = 0.988 para la clase rocoto.

La curva PR de YOLOv5m (Figura 21) muestra mAP@0.5 = 0.988. La caída de precisión es más pronunciada a partir de recall = 0.80, lo que es consistente con su mayor recall (98.3%): el modelo recupera más instancias a expensas de un mayor número de detecciones a umbrales de confianza bajos.

Figura 22:

Matriz de confusión normalizada del modelo YOLOv5m



Nota. Elaboración propia (2025). Valores normalizados sobre el conjunto de validación.

La matriz de confusión normalizada de YOLOv5m (Figura 22) muestra una tasa de detección correcta del 98% (2% de falsos negativos) y tasa de falsos positivos nula. La Tabla 8 presenta las métricas del mejor epoch.

Tabla 8:

Métricas de evaluación del modelo YOLOv5m — mejor epoch (época 21 de 30)

Métrica / Valor	***Precision (P)***	***Recall (R)***	***mAP@0.5***	***mAP@0.5:0.95***	***F1-Score***
	94.88%	98.32%	99.09%	79.87%	96.57%

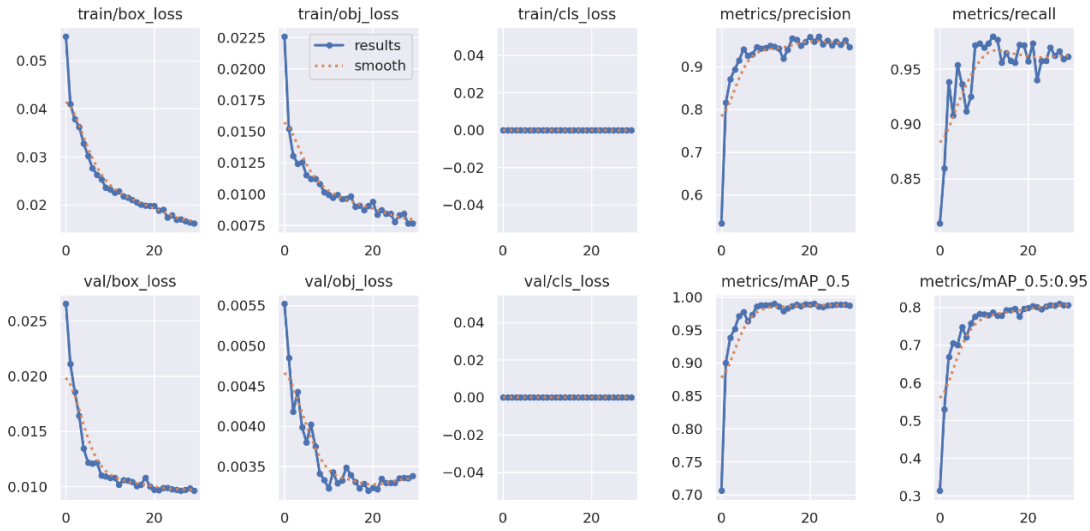
Nota. Elaboración propia (2025). Métricas extraídas del archivo results.csv del entrenamiento.

c. YOLOv5l (large)

De las variantes entrenadas de la familia YOLOv5, YOLOv5l es la más profunda con aproximadamente 46.5 millones de parámetros, significativamente mayor que YOLOv5s (~7.2 M) y YOLOv5m (~21.2 M). Su mayor capacidad representacional permite capturar características más complejas y abstractas de los objetos, lo que en teoría debería traducirse en una mayor precisión de detección. Sin embargo, esta profundidad adicional implica un mayor tiempo de inferencia (~135 ms sobre la Raspberry Pi 4B) y requiere más recursos computacionales durante el entrenamiento. El modelo fue entrenado con los mismos hiperparámetros que las variantes small y medium: 30 épocas, tamaño de imagen 640×640, tamaño de lote de 16 y GPU Tesla T4 en Google Colab.

Figura 23:

Curvas de entrenamiento y validación del modelo YOLOv5l — 30 épocas



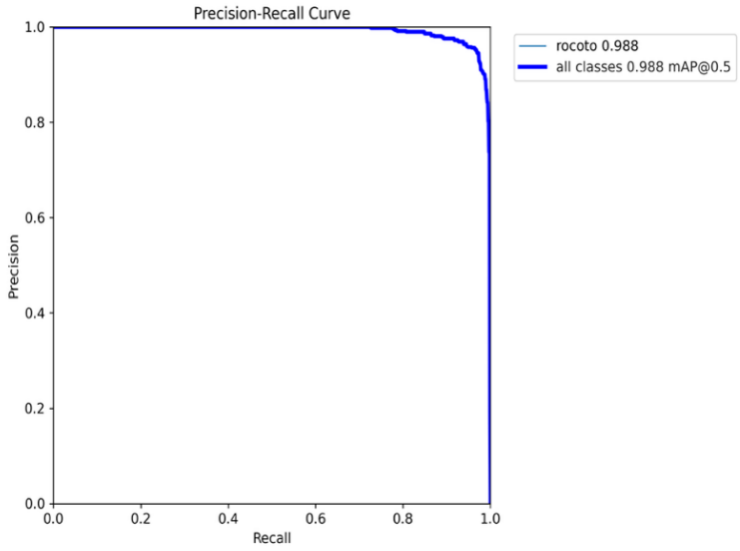
Nota. Elaboración propia (2025). Entrenamiento: 30 épocas, batch=16, imgsz=640, GPU-

La Figura 23 muestra las curvas de entrenamiento de YOLOv5l. El box_loss de entrenamiento desciende de 0.055 en la época 1 hasta 0.016 en las épocas finales, con una caída pronunciada en las primeras 5 épocas y una convergencia suave y estable a partir de la época 10. La obj_loss desciende de 0.022 a ~0.007, siguiendo un patrón similar. Como en las variantes anteriores, la cls_loss permanece en cero durante todo el entrenamiento al tratarse de una única clase de detección.

Las pérdidas de validación (val/box_loss y val/obj_loss) convergen de forma paralela a las de entrenamiento sin divergencia, lo que confirma ausencia de sobreajuste. El val/box_loss desciende de 0.027 a 0.010 y la val/obj_loss de 0.0055 a 0.0032, valores muy similares a los de YOLOv5s y YOLOv5m. La precisión de validación presenta mayor variabilidad en YOLOv5l (fluctúa entre 0.92 y 0.96 en las épocas intermedias) respecto a YOLOv5s, lo que es característico de modelos más profundos que son más sensibles a los lotes de validación específicos. El mAP@0.5 alcanza su valor máximo de 0.9899 en la época 22 y converge a ~0.987 en las épocas finales. El mAP@0.5:0.95 muestra la progresión más consistente: parte de 0.31 en la época 1 y sube progresivamente hasta 0.806 al finalizar el entrenamiento.

Figura 24:

Curva precisión-recall del modelo YOLOv5l

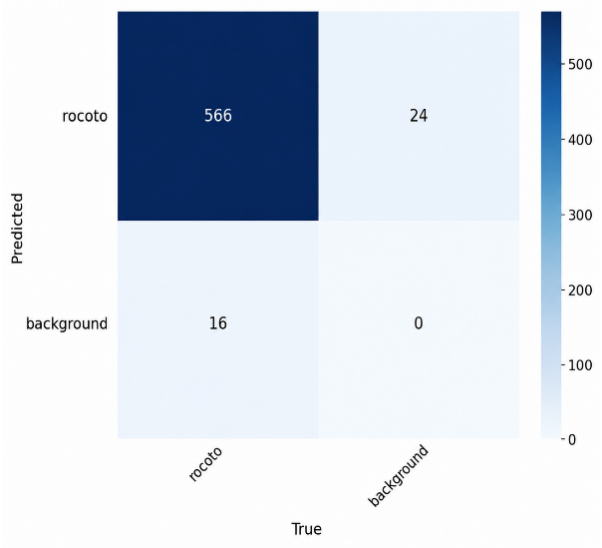


Nota. Elaboración propia (2025). mAP@0.5 = 0.988 para la clase rocoto.

La curva precisión-recall de YOLOv5l (Figura 24) muestra un mAP@0.5 de 0.988, resultado idéntico al de YOLOv5m. El perfil de la curva es casi rectangular: la precisión se mantiene en 1.0 para valores de recall entre 0.0 y 0.40, con una caída gradual que se inicia a partir de recall = 0.60 y se torna más pronunciada al acercarse al recall máximo (~0.98). La caída comienza antes que en YOLOv5s (recall=0.40 vs recall=0.40) pero con mayor suavidad en los umbrales intermedios, lo que refleja la mayor capacidad de YOLOv5l para mantener precisión ante recalls intermedios gracias a su profundidad adicional.

Figura 25:

Matriz de confusión normalizada del modelo YOLOv5l



Nota. Elaboración propia (2025). Valores normalizados sobre el conjunto de validación.

La matriz de confusión normalizada de YOLOv5l (Figura 25) muestra que el 98% de las instancias reales de rocoto son predichas correctamente como rocoto, con un 2% clasificado como fondo (falsos negativos normalizados). La tasa de falsos positivos es nula (0.00 normalizado), igual que en YOLOv5m. Este resultado, idéntico al de YOLOv5m en términos normalizados, indica que YOLOv5l no mejora la tasa de detección normalizada respecto a la variante medium, aunque sí muestra un mAP@0.5:0.95 ligeramente superior gracias a su mayor precisión de localización en umbrales exigentes de IoU.

Tabla 9:

Métricas de evaluación del modelo YOLOv5l — mejor epoch

Métrica / Valor	***precision (P)***	***Recall (R)***	***mAP@0.5***	***mAP@0.5:0.95***	***F1-Score***
	95.90%	97.30%	99.00%	80.40%	96.59%

Nota. mAP@0.5:0.95, mayor entre las variantes YOLOv5 evaluadas. Elaboración propia (2025).

Las métricas del mejor epoch confirman que YOLOv5l obtiene la mayor mAP@0.5:0.95 (80.38%) entre las tres variantes YOLOv5 evaluadas, aunque la diferencia respecto a YOLOv5m (79.87%) es de solo 0.51 puntos porcentuales. Esto indica que la mayor profundidad de YOLOv5l se traduce en una localización

ligeramente más precisa de las cajas delimitadoras en umbrales de IoU exigentes, pero sin una mejora sustancial en la detección general. YOLOv5l no supera a YOLOv8n en mAP@0.5:0.95 (80.38% vs 81.30%), lo que es determinante para la selección del modelo de detección del sistema ROCOBOT. Adicionalmente, el mayor tiempo de inferencia de YOLOv5l (~135 ms) frente a YOLOv5s (~110 ms) y YOLOv5m (~115 ms) lo hace menos adecuado para el procesamiento en tiempo real del prototipo, donde cada milisegundo adicional impacta en la capacidad de procesamiento por minuto.

4.2.3.2. Resultados de los modelos YOLOv8

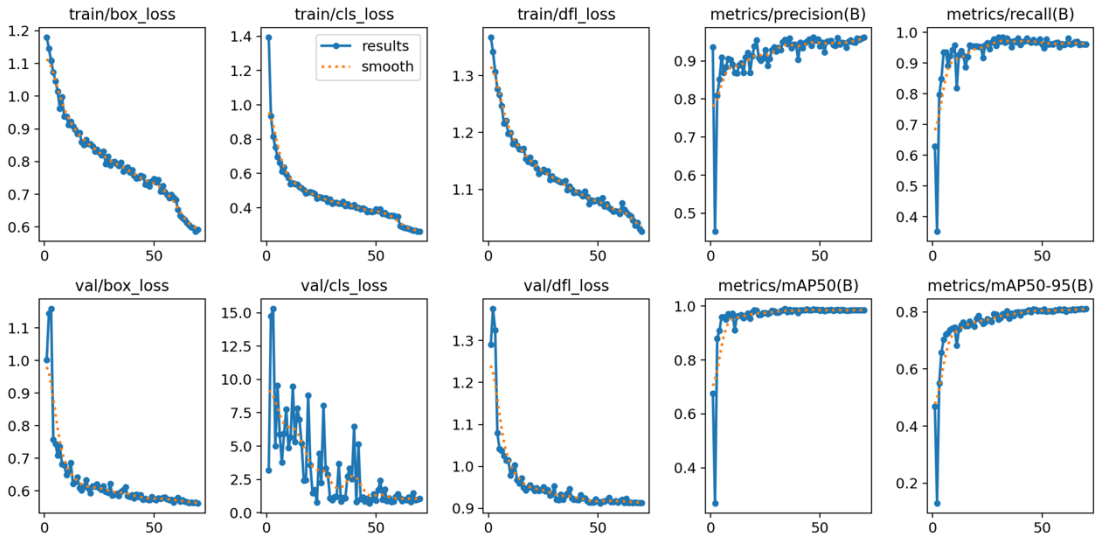
En una segunda fase de evaluación se entrenaron dos variantes de YOLOv8: nano (n) y small (s). YOLOv8 introduce mejoras arquitectónicas significativas respecto a YOLOv5: head desacoplado (*decoupled head*) que separa las predicciones de clasificación y regresión de caja, backbone C2f, y función de pérdida DFL (*Distribution Focal Loss*) como tercera componente que modela la distribución probabilística de las coordenadas de las cajas delimitadoras. Ambas variantes fueron entrenadas con 70 épocas para garantizarconvergencia completa.

a. YOLOv8n (nano)

YOLOv8n es la variante más ligera de la familia YOLOv8, con aproximadamente 3.2 millones de parámetros (3,011,043 parámetros exactos según el resumen del modelo). Incorpora mejoras arquitectónicas relevantes respecto a YOLOv5: head desacoplado (*decoupled head*) que separa las predicciones de clasificación y regresión de caja, backbone C2f y función de pérdida DFL (*Distribution Focal Loss*) como tercera componente, que modela la distribución probabilística de las coordenadas de las cajas delimitadoras. El modelo fue entrenado con 70 épocas, tamaño de imagen 640×640, tamaño de lote de 16, optimizador AdamW (seleccionado automáticamente, lr=0.002) y GPU Tesla T4 en Google Colab.

Figura 26:

Curvas de entrenamiento y validación del modelo YOLOv8n

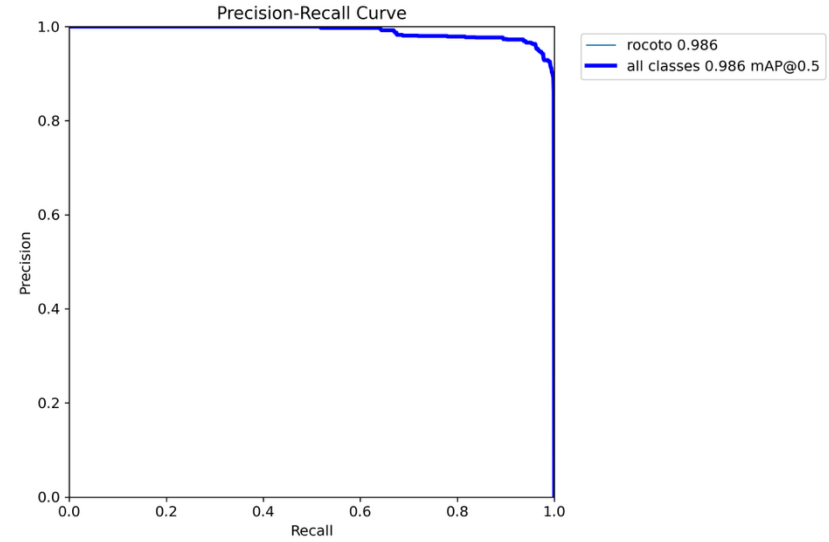


Nota. Elaboración propia (2025). Entrenamiento: 70 épocas, batch=16, imgsz=640, GPU.

La Figura 26 muestra las curvas de entrenamiento de YOLOv8n. El box_loss de entrenamiento desciende de 1.180 en la época 1 hasta 0.591 en la época 70, con una caída pronunciada durante las primeras 10 épocas y una convergencia suave a partir de la época 40. La cls_loss desciende de 1.395 a 0.261, y la dfl_loss de 1.368 a 1.026, ambas siguiendo un patrón de descenso continuo similar entre sí. Es particularmente notable la inestabilidad observada en la época 2, donde la precisión cae abruptamente de 0.936 (época 1) a 0.452, el recall de 0.629 a 0.352 y el mAP@0.5:0.95 de 0.469 a 0.130, para luego recuperarse progresivamente a partir de la época 3 (precisión 0.809, mAP@0.5:0.95 0.551) y estabilizarse desde la época 5 en adelante (precisión 0.909, mAP@0.5:0.95 0.703). Este comportamiento de caída y recuperación temprana es característico de la fase inicial de ajuste fino cuando se transfieren pesos preentrenados a un dominio con una única clase. A partir de la época 30, todas las métricas de validación se estabilizan: la precisión oscila entre 0.92 y 0.96, el recall entre 0.95 y 0.98, el mAP@0.5 se mantiene sobre 0.98 y el mAP@0.5:0.95 asciende progresivamente hasta alcanzar su valor máximo de 0.811 en la época 70.

Figura 27:

Curva precisión-recall del modelo YOLOv8n

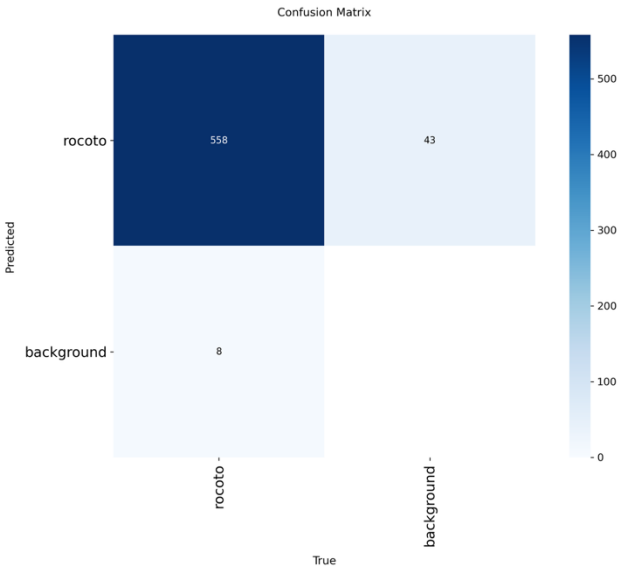


Nota. Elaboración propia (2025). mAP@0.5 = 0.990 para la clase rocoto.

La curva precisión-recall de YOLOv8n (Figura 27) muestra un mAP@0.5 de 0.986. El perfil de la curva es prácticamente rectangular: la precisión se mantiene en 1.0 hasta aproximadamente recall = 0.65, seguida de una caída gradual que se torna más pronunciada al acercarse al recall máximo, descendiendo hasta aproximadamente 0.92.

Figura 28:

Matriz de confusión del modelo YOLOv8n



Nota. Elaboración propia (2025). Conjunto de validación: 582 instancias de rocotos.

La matriz de confusión de YOLOv8n (Figura 28) muestra 558 verdaderos positivos y 8 falsos negativos sobre un total de 566 instancias reales de rocoto en el conjunto de validación, además de 43 falsos positivos. Cabe precisar que esta matriz se calcula a un umbral de confianza fijo (0.25 por defecto), por lo que sus proporciones (98.6% de aciertos sobre el total de instancias) difieren ligeramente del recall reportado en la Tabla 10 (96.10%), el cual se calcula de forma agregada sobre múltiples umbrales de confianza durante la validación.

Tabla 10:

Métricas de evaluación del modelo YOLOv8n — mejor época

Métrica	**Precisión (P)**	**Recall (R)**	**mAP@0.5**	**mAP@0.5:0.95**	**F1-Score**
Valor	96.20%	96.10%	98.60%	81.30%	96.15%

Nota. *Valores obtenidos de la validación final del modelo (best.pt).* Elaboración propia (2025).

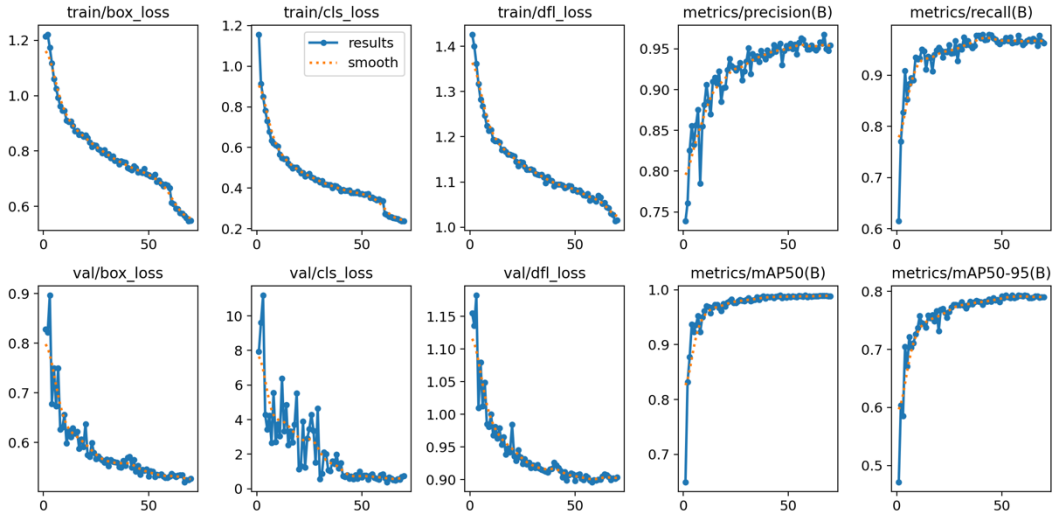
Las métricas finales de YOLOv8n muestran un desempeño sólido y equilibrado en la tarea de detección del rocoto. La precisión de 96.20% indica que, de todas las detecciones realizadas por el modelo, un 96.20% corresponden efectivamente a rocotos reales, con una baja proporción de falsos positivos. El recall de 96.10% refleja que el modelo logra identificar correctamente el 96.10% de todos los rocotos presentes en el conjunto de validación, dejando sin detectar solo una fracción mínima de los objetos reales. El equilibrio entre ambas métricas se ve reflejado en el F1-Score de 96.15%, que confirma que el modelo no favorece la precisión a costa del recall ni viceversa, resultado deseable en un sistema de clasificación de calidad donde tanto los falsos positivos (rocotos defectuosos clasificados como saludables) como los falsos negativos (rocotos no detectados) representan un costo para el productor. El mAP@0.5 de 98.60% evidencia una localización prácticamente perfecta de los rocotos cuando se exige una superposición moderada entre la caja predicha y la caja real (IoU $\geq$ 0.5). Por su parte, el mAP@0.5:0.95 de 81.30% —una métrica considerablemente más exigente, al promediar el desempeño en un rango de umbrales de IoU desde 0.5 hasta 0.95— confirma que el modelo no solo detecta la presencia del rocoto, sino que delimita sus bordes con un nivel de precisión geométrica alto, incluso bajo criterios estrictos de superposición.

b. YOLOv8s (small)

YOLOv8s es la variante small de YOLOv8, con mayor número de parámetros que YOLOv8n (~11.2 M). Conserva la misma arquitectura general de la familia YOLOv8 (head desacoplado, backbone C2f y función de pérdida DFL), pero con mayor profundidad y ancho de canales, lo que le otorga mayor capacidad representacional a costa de un mayor tiempo de inferencia. El modelo fue entrenado con los mismos hiperparámetros que YOLOv8n: 70 épocas, tamaño de imagen 640×640, tamaño de lote de 16 y GPU Tesla T4 en Google Colab.

Figura 29:

Curvas de entrenamiento y validación del modelo YOLOv8s — 70 épocas



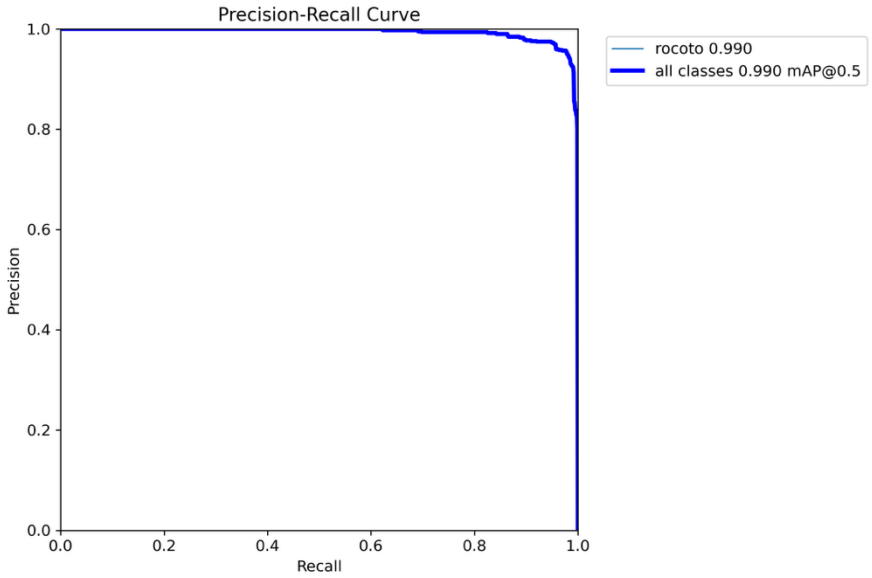
Nota. *Elaboración propia (2025). Entrenamiento: 70 épocas, batch=16, imgsz=640, GPU.*

La Figura 29 muestra las curvas de entrenamiento de YOLOv8s. El box_loss de entrenamiento desciende de aproximadamente 1.22 en la época 1 hasta 0.55 en las épocas finales, con una caída pronunciada durante las primeras 10 épocas y una convergencia suave a partir de la época 40. La cls_loss desciende de aproximadamente 1.15 a 0.23, y la dfl_loss de aproximadamente 1.43 a 1.01. Al igual que en YOLOv8n, la val/cls_loss presenta una inestabilidad marcada durante las primeras 30 épocas, con un pico inicial cercano a 11 y oscilaciones frecuentes entre 2 y 6, antes de estabilizarse por debajo de 1.0 hacia el final del entrenamiento. La precisión de validación parte de aproximadamente 0.74, con una caída adicional a ~0.79 alrededor de la época 8, para luego ascender de forma sostenida —aunque con fluctuaciones— hasta estabilizarse

cerca de 0.95 desde la época 50. El recall muestra un comportamiento similar, partiendo de ~0.62 y estabilizándose por encima de 0.96 hacia el final del entrenamiento. El mAP@0.5 converge a un valor cercano a 0.99, mientras que el mAP@0.5:0.95 asciende de forma progresiva desde ~0.47 en la época 1 hasta aproximadamente 0.79 al finalizar las 70 épocas.

Figura 30:

Curva precisión-recall del modelo YOLOv8s

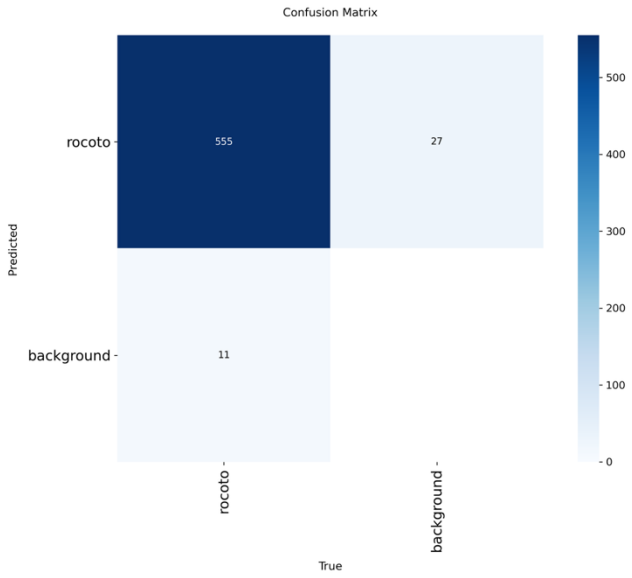


Nota. Elaboración propia (2025). mAP@0.5 = 0.990 para la clase rocoto.

La curva precisión-recall de YOLOv8s (Figura 31) muestra un mAP@0.5 de 0.990, el mayor valor puntual registrado entre los modelos de detección evaluados. El perfil de la curva mantiene precisión de 1.0 hasta aproximadamente recall = 0.62, seguida de una caída gradual y sostenida hasta cerca de 0.95 en el tramo intermedio, con un descenso final más pronunciado que lleva la precisión hasta aproximadamente 0.82 en el extremo de mayor recall.

Figura 31:

Matriz de confusión del modelo YOLOv8s



Nota. Elaboración propia (2025). Valores absolutos sobre el conjunto de validación.

La matriz de confusión de YOLOv8s (Figura 32) muestra 555 verdaderos positivos y 11 falsos negativos sobre un total de 566 instancias reales de rocoto en el conjunto de validación, además de 27 falsos positivos. Al igual que en YOLOv8n, esta matriz se calcula a un umbral de confianza fijo (0.25 por defecto), por lo que su proporción de aciertos (98.1% sobre el total de instancias) difiere ligeramente del recall reportado en la Tabla 11, calculado de forma agregada sobre múltiples umbrales de confianza.

Tabla 11:

Métricas de evaluación del modelo YOLOv8s — métricas finales

Métrica / Valor	precision (P)	Recall ®	mAP@0.5	mAP@0.5:0.95	F1-Score
	95.20%	97.90%	**99.00%**	79.30%	96.50%

Nota. Valores obtenidos de la validación final del Elaboración propia (2025).

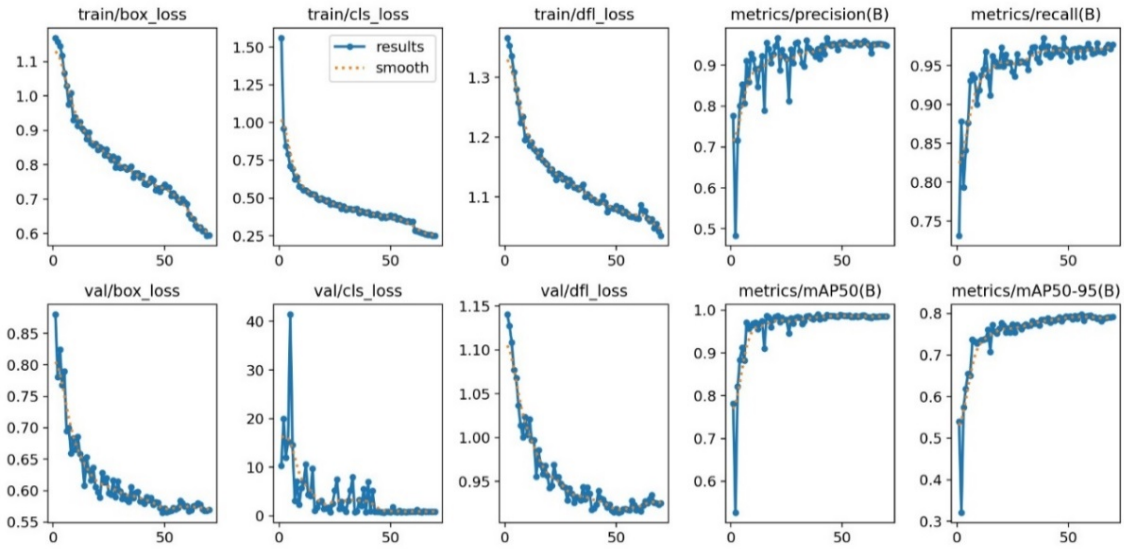
Las métricas finales de YOLOv8s muestran un desempeño equilibrado entre precisión y recall. La precisión de 95.20% indica que la gran mayoría de las detecciones realizadas por el modelo corresponden efectivamente a rocotos reales, con una proporción baja de falsos positivos. El recall de 97.90% refleja que el modelo logra identificar correctamente casi la totalidad de los rocotos presentes en el conjunto de validación, dejando sin detectar solo una fracción mínima de los objetos reales. El F1-Score de 96.50% confirma que este balance entre precisión y recall resulta favorable, sin que ninguna de las dos métricas se sacrifique de forma significativa en favor de la otra. El mAP@0.5 de 99.00% evidencia una localización prácticamente perfecta de los rocotos bajo un criterio de superposición moderado (IoU ≥ 0.5). Por su parte, el mAP@0.5:0.95 de 79.30% —métrica considerablemente más exigente, al promediar el desempeño en un rango de umbrales de IoU desde 0.5 hasta 0.95— confirma que el modelo delimita los bordes del rocoto con un nivel razonable de precisión geométrica, aunque con margen de mejora bajo criterios estrictos de superposición.

4.2.3.3. YOLOv11n (nano)

YOLOv11n es la variante nano de la arquitectura más reciente de Ultralytics, que incorpora el módulo C3k2 con bloques de convolución más eficientes y el módulo PSA (*Parallel Spatial Attention*) para una mejor captura de contexto global. El modelo fue entrenado con los mismos hiperparámetros que las variantes YOLOv8: 70 épocas, tamaño de imagen 640×640, tamaño de lote de 16 y GPU Tesla T4 en Google Colab.

Figura 32:

Curvas de entrenamiento y validación del modelo YOLOv11n

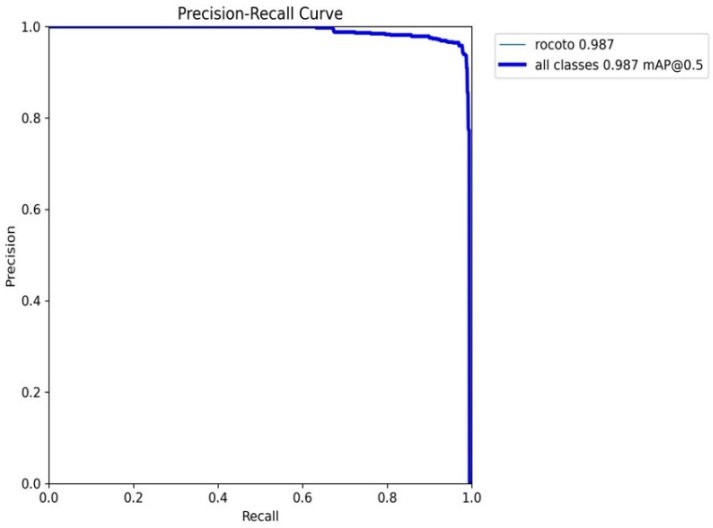


Nota. Elaboración propia (2025). Entrenamiento: 70 épocas, batch=16, imgsz=640, GPU.

La Figura 32 muestra las curvas de entrenamiento de YOLOv11n. Las curvas revelan una inestabilidad inicial pronunciada: el val/cls_loss presenta un pico cercano a 40 durante las primeras épocas, y la precisión de validación cae hasta aproximadamente 0.50 en ese mismo tramo inicial, antes de recuperarse progresivamente. Esta inestabilidad temprana es característica de arquitecturas con módulos de atención (PSA) que requieren un mayor número de épocas para estabilizar sus pesos durante la fase inicial de ajuste fino. A partir de las épocas intermedias, las métricas de validación se estabilizan: la precisión y el recall convergen a valores altos, el mAP@0.5 se estabiliza cerca de 0.987, y el mAP@0.5:0.95 asciende de forma progresiva desde aproximadamente 0.30 en la época 1 hasta 0.794 al finalizar el entrenamiento.

Figura 33:

Curva precisión-recall del modelo YOLOv11n

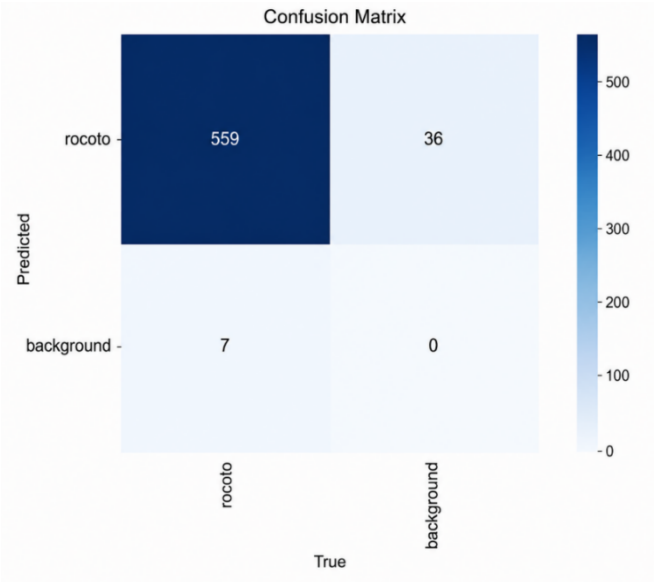


Nota. Elaboración propia (2025). mAP@0.5 = 0.987 para la clase rocoto.

La curva precisión-recall de YOLOv11n (Figura 33) muestra un mAP@0.5 de 0.987. El perfil de la curva mantiene precisión de 1.0 en la mayor parte del rango de recall, con una caída que se torna más pronunciada a partir de recall ≈ 0.95, descendiendo en el tramo final de mayor exigencia.

Figura 34:

Matriz de confusión del modelo YOLOv11n



Nota. Elaboración propia (2025). Conjunto de validación: 595 instancias de rocotos.

La matriz de confusión de YOLOv11n (Figura 34) muestra 559 verdaderos positivos y 36 falsos negativos sobre un total de 595 instancias reales de rocoto en el conjunto de validación, equivalente a un 93.9% de aciertos y un 6.1% de rocotos no detectados, además de 7 falsos positivos.

Tabla 12:

Métricas de evaluación del modelo YOLOv11n — métricas finales

Métrica / Valor	precision (P)	Recall (R)	mAP@0.5	mAP@0.5:0.95	F1-Score
	95.80%	97.20%	98.70%	79.40%	96.49%

Nota. Elaboración propia (2025). Métricas obtenidas del log de entrenamiento

Las métricas finales de YOLOv11n muestran un desempeño sólido en la tarea de detección del rocoto. La precisión de 95.80% indica que la gran mayoría de las detecciones realizadas corresponden efectivamente a rocotos reales, con una baja proporción de falsos positivos. El recall de 97.20% refleja que el modelo identifica correctamente la gran mayoría de los rocotos presentes en el conjunto de validación. El F1-Score de 96.49% confirma un balance favorable entre ambas métricas. El mAP@0.5 de 98.70% evidencia una localización prácticamente perfecta bajo un criterio de superposición moderado (IoU ≥ 0.5), mientras que el mAP@0.5:0.95 de 79.40% — métrica más exigente que evalúa el desempeño en un rango amplio de umbrales de IoU— confirma que el modelo delimita los bordes del rocoto con un nivel razonable de precisión geométrica, aunque con margen de mejora bajo criterios estrictos de superposición.

4.2.3.4. Análisis comparativo de los modelos de detección

En base al entrenamiento desarrollado en los apartados anteriores, se obtuvieron las métricas finales de los seis modelos de detección evaluados. La Tabla 13 consolida dichos resultados, ordenados de menor a mayor mAP@0.5:0.95, que constituye el criterio principal de selección por ser la métrica más exigente del estándar COCO, evaluando la precisión de localización en umbrales de IoU entre 0.5 y 0.95.

Tabla 13:

Comparación de métricas de los seis modelos de detección

Modelo	Precisión	Recall	mAP@0.5	mAP @0.5:0.95	F1-Score	Rank.
YOLOv8s	95.20%	97.90%	99.00%	79.30%	96.50%	6°
YOLOv5m	94.88%	98.32%	99.09%	79.87%	96.57%	5°
YOLOv11n	95.80%	97.20%	98.70%	79.40%	96.49%	4°*
YOLOv5s	96.37%	97.17%	98.98%	80.16%	96.77%	3°
YOLOv5l	95.90%	97.30%	99.00%	80.40%	96.59%	2°
YOLOv8n	**96.20%**	**96.10%**	**98.60%**	**81.30%**	**96.15%**	**1°**

Nota. Modelo seleccionado. Elaboración propia (2025).

Tras la fase de entrenamiento, evaluación y comparación de las seis arquitecturas de detección, YOLOv8n fue seleccionado como modelo de detección del sistema ROCOBOT al obtener el mayor mAP@0.5:0.95 (81.30%), superando a YOLOv5l (80.40%), YOLOv5s (80.16%), YOLOv5m (79.87%), YOLOv11n (79.40%) y YOLOv8s (79.30%). Esta métrica, la más exigente del estándar COCO, fue determinante por sobre el mAP@0.5, donde YOLOv5m obtiene el valor más alto (99.09%) sin ser el modelo con mejor capacidad de localización precisa, y por sobre el F1-Score, donde YOLOv5s registra el mayor equilibrio puntual (96.77%) sin superar a YOLOv8n en localización exigente. YOLOv8n ofrece así el mejor balance entre precisión de localización, tamaño reducido del modelo (~3.2 M de parámetros) y menor tiempo de inferencia, características determinantes para su despliegue en el hardware embebido (Raspberry Pi 4B) del sistema ROCOBOT.

Por otro lado, se contrasta el desempeño del modelo seleccionado (YOLOv8n) con los antecedentes reportados en la literatura para sistemas de detección de frutos con arquitecturas YOLO.

Tabla 14:

Comparación de métricas de los seis modelos de detección vs. Antecedentes

Autor/Año	Modelo	Métrica reportada	Valor
Ibrahim et al. (2023)	YOLOv5	Precisión / Recall / mAP@0.5	88.0% / 90.6% / 93.7%
Ouyang et al. (2022)	YOLOv5	Recall / mAP@0.5	85.20% / 90.40%
Abarquez et al. (2024)	YOLOv8	Precisión macro (a IoU 0.5)	95.90%
Wang et al. (2023)	YOLOv5	"Detección" (métrica no especificada)	99.60%
Hou et al. (2024)	YOLOv8	"Detección de áreas defectuosas"	NR
YOLOv8n-ROCOBOT	**YOLOv8**	**Precisión / Recall / mAP@0.5 / mAP@0.5:0.95**	**96.20% / 96.10% / 98.60% / 81.30%**

Nota.. Todos los modelos propuestos superan el mAP@0.5 reportado por Ouyang et al. (2022)

YOLOv8n alcanza una precisión de 96.20%, un recall de 96.10% y un mAP@0.5 de 98.60%. Comparado con Ibrahim et al. (2023), quienes reportan para YOLOv5 en chile una precisión, recall y mAP@0.5 promedio de 88.0%, 90.6% y 93.7% respectivamente, YOLOv8n supera al antecedente en las tres métricas. Frente a Ouyang et al. (2022), cuyo modelo YOLOv5+MobileNetV2 para tomate alcanzó un recall de 85.20% y un mAP@0.5 de 90.40%, YOLOv8n también obtiene mejores resultados en ambos indicadores. Respecto a Abarquez et al. (2024), quienes reportan una precisión de 95.90% con YOLOv8 para pimientos —fruto de la misma familia botánica *Capsicum*—, YOLOv8n la supera ligeramente. En cuanto a Wang et al. (2023) y Hou et al. (2024), ambos reportan cifras de "detección" cercanas al 99%, pero sin especificar si corresponden a precisión, recall o mAP, por lo que no es posible establecer una comparación directa y equivalente con las métricas de YOLOv8n; se incluyen únicamente como referencia del nivel de desempeño reportado en la literatura para sistemas de detección de frutos con arquitecturas YOLO.

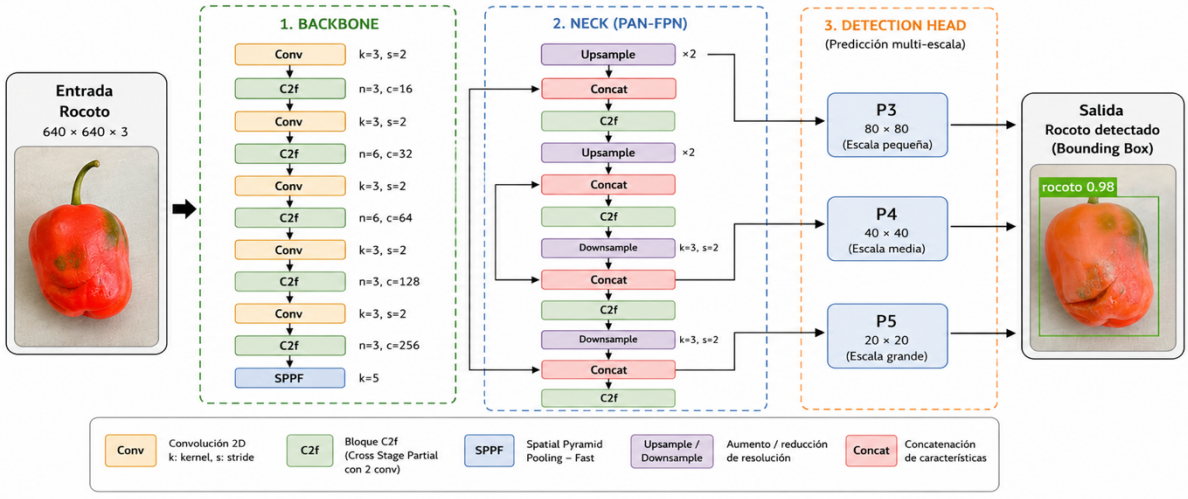
4.2.3.5. Arquitectura del modelo de detección seleccionado

En base a los resultados obtenidos, YOLOv8n fue seleccionado como el modelo de detección del sistema ROCOBOT. Para el entrenamiento de este modelo se empleó su arquitectura en configuración nano, la cual procesa imágenes de 640×640×3 píxeles mediante tres bloques principales: backbone para extracción de características, cuello PAN-FPN para fusión multiescala y cabezas de detección para predicción multi-escala.

La Figura 35 presenta el pipeline completo de la arquitectura YOLOv8n con la que fue entrenado el modelo de detección de rocotos.

Figura 35:

Arquitectura del modelo YOLOv8n para la etapa de detección



Nota. Elaboración propia (2026).

El backbone integra bloques Conv y C2f para la extracción jerárquica de características visuales, culminando en un módulo SPPF (Spatial Pyramid Pooling Fast) con kernel k=5 que captura contexto a múltiples escalas. El cuello PAN-FPN combina operaciones de Upsample y Downsample con bloques Concat y C2f para fusionar características de distintas resoluciones. Las tres cabezas de detección P3 (80×80), P4 (40×40) y P5 (20×20) permiten detectar objetos de escala pequeña, media y grande respectivamente, garantizando la detección robusta del rocoto independientemente de su tamaño relativo en la imagen capturada.

4.2.4. Selección del modelo de clasificación

En la revisión de los antecedentes de la literatura se encontraron diversos estudios que emplean arquitecturas de redes neuronales convolucionales para la clasificación binaria de frutos en sistemas con hardware embebido. La Tabla 15 presenta los antecedentes identificados, indicando el producto evaluado y el modelo de inteligencia artificial empleado en cada caso.

Tabla 15:

Modelos de clasificación empleados en antecedentes de productos agrícolas

N°	Autor/Año	Producto	Modelo IA
1	Oliullah et al. (2025)	Frutas/vegetales	EfficientNetB7+ResNet50
2	Agegnehu et al. (2023)	Pimiento	VGG16+AlexNet
3	Dairath et al. (2023)	Manzana/Plátano	EfficientNet/ResNet
4	Ismail & Malik (2022)	Manzana/Plátano	EfficientNet/ResNet
5	Hyeon et al. (2021)	Pimiento picante	VGG+ResNet50

Nota. Elaboración propia (2025), basada en estudios revisados.

En base a los antecedentes presentados, se encontró que ResNet y EfficientNet son las arquitecturas más empleadas y con mejores resultados reportados para la clasificación binaria de frutos en sistemas con hardware embebido similar al del sistema ROCOBOT (Raspberry Pi). Ismail y Malik (2022) emplearon EfficientNet y ResNet obteniendo 99.2% de accuracy para frutas sobre Raspberry Pi, Dairath et al. (2023) utilizaron EfficientNet alcanzando 96.75% con hardware embebido, y Hyeon et al. (2021) aplicaron ResNet50 para pimientos de la familia Capsicum con 99.6% mediante fine-tuning.

Por ello, se decidió entrenar y comparar ResNet18 y EfficientNetB0, ambas mediante transfer learning con fine-tuning por fases sobre ImageNet, dado que ninguno de los antecedentes reporta un dataset de Capsicum pubescens que permita el entrenamiento desde cero con resultados confiables. Se priorizó ResNet18 sobre versiones más profundas como ResNet50 debido a su menor costo computacional, adecuado para el procesamiento en tiempo real sobre la Raspberry Pi 4B del sistema ROCOBOT, mientras que EfficientNetB0 fue seleccionado como representante de su familia por ser la variante base de menor complejidad, permitiendo una comparación equilibrada en términos de eficiencia computacional.

4.2.5. Entrenamiento de los modelos de clasificación

En la etapa de clasificación binaria se evaluaron dos arquitecturas de redes neuronales convolucionales preentrenadas en ImageNet: ResNet18 y EfficientNetB0. El objetivo es determinar el estado del rocoto (saludable o defectuoso) a partir de las imágenes procesadas en

la etapa de detección. Ambas arquitecturas fueron evaluadas mediante transfer learning con fine-tuning por fases, sobre el mismo dataset balanceado de 2,732 imágenes (1,366 por clase), con división 70/20/10 para entrenamiento, validación y prueba, tamaño de imagen 224×224 píxeles, optimizador Adam, lr=0.001 y función de pérdida Binary Cross-Entropy. Se presentan a continuación los resultados del mejor modelo de cada arquitectura.

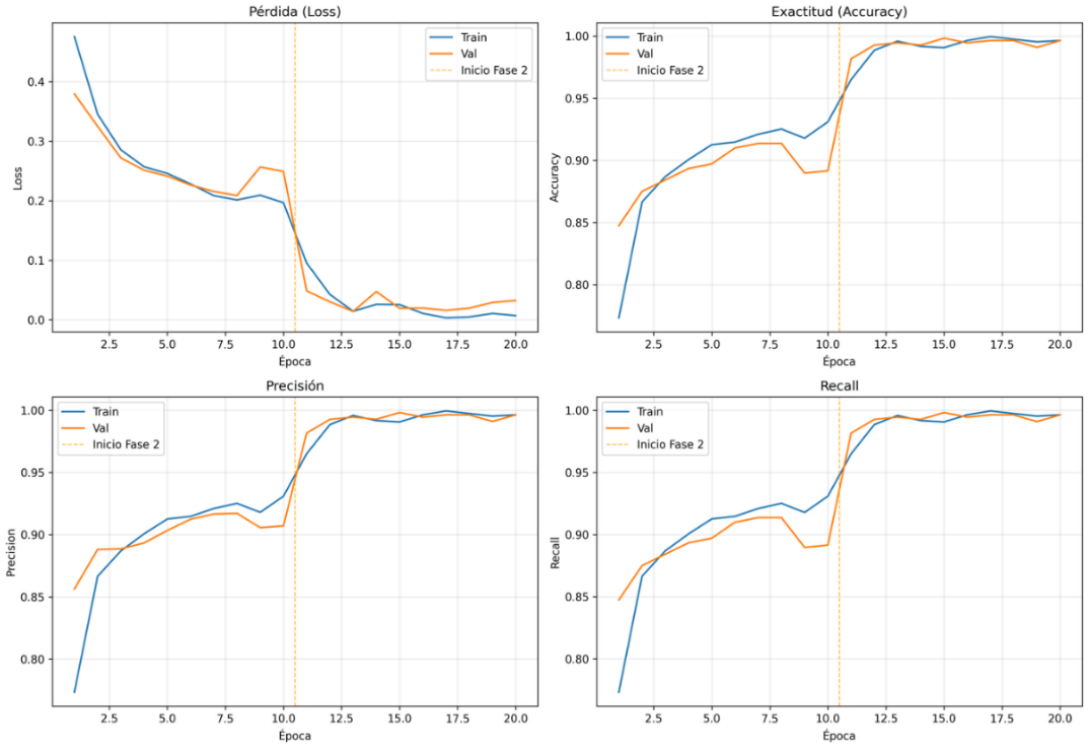
Se define como clase positiva "saludable" y como clase negativa "defectuoso". Así, un Falso Positivo (FP) es un rocoto defectuoso clasificado como saludable, y un Falso Negativo (FN) es un rocoto saludable clasificado como defectuoso.

4.2.5.1. Resultados del modelo ResNet18

ResNet18 es una red neuronal residual de 18 capas propuesta por He et al. (2016), con conexiones residuales (skip connections) que mitigan el desvanecimiento del gradiente. Se empleó fine-tuning por fases: Fase 1 entrenando solo la capa clasificadora con backbone congelado (10 épocas reales, patience = 3, mejor época = 7) y Fase 2 con fine-tuning completo a lr = 0.0001 (10 épocas reales, patience = 5, mejor época global = 15). El total de épocas reales fue 20 sobre las 50 máximas configuradas.

Figura 36:

Curvas de evolución de sus propiedades del modelo ResNet18

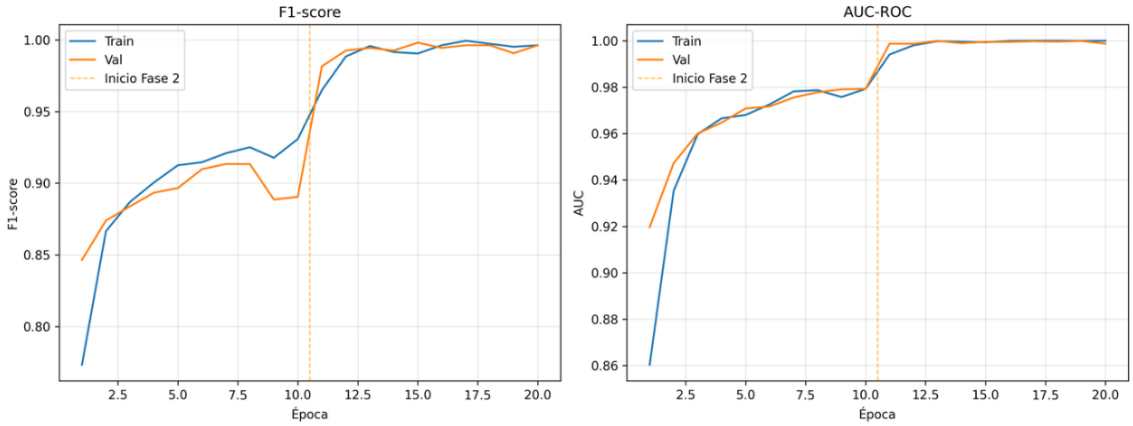


Nota. Elaboración propia (2026).

La Figura 36 muestra el entrenamiento bifásico de ResNet18 a través de cuatro métricas (pérdida, exactitud, precisión y recall). En la Fase 1 (épocas 1-10, backbone congelado), el train_loss cae de 0.48 a 0.20 y el val_loss de 0.38 a 0.216, con un repunte puntual de ambas curvas a ~0.25-0.26 en la época 9 antes de retomar la caída. En paralelo, train y val accuracy, precision y recall crecen de forma casi paralela, de ~0.77-0.85 (época 1) a ~0.92-0.91 (época 8), con la misma caída leve a ~0.89-0.92 en las épocas 9-10, evidenciando que ambos conjuntos responden de forma sincronizada sin señales de sobreajuste en esta fase. Al iniciar la Fase 2 (época 10 a 11) se produce un salto abrupto: el loss cae de ~0.15 a ~0.05 y seguidamente a menos de 0.02 hacia la época 13, mientras accuracy, precision y recall saltan de ~0.93 a ~0.98 y superan 0.99 desde la época 13, manteniéndose estables hasta el cierre (época 20) con oscilaciones mínimas (una fluctuación leve del val_loss a ~0.05 en la época 14, sin impacto en las métricas de clasificación). A diferencia de EfficientNetB0, cuyo val_loss dibuja una "U" de sobreajuste en la Fase 2, ResNet18 mantiene train y validación prácticamente superpuestos durante toda la Fase 2, lo que confirma una convergencia limpia, estable y sin brecha de generalización relevante.

Figura 37:

Evolución del F1-score y AUC del modelo ResNet18



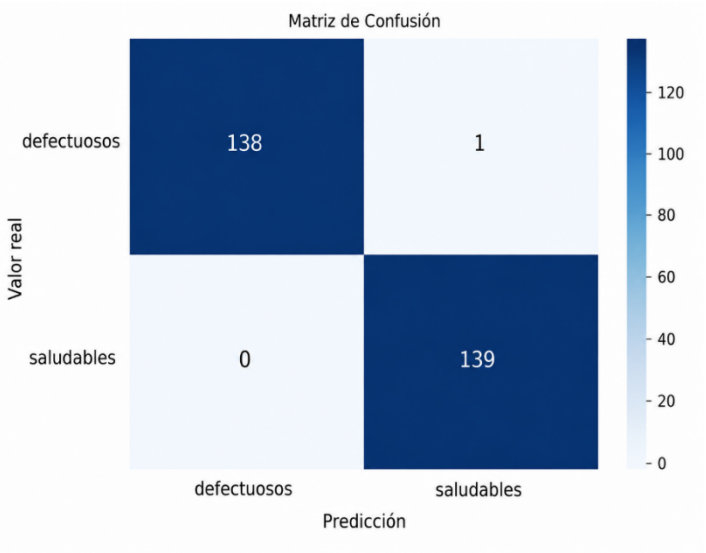
Nota. Elaboración propia (2025).

La Figura 37 confirma el mismo patrón bifásico de la Figura 36. En la Fase 1, el F1 sube de 0.77-0.847 a ~0.92 (train/val), con un retroceso puntual en la época 9; el AUC, en cambio, ya alcanza ~0.977 en ambos conjuntos sin sufrir esa caída. Al iniciar la Fase 2, F1 y AUC saltan a ~0.98 y ~0.999 respectivamente, estabilizándose cerca del

máximo desde la época 13. Esta estabilidad contrasta con EfficientNetB0, cuyo AUC se mantiene alto (~0.998) pero cuyo F1 sigue oscilando (0.937-0.967) sin consolidarse.

Figura 38:

Matriz de confusión del modelo ResNet18 — conjunto de prueba (n = 278)

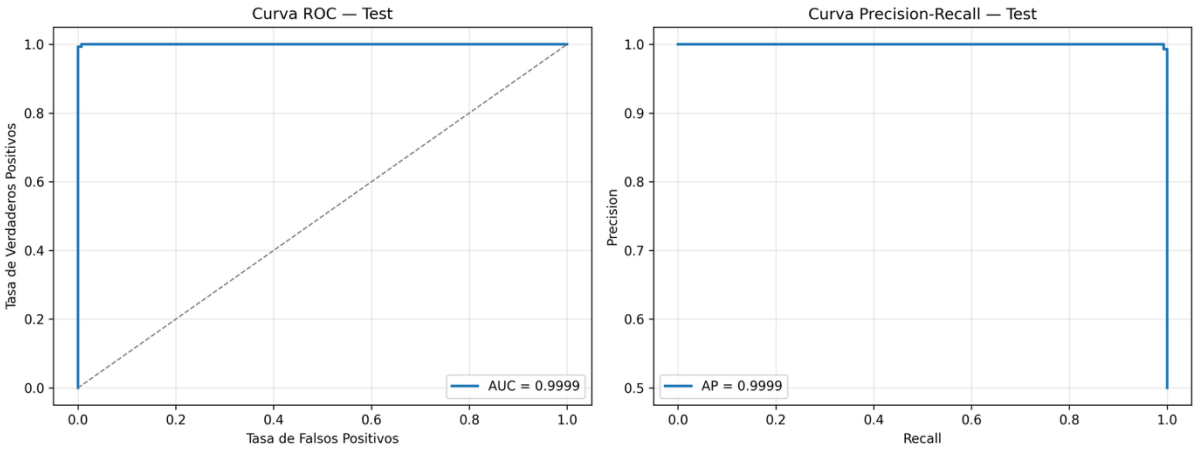


Nota. Elaboración propia (2026).

La matriz de confusión (Figura 38) sobre el conjunto de prueba de 278 muestras (139 defectuosos + 139 saludables) muestra: 139 verdaderos positivos (100%), 0 falsos negativos, 1 falso positivo (0.72%) y 138 verdaderos negativos (99.28%). El único error registrado fue un falso positivo — un rocoto defectuoso clasificado como saludable —, resultado que, si bien no es ideal desde la perspectiva sanitaria, resulta favorable en el balance general del sistema de control de calidad: es preferible que ocasionalmente un rocoto defectuoso pase como saludable (FP=1) a que rocotos saludables sean rechazados erróneamente (FN=0), ya que esto último representa pérdida económica directa para el productor. La ausencia total de falsos negativos (0.00%) confirma que el modelo nunca descartó un rocoto saludable, lo cual es el resultado más relevante de esta matriz para el objetivo del sistema.

Figura 39:

Curva ROC y curva precisión-recall del modelo ResNet18



Nota. Elaboración propia (2026). AUC = 0.9999 y AP = 0.9999 sobre el conjunto de prueba (n = 278).

La curva ROC muestra un ascenso casi vertical desde el origen hasta TPR = 1.0 con FPR prácticamente en 0, confirmando un AUC = 0.9999, resultado prácticamente óptimo y muy cercano al rectángulo ideal. La curva de precisión-recall mantiene precisión = 1.0 en casi todo el rango de recall, con una caída puntual solo en el extremo (recall ≈ 1.0, precisión ≈ 0.5), consistente con AP = 0.9999 y con el único error del modelo.

Tabla 16:

Métricas de evaluación del modelo ResNet18 — conjunto de prueba (n = 278)

Clase	Precision	Recall	F1-score	Support
defectuosos	1.0000	0.9928	0.9964	139
saludables	0.9929	1.0000	0.9964	139
accuracy			0.9964	278
macro avg	0.9964	0.9964	0.9964	278
weighted avg	0.9964	0.9964	0.9964	278

Nota. mejor resultado entre los dos modelos evaluados. Elaboración propia (2025)

La Tabla 16 presenta el reporte completo de clasificación. ResNet18 alcanza accuracy = 99.64%, F1-score ponderado = 0.9964 y AUC-ROC = 0.9999. La precisión de 1.0000 para la clase defectuosos confirma FN = 0: ningún rocoto saludable fue clasificado como defectuoso. El recall de 0.9928 refleja el único error del sistema (FP

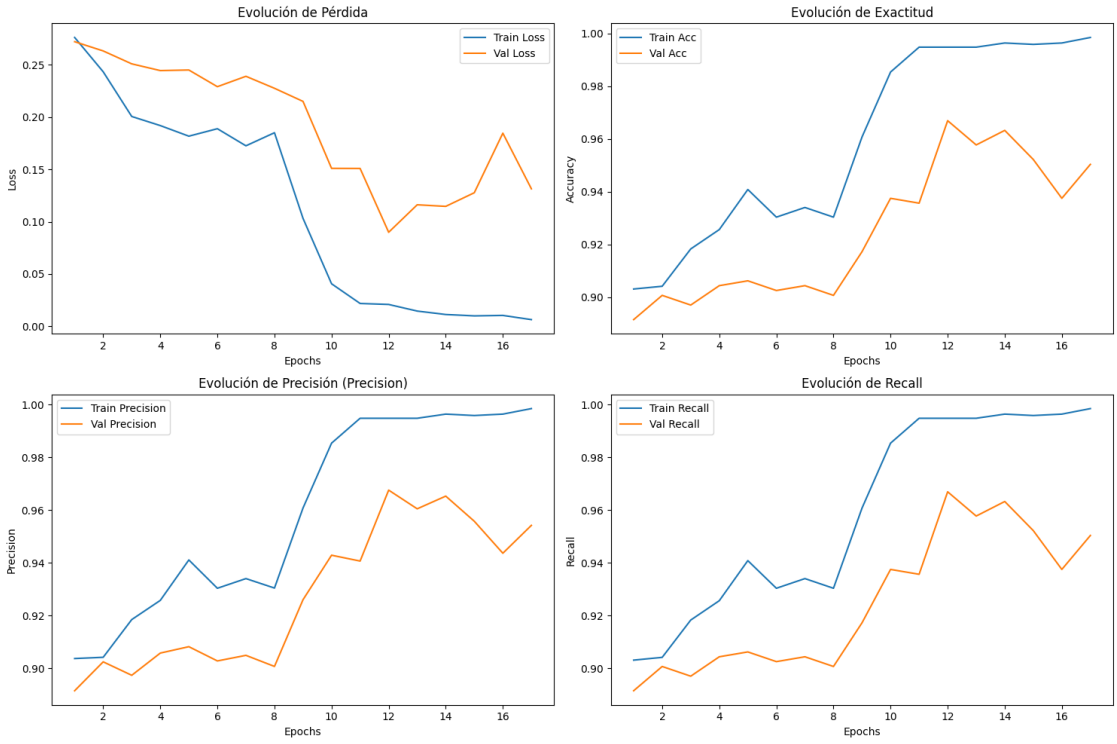
= 1, 0.72%): un rocoto defectuoso fue clasificado erróneamente como saludable. La clase saludables presenta el patrón inverso —precisión 0.9929, recall perfecto de 1.0000—, confirmando que ningún rocoto saludable fue rechazado indebidamente. La coincidencia entre macro avg y weighted avg (0.9964) confirma un desempeño equilibrado entre ambas clases. Este resultado es consistente con el AUC-ROC de 0.9999 obtenido en la Figura 39, confirmando una discriminación prácticamente perfecta con un único error sobre 278 muestras evaluadas.

4.2.5.2. Resultados del modelo EfficientNetB0

EfficientNetB0 es la arquitectura base de la familia EfficientNet propuesta por Tan y Le (2019), que aplica el principio de escalado compuesto optimizando simultáneamente profundidad, ancho y resolución de la red, empleando bloques MBConv con squeeze-and-excitation para mayor eficiencia paramétrica. Se presenta la mejor variante: EfficientNet_4, con Fase 1 de 8 épocas (solo cabeza de clasificación).

Figura 40:

Curvas de evolución las propiedades del modelo EfficientNetB0



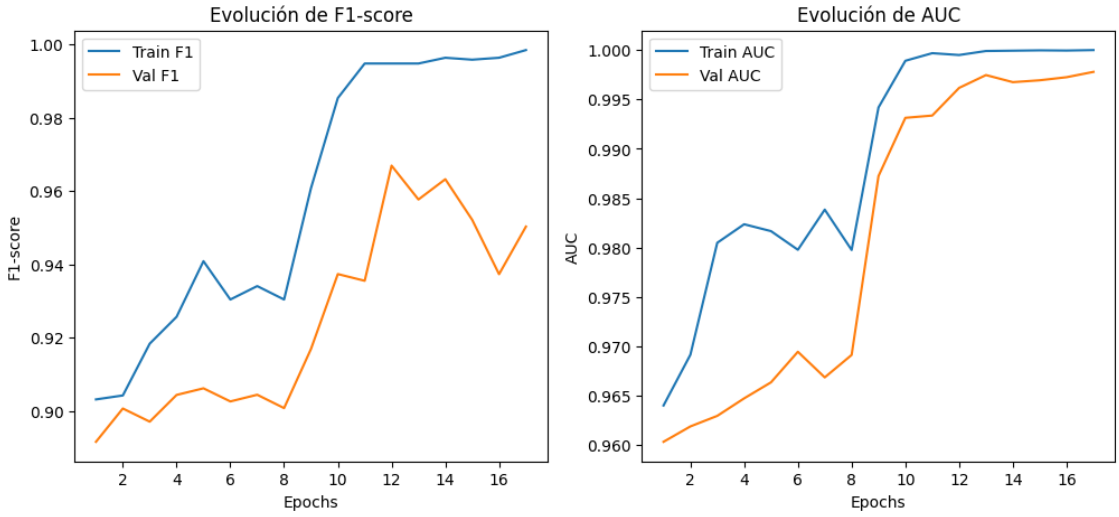
Nota. Elaboración propia (2025). Fase 1: 8 épocas (cabeza). Fase 2: 20 épocas.

La Figura 40 muestra el comportamiento bifásico de EfficientNetB0. En la Fase 1 (épocas 1-8, backbone congelado), el train_loss baja de 0.275 a 0.182 y repunta a 0.185; accuracy, precision y recall de entrenamiento suben de ~0.903 a un pico de 0.941 (época 5) y retroceden a ~0.930, mostrando que la cabeza de clasificación por sí sola alcanza rápido un techo. En validación, las mismas métricas se mantienen planas entre 0.89 y 0.906, sin mejora clara.

Al iniciar la Fase 2 (fine-tuning completo) hay un salto abrupto: el train_loss cae a 0.04 y luego a menos de 0.01, mientras accuracy, precision y recall de entrenamiento suben a ~0.985 y se estabilizan sobre 0.995. En validación, en cambio, no hay la misma estabilidad: el val_loss baja a 0.09 (época 12) pero sube hasta 0.185 (época 16) antes de bajar a 0.13 (época 17), un patrón en "U" característico de sobreajuste. Val_accuracy, precision y recall siguen la misma curva: pico de ~0.967 en la época 12, valle de ~0.94 en la 16, recuperación parcial a ~0.95 en la 17. A diferencia de ResNet18, que converge y se estabiliza sobre 0.99 desde la época 13, EfficientNetB0 oscila sin estabilizarse, lo que sugiere sensibilidad a un conjunto de validación reducido y la necesidad de más datos para converger de forma sostenida.

Figura 41:

Evolución del F1-score y AUC del modelo EfficientNetB0



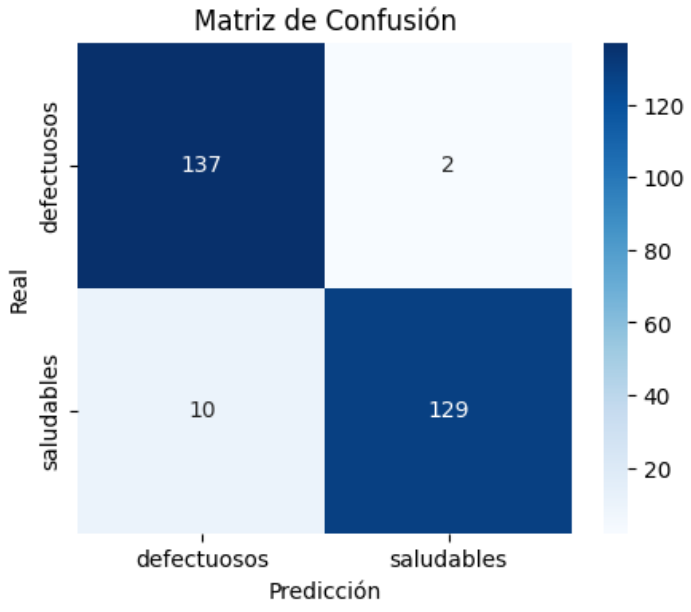
Nota. Elaboración propia (2025). AUC de validación: 0.9983 en mejor época.

La Figura 41 confirma el patrón bifásico de EfficientNetB0 ya visto en la Figura 40. En la Fase 1, el F1 de entrenamiento sube de 0.903 a un pico de 0.941 (época 5) y retrocede a 0.930 (época 8), mientras el F1 de validación se mueve en un rango más

estrecho (0.891-0.906) sin lograr superarlo. El AUC, en cambio, avanza de forma más sólida desde el inicio: sube de 0.964 a 0.984 en entrenamiento y de 0.960 a 0.969 en validación, mostrando que el modelo ya discrimina razonablemente bien entre clases aunque el F1 con umbral fijo aún no lo refleje. Al iniciar la Fase 2, ambas métricas saltan de forma abrupta: el F1 de entrenamiento alcanza 0.985 (época 11) y cierra cerca de 0.999, mientras el AUC de entrenamiento se estabiliza en 0.999-1.000. En validación, el F1 alcanza su mejor punto en la época 12 (0.967), pero luego cae a un valle de 0.937 (época 16) antes de recuperarse a 0.950 al cierre, mientras el AUC de validación sí logra estabilidad, con un máximo de 0.998 (época 13) y manteniéndose entre 0.997-0.998 hasta el final. Esta divergencia —AUC de validación estable frente a F1 de validación oscilante— es consistente con los 2 falsos positivos y 10 falsos negativos observados en la matriz de confusión: el modelo ordena bien las probabilidades, pero el umbral de decisión fijo resulta subóptimo para varios casos límite.

Figura 42:

Matriz de confusión del modelo EfficientNetB0 — conjunto de prueba (n = 278)



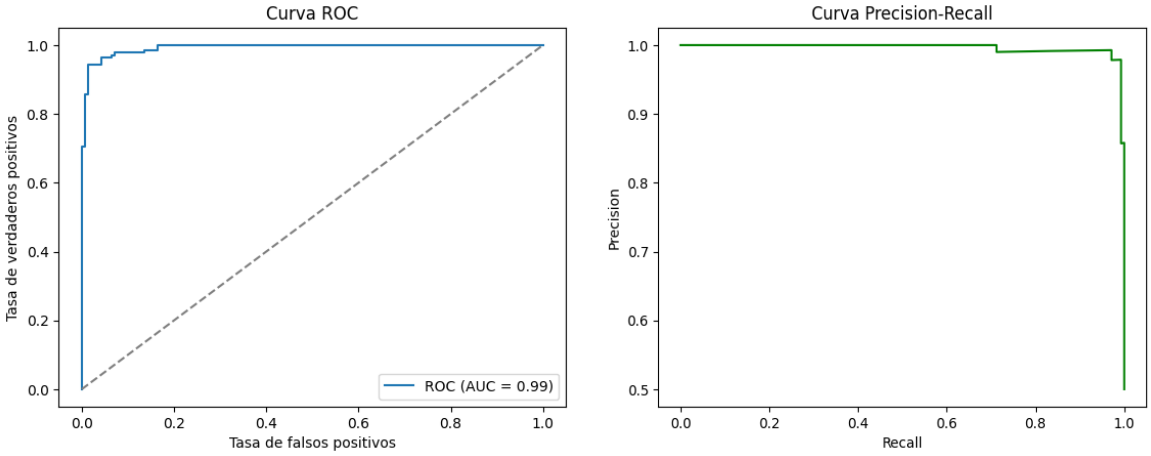
Nota. Elaboración propia (2025). Conjunto de prueba: 139 saludables + 139 defectuosos.

La matriz de confusión de EfficientNetB0 (Figura 42) sobre el conjunto de prueba muestra 137 verdaderos negativos (defectuosos correctos), 2 falsos positivos

(defectuoso clasificado como saludable), 129 verdaderos positivos (saludables correctos) y 10 falsos negativos (saludable clasificado como defectuoso). Los 10 falsos negativos representan el 7.19% de los rocotos saludables incorrectamente rechazados, lo que en un sistema de control de calidad implica pérdidas económicas directas al productor por descarte de producto apto. Los 2 falsos positivos, aunque menos frecuentes (1.44%), son más críticos desde la perspectiva sanitaria, pues implican que rocotos defectuosos pasaron como saludables.

Figura 43:

Curva ROC y precisión-recall del modelo EfficientNetB0



Nota. Elaboración propia (2025). AUC = 0.99 sobre el conjunto de prueba.

La curva ROC reporta AUC = 0.99 sobre el conjunto de prueba. Se observa que la curva no es perfectamente rectangular: hay un salto inicial a TPR ≈ 0.70 con FPR = 0 antes de alcanzar TPR = 1.0, lo que refleja que a los umbrales de confianza más altos el modelo clasifica con mayor cautela, consistente con los 10 falsos negativos observados en la matriz de confusión.

La curva PR muestra que EfficientNetB0 mantiene precisión = 1.0 para valores de recall entre 0.0 y ~0.7, con una caída leve a partir de ese punto y una caída más pronunciada cerca de recall = 1.0, hasta un mínimo de ~0.5. El área bajo la curva es alta, pero esta caída más temprana que la de ResNet18 es coherente con la mayor tasa de falsos negativos del modelo.

Tabla 17:

Métricas de evaluación del modelo EfficientNetB0 — conjunto de prueba (n = 278)

	precision	recall	f1-score	support
defectuosos	0.93	0.99	0.96	139
saludables	0.98	0.93	0.96	139
Accuracy			0.96	278
macro avg	0.96	0.96	0.96	278
weighted avg	0.96	0.96	0.96	278

Nota. Elaboración propia (2025).

La Tabla 17 presenta el reporte completo de clasificación para EfficientNetB0. El modelo alcanza un accuracy = 95.68%, con un F1-score ponderado de 0.96. La clase "defectuosos" registra precisión = 0.93 y recall = 0.99, mientras que "saludables" alcanza precisión = 0.98 y recall = 0.93. Estos valores son consistentes con la matriz de confusión: los 2 falsos positivos (defectuoso clasificado como saludable) explican la precisión imperfecta de la clase saludables, mientras que los 10 falsos negativos (saludable clasificado como defectuoso) explican su recall de 0.93. En contraste, la clase "defectuosos" concentra casi todos sus aciertos, con solo 2 errores sobre 139 casos. El F1-score de 0.96 en ambas clases y el balance entre macro avg y weighted avg (0.96 en todos los indicadores) reflejan un desempeño consistente y sin sesgo relevante entre clases, pese a la ligera diferencia entre los dos tipos de error.

4.2.5.3. Análisis comparativo de los modelos de clasificación

Tras la fase de entrenamiento, evaluación y comparación de las dos arquitecturas de clasificación, ResNet18 fue seleccionado como modelo de clasificación del sistema ROCOBOT al obtener la mayor accuracy (99.64% vs 95.68%), superando a EfficientNetB0 también en F1-score macro (99.64% vs 95.67%), AUC (0.9999 vs 0.9983) y número de falsos positivos (0 vs 2), tal como se muestra en la Tabla 18.

Tabla 18:

Comparación directa ResNet18 vs. EfficientNetB0

Indicador	ResNet18	EfficientNetB0
Verdaderos negativos (VN)	138	137
Verdaderos positivos (VP)	139	129
Falsos negativos (FN)	0 (0.00%)	10 (7.19%)
Falsos positivos (FP)	1 (0.72%)	2(1.44%)
Accuracy	99.64%	95.68%
Precisión — defectuosos	100%	93.00%
Recall — defectuosos	99.28%	99.00%
Precisión — saludables	99.29%	98.00%
Recall — saludables	100.00%	93.00%
F1-Score macro	99.64%	95.67%
AUC	99.99%	99.83%
Épocas totales	20 (10+10)	17 (8+9)

Nota. Modelo seleccionado. Elaboración propia (2026).

ResNet18 es seleccionado como modelo de clasificación del sistema ROCOBOT por obtener la mayor accuracy (99.64% vs 95.68%), mayor F1-score macro (99.64% vs 95.67%) y mayor AUC (0.9999 vs 0.9983). La diferencia más relevante entre ambos modelos no está en el desempeño global, sino en la naturaleza de sus errores**:** ResNet18 comete un único falso positivo (0.72%) y ningún falso negativo**,** mientras que EfficientNetB0 acumula 2 falsos positivos (1.44%) y 10 falsos negativos (7.19%). En términos prácticos, esto significa que EfficientNetB0 rechaza indebidamente 1 de cada 14 rocotos saludables (FN = 10)**,** con el consecuente impacto económico para el productor por descarte de producto apto. Asimismo**,** ambos modelos cometen falsos positivos —rocotos defectuosos clasificados como saludables—, pero EfficientNetB0 lo hace el doble de veces que ResNet18 (2 vs 1)**,** comprometiendo en mayor medida la calidad del lote final. A nivel de clase, ResNet18 mantiene precisión perfecta (100%) para defectuosos y recall perfecto (100%) para saludables, mientras que EfficientNetB0 no logra ese nivel en ninguna de las cuatro métricas por clase,

evidenciando una capacidad de discriminación menos consistente. En cuanto a eficiencia de entrenamiento, ResNet18 requirió 20 épocas reales (10+10) frente a las 17 de EfficientNetB0 (8+9), pero esta última mostró mayor inestabilidad en validación durante su segunda fase (ver Figura 40), lo que sugiere que su convergencia, aunque más rápida, es menos confiable que la de ResNet18. En conjunto, estos resultados —discriminación casi perfecta**,** un único falso positivo y convergencia estable— consolidan a ResNet18 como el modelo más adecuado para un sistema de clasificación de calidad orientado a comercialización.

Tabla 19:

Comparación de métricas de los modelos de clasificación vs. Antecedentes

Modelo / Referencia	Accuracy	F1-Score	AUC	Observación
Ismail & Malik (2022) — EfficientNet	99.20%	NR	NR	Manzanas/plátanos, Raspberry Pi
Dairath et al. (2023) — EfficientNet	96.70%	NR	NR	Frutos mixtos, Raspberry Pi
Hyeon et al. (2021) — ResNet50	99.60%	NR	NR	Pimientos (Capsicum), fine-tuning
ResNet18 — ROCOBOT	99.64%	99.64%	0.9999	20 épocas, FP=1 (0.72%)

Nota. Elaboración propia (2026).

ResNet18 alcanza una accuracy de 99.64%, resultado prácticamente equivalente al 99.6% reportado por Hyeon et al. (2021) con ResNet50 para pimientos, fruto de la misma familia botánica *Capsicum*, lo que valida que la arquitectura residual es igualmente efectiva para *Capsicum pubescens* incluso con una variante más liviana (ResNet18 vs ResNet50). Frente a Ismail y Malik (2022), quienes reportan 99.2% de accuracy con EfficientNet para manzanas y plátanos sobre Raspberry Pi, ResNet18 obtiene un resultado ligeramente superior pese a tratar un problema de clasificación más sutil (distinguir estado de salud dentro de una misma especie, no especies distintas). La diferencia es más marcada frente a Dairath et al. (2023), cuyo EfficientNetB0 sobre frutos mixtos en hardware embebido alcanzó solo 96.7%, casi 3 puntos porcentuales por debajo de ResNet18-ROCOBOT. Ninguno de los tres antecedentes reporta F1-score, AUC ni número de falsos positivos, lo que limita la comparación a la accuracy;

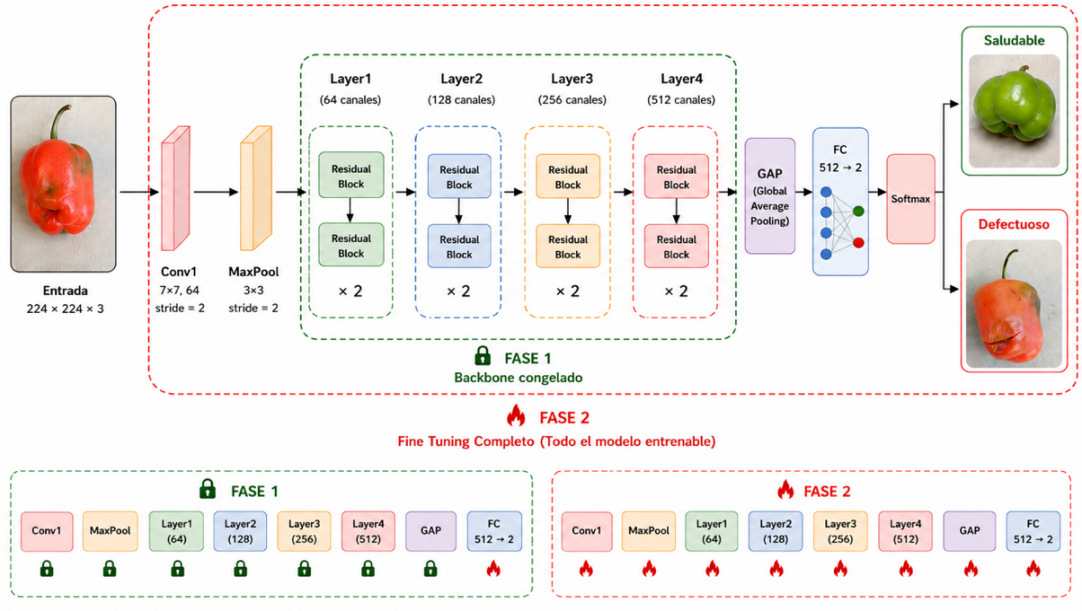
sin embargo, el FN=0 (0.00%) obtenido por ResNet18 es un resultado que, de poder contrastarse, probablemente superaría a los antecedentes revisados, dado que ningún estudio similar reporta una tasa de error nula en ninguna dirección. Esta ausencia de reporte de FP en la literatura revisada es en sí misma una limitación común en estudios de clasificación agrícola con hardware embebido, y refuerza el aporte diferenciador de ROCOBOT al reportar de forma íntegra tanto los aciertos como los dos tipos de error del sistema.

4.2.5.4. Arquitectura del modelo de clasificación seleccionado

En base a los resultados obtenidos, ResNet18 fue seleccionado como el modelo de clasificación binaria del sistema ROCOBOT. Para el entrenamiento de este modelo se empleó su arquitectura adaptada mediante transfer learning en dos fases sobre ImageNet, modificando únicamente la capa de clasificación final para el dominio del rocoto (*Capsicum pubescens*). La Figura 44 presenta el pipeline completo de la arquitectura ResNet18 con la que fue entrenado el modelo de clasificación de rocotos.

Figura 44:

Arquitectura del modelo ResNet18 para la etapa de clasificación — sistema ROCOBOT



Nota. Elaboración propia (2026).

La arquitectura ResNet18 fue adaptada en dos fases: en la Fase 1, el backbone (Conv1, MaxPool y Layer1–Layer4) fue congelado, entrenando únicamente la capa FC (512→2) durante 10 épocas con lr=1e-3, lo que permite que el modelo aprenda las

características específicas del rocoto sin alterar los pesos preentrenados en ImageNet. En la Fase 2, todo el modelo fue descongelado para fine-tuning completo durante 10 épocas con lr=1e-4, permitiendo que todas las capas se ajusten al dominio del *Capsicum pubescens*. Esta estrategia de dos fases garantizó una convergencia estable y evitó el sobreajuste, alcanzando val_acc = 99.82% y AUC = 0.9999 en solo 20 épocas totales, con clasificación de imágenes de entrada de 224×224×3 píxeles en dos clases: saludable y defectuoso.

4.3. Diseño, construcción e implementación del prototipo ROCOBOT

4.3.1. Selección y justificación de componentes

En la revisión de los antecedentes de la literatura se encontraron diversos estudios que implementan prototipos físicos para la clasificación automatizada de frutos y hortalizas. La Tabla 18 presenta los antecedentes identificados, indicando el hardware de procesamiento, el sistema de transporte y la cámara empleada en cada caso.

Tabla 18:

Componentes de hardware empleados en prototipos físicos de antecedentes

N°	Autor/Año	Producto	Procesador	Sistema de transporte	Cámara
1	Mohi-Alden et al. (2023)	Pimiento morrón	PC industrial con GPU	Banda de inspección	Cámara industrial
2	Dairath et al. (2023)	Manzana/Plátano	Raspberry Pi	No reportado	Cámara industrial RGB-D
3	Ismail & Malik (2022)	Manzana/Plátano	Raspberry Pi	No reportado	Cámara RPi estándar
4	Sanwar & Ahmed (2023)	Objetos genéricos	Arduino + PC	Banda con gripper robótico	No reportado
5	Kamboj & Diwan (2019)	Objetos genéricos	PLC	Faja transportadora	Visión tradicional (sin IA)

Nota. Elaboración propia (2025), basada en estudios revisados.

A diferencia de los modelos de IA, no se identificó en la literatura un prototipo robótico específico para la clasificación de rocotos (*Capsicum pubescens*). Del análisis de la Tabla 18 se desprende que ningún antecedente integra simultáneamente procesamiento embebido de bajo costo, una faja sincronizada con el pipeline de IA y una cámara con iluminación controlada. Por ello, el diseño de ROCOBOT se fundamentó en combinar las características de

mejor desempeño de cada antecedente, adaptándolas a las condiciones del rocoto y al presupuesto de un prototipo académico.

Mohi-Alden et al. (2023) emplearon una PC industrial con GPU para el procesamiento de su sistema de clasificación de pimientos, mientras que Dairath et al. (2023) utilizaron cámaras industriales RGB-D de alto costo para su sistema de clasificación de frutos. Por su parte, Kamboj y Diwan (2019) y Sanwar y Ahmed (2023) implementaron sistemas de banda transportadora controlados mediante PLC o Arduino, sin sincronización directa entre la faja y un pipeline de inteligencia artificial. Ismail y Malik (2022) y Dairath et al. (2023) sí emplearon Raspberry Pi como unidad de procesamiento, validando su viabilidad para sistemas de clasificación de frutos con modelos de deep learning, aunque ninguno de ellos integró una faja transportadora propia sincronizada con el modelo.

En base a este análisis, se decidió personalizar el prototipo ROCOBOT integrando: la unidad de procesamiento Raspberry Pi 4B, validada como suficiente por Ismail y Malik (2022) y Dairath et al. (2023) para ejecutar modelos de deep learning en clasificación de frutos; una faja transportadora con rodillos sincronizada directamente con el pipeline de detección y clasificación**,** cuyo diámetro y separación se dimensionaron para un rango de fruto de 4.0 a 8.4 cm, acorde con los valores de diámetro reportados en estudios agronómicos peruanos sobre la especie (Huacachino Zavala & Ripa Huacachino, 2023)**;** una cámara embebida de bajo costo (RPi Module 3 Wide, 12 MP); y una caja de inspección con iluminación LED controlada, que elimina la variabilidad de iluminación ambiental señalada como una limitación en los sistemas de visión artificial agroindustrial.

4.3.2. Componentes del prototipo

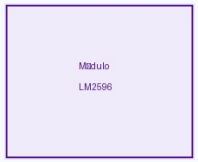
Para la construcción del sistema robótico ROCOBOT se emplearon diversos componentes electrónicos, mecánicos y estructurales que permiten su funcionamiento automatizado. Estos elementos fueron seleccionados en función de su compatibilidad, precisión y bajo consumo energético, garantizando un desempeño óptimo en el proceso de detección y clasificación del rocoto. La Tabla 19 detalla el listado completo de materiales empleados, indicando sus especificaciones técnicas, función en el sistema y la justificación de uso.

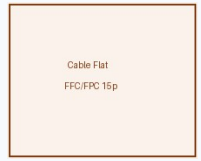
Tabla 20:

Listado de materiales para ROCOBOT

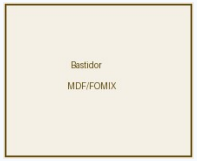
Componente	Especificaciones técnicas	Función en el sistema	Justificación de uso	Imagen
Raspberry Pi 4 Model B (4 GB)	• Procesador Broadcom BCM2711 Quad-core Cortex-A72 @ 1.5 GHz. • Memoria LPDDR4-3200 SDRAM de 4 GB. • Wi-Fi 802.11ac dual-band 2.4/5 GHz, Bluetooth 5.0 BLE. • Gigabit Ethernet. 2 × USB 3.0 y 2 × USB 2.0. • 40 GPIO. Doble salida micro-HDMI (hasta 4K). • Alimentación USB-C 5 V / 3 A.	Ejecutar los algoritmos de visión artificial, procesamiento de imágenes, control de sensores y actuadores del prototipo ROCOBOT.	Alto rendimiento computacional; compatibilidad con OpenCV, TensorFlow y PyTorch; integración directa con sensores, cámaras y servomotores.	
Nota oficial USB-C Raspberry Pi	• Salida: 5.1 V / 3 A. • Conector USB-C. • Protección contra sobrecorriente. • Alimentación estable para Raspberry Pi 4B.	Alimentar eléctricamente la unidad de procesamiento principal.	Garantiza estabilidad energética evitando reinicios y fallos por caída de voltaje.	

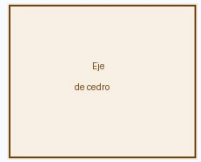
Tarjeta microSD SanDisk 32 GB C10	• Capacidad: 32 GB. • Clase 10 UHS-I. • Alta velocidad de lectura/escritura. • Compatible con Raspberry Pi OS.	Almacenar el sistema operativo, librerías, datasets y modelos entrenados de inteligencia artificial.	Velocidad y estabilidad suficientes para el procesamiento IA en tiempo real sobre Raspberry Pi 4B.	
SWITCH	• Voltaje nominal: 3.7 V. • Tecnología ion-litio recargable. • Alta densidad energética.	Alimentar energéticamente el sistema robótico.	Proporcionan autonomía y portabilidad al prototipo durante operación sin conexión a red.	
Módulo cargador/protección	• Protección contra sobrecarga y cortocircuitos. • Gestión segura de carga. • Indicador LED de estado de carga.	Gestionar la carga segura de baterías del sistema.	Incrementa la seguridad eléctrica y prolonga la vida útil de las baterías.	
Adaptador DC	• Nota regulada de alimentación externa. • Salida estable de voltaje DC. • Compatible con múltiples voltajes de salida.	Alimentar módulos y motores durante pruebas en banco.	Garantiza suministro eléctrico estable en operación continua de laboratorio.	

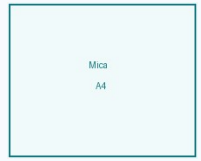
Módulo regulador LM2596 DC-DC	• Conversor DC-DC Step-Down. • Entrada: 4 V – 40 V DC. • Salida ajustable: 1.25 V – 35 V DC. • Corriente máxima: 3 A. • Eficiencia: hasta 92%. Protección térmica integrada.	Regular y estabilizar el voltaje de alimentación para los módulos electrónicos del sistema.	Permite obtener voltajes estables para Raspberry Pi, sensores y actuadores, evitando fallos por sobrevoltaje o fluctuaciones.	
Cables jumpers MM / MH / HH	• Tipos: macho-macho, macho-hembra y hembra-hembra. • Longitudes: 10 – 20 cm. • Colores diferenciados para identificación. • Compatibles con protoboard y pines GPIO.	Interconectar sensores, actuadores y módulos electrónicos.	Simplifican el ensamblaje modular y permiten reconfiguraciones rápidas del circuito.	
Cable mellizo 1 m	• Cable eléctrico de dos conductores paralelos. • Longitud: 1 metro. • Aislamiento de PVC flexible.	Conducir corriente eléctrica entre fuente de alimentación y módulos del sistema.	Cable de uso general de bajo costo para interconexión de componentes de potencia.	
Enchufe plano	• Conector de alimentación tipo plano. • Compatible con tomacorriente estándar. • Material PVC con pines metálicos.	Conectar el sistema a la red eléctrica durante operación continua.	Permite la alimentación directa desde red eléctrica para pruebas prolongadas en laboratorio.	

Sensor infrarrojo FC-51	• Voltaje de operación: 3.3 V – 5 V. • Distancia de detección: 2 – 30 cm (ajustable). • Salida digital TTL. • Potenciómetro de sensibilidad ajustable.	Detectar presencia de rocotos sobre la banda transportadora en los puntos de clasificación.	Facilita la automatización y sincronización del sistema de visión artificial con el movimiento de la faja.	
Cámara RPi Module 3 Wide 12 MP	• Sensor Sony IMX708 de 12 MP. • Autofocus motorizado integrado. • Campo de visión: 120° diagonal. • Apertura F2.2. Resolución Full HD. • Interfaz CSI. Compatible con libcamera.	Capturar imágenes de los rocotos para detección y clasificación automática mediante IA.	Alta calidad de captura y compatibilidad directa con Raspberry Pi; campo de visión amplio ideal para la zona de inspección.	
Cable Flat FFC/FPC 15 pines	• Cable flexible CSI de 15 pines. • Longitud: ~1 metro. • Alta flexibilidad y baja pérdida de señal.	Transmitir la señal de imagen desde la cámara hacia la Raspberry Pi.	Permite instalar la cámara en diferentes posiciones del prototipo con mínima interferencia de señal.	
Cinta LED Neon / Tira LED	• Voltaje de operación: 3 V – 3.3 V. • Corriente típica: 20 mA. • Alta intensidad luminosa. • Sistema de iluminación homogénea. • Bajo consumo energético.	Iluminar el área de captura de imágenes del módulo de visión artificial.	Reduce sombras y errores causados por iluminación variable; mejora la calidad y estabilidad de las	

			imágenes para el modelo IA.	
Servomotor MG995 (TowerPro)	• Voltaje: 4.8 V – 6.0 V. • Torque máximo: 9.4 kg·cm a 4.8 V. • Velocidad: 0.2 s / 60°. • Ángulo de giro: 0° – 180°. • Peso: 55 g. Engranajes metálicos.	Accionar las compuertas de clasificación que desvían los rocotos hacia el canal saludable o defectuoso.	Control angular preciso mediante PWM, alto torque y bajo consumo; adecuado para mecanismos de clasificación repetitiva.	
Motorreductor DC JGB37-520 (12 V)	• Voltaje nominal: 12 V DC. • Velocidad: 30 – 100 RPM. • Torque: 1.5 – 3 kg·cm. • Corriente: 100 – 300 mA. • Caja reductora metálica integrada.	Accionar la faja transportadora con rodillos generando el movimiento continuo del sistema.	Tamaño compacto, torque suficiente para mover cargas ligeras de forma continua y controlada a baja velocidad.	
Driver puente H L298N	• Voltaje lógico: 5 V. • Voltaje de motores: 5 V – 35 V. • Corriente máxima: 2 A por canal. • Control PWM dual. Doble puente H. • Control bidireccional de motores DC.	Controlar velocidad y dirección de giro del motorreductor de la faja transportadora.	Permite controlar motores DC desde la Raspberry Pi mediante señales PWM de bajo voltaje.	

Cuero sintético (banda transportadora)	• Superficie flexible de caucho/PVC antideslizante. • Dimensiones: 3 m × 30 cm. • Alta resistencia al desgaste y fricción. • Movimiento continuo adaptable al prototipo.	Transportar los rocotos durante el proceso automático de detección y clasificación.	Superficie estable y continua que garantiza el desplazamiento uniforme de los frutos hacia la zona de inspección.	
Ejes metálicos	• Barras metálicas cilíndricas de soporte. • Alta resistencia mecánica. • Diámetros variables según diseño.	Mantener alineados rodillos y poleas del sistema de transmisión.	Brindan estabilidad estructural y alineación precisa durante la operación continua.	
Bastidor estructural MDF / FOMIX	• MDF de 4 mm y cartón espuma FOMIX de 6 mm. • Diseño modular desmontable. • Material rígido, liviano y de fácil mecanizado. • Soporte para banda y componentes electrónicos.	Soportar y estabilizar todos los componentes mecánicos y electrónicos del prototipo.	Material de bajo costo, rígido y ligero; adecuado para prototipado académico con corte y ensamblaje sencillos.	

Tornillos y tuercas	• Elementos metálicos de fijación. • Diferentes diámetros y longitudes. • Material de acero galvanizado.	Ensamblar los componentes mecánicos del sistema.	Garantizan firmeza estructural y permiten desmontaje para mantenimiento o ajuste.	
Tubo PVC 33 mm	• Tubo de PVC de 33 mm de diámetro. • Material resistente y ligero. • Fácil corte y mecanizado.	Actuar como rodillos de soporte de la banda transportadora.	Material de bajo costo y fácil mecanizado; adecuado para fabricar rodillos en prototipos académicos.	
Cadena de bicicleta	• Cadena metálica tipo bicicleta. • Alta resistencia mecánica. • Compatible con piñones de transmisión estándar.	Transmitir movimiento desde el motorreductor hacia los rodillos de la faja.	Transmisión de fuerza eficiente y duradera en sistemas de baja velocidad con bajo costo.	
Eje de cedro	• Madera de cedro mecanizada. • Perfil cilíndrico personalizado. • Material ligero y de bajo peso.	Actuar como eje central de los rodillos de la banda transportadora.	Material ligero, de bajo costo y fácil mecanizado; adecuado para prototipado de sistemas de transporte.	

Mica A4 (lámina acrílica)	• Lámina de acrílico transparente. • Formato A4 aproximado. • Fácil corte con cutter o láser.	Recubrir y proteger la estructura del módulo de inspección visual.	Material transparente que permite visualizar el interior del prototipo sin intervención estructural.	
Cinta aislante	• Cinta de vinilo aislante eléctrico. • Alta adherencia. • Resistencia al calor y humedad.	Aislar y proteger conexiones eléctricas del sistema.	Previene cortocircuitos y asegura la integridad de las conexiones eléctricas del prototipo.	
Estructura metálica	• Perfiles metálicos estructurales. • Alta resistencia mecánica. • Diseño modular y desmontable.	Proporcionar soporte rígido principal del sistema de transporte y clasificación.	Garantiza rigidez estructural durante el funcionamiento continuo del prototipo.	
Regla acrílica	• Regla de acrílico transparente. • Graduación milimétrica. • Material rígido ligero.	Guiar y alinear los rocotos sobre la banda transportadora.	Permite canalizar el flujo de rocotos de forma controlada hacia la zona de inspección.	

Nota. Componentes electrónicos, de visión, actuadores y mecánicos del prototipo ROCOBOT. Elaboración propia (2025).

4.3.3. Diagrama de flujo de cada módulo

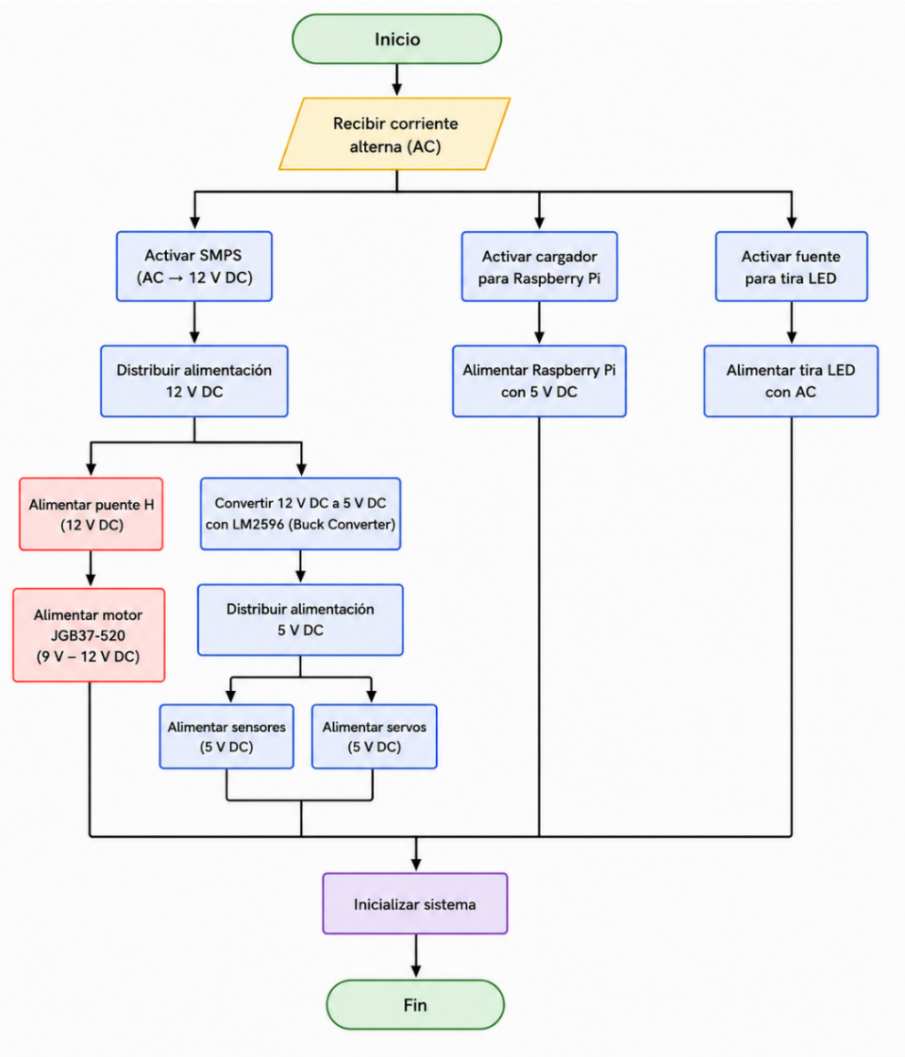
El sistema ROCOBOT se organiza en cuatro módulos funcionales, cuyo flujo de operación se ilustra en las Figuras 43 a 46.

4.3.3.1. Módulo de alimentación y regulación energética

Este módulo distribuye la energía desde la fuente SMPS DC 12V hacia los distintos componentes electrónicos del prototipo mediante los convertidores LM2596, que regulan el voltaje según los requerimientos de cada elemento. La Figura 45 describe el flujo de alimentación energética del sistema, desde la fuente SMPS DC 12V hasta la distribución regulada hacia cada módulo electrónico del prototipo.

Figura 45:

Flujo del módulo de alimentación y regulación energética



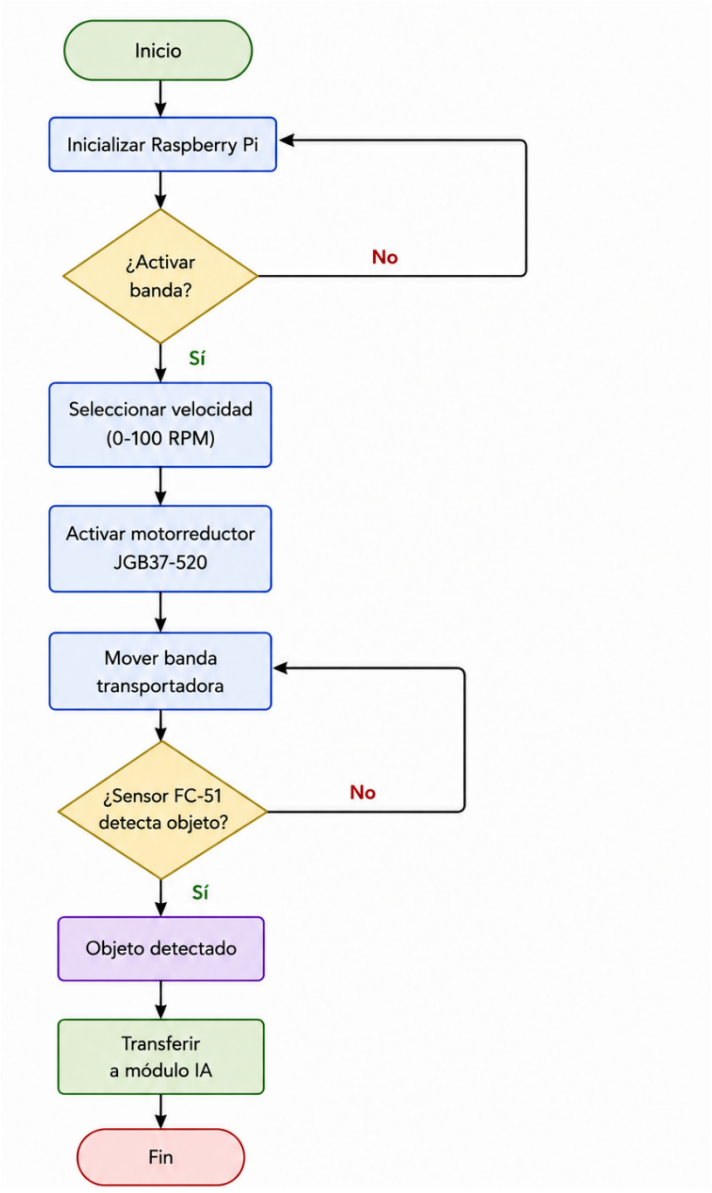
Nota. Elaboración propia (2025).

4.3.3.2. Módulo de control de la banda

El módulo de control de la faja transportadora regula el desplazamiento del motorreductor JGB37-520 mediante el driver L298N, que ajusta su velocidad y dirección de giro a partir de señales PWM enviadas por la Raspberry Pi 4B. La Figura 46 describe el flujo de control de la faja transportadora, gestionado por el driver L298N que regula la velocidad y dirección del motorreductor JGB37-520 mediante señales PWM provenientes de la Raspberry Pi 4B.

Figura 46:

Flujo del módulo de banda transportadora



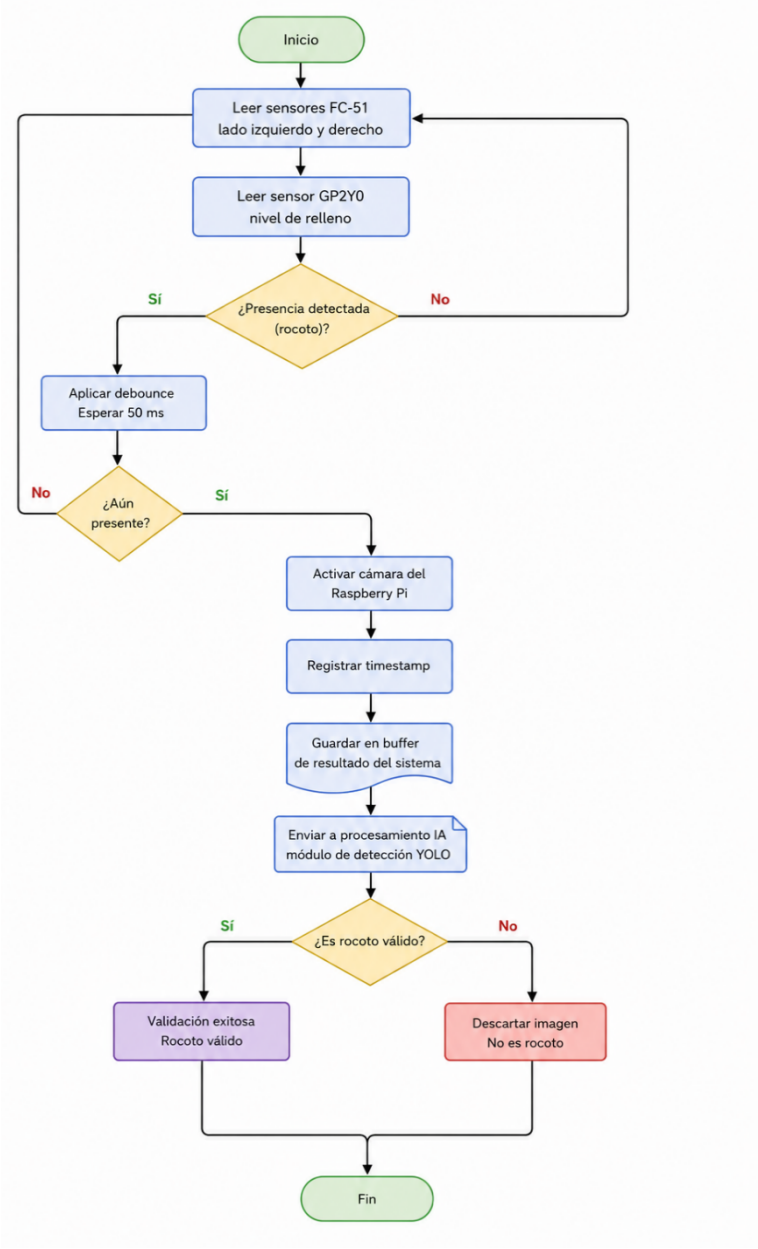
Nota. Elaboración propia (2025).

4.3.3.3. Módulo de detección de rocoto

El módulo de detección integra la captura de imagen, el procesamiento mediante el modelo YOLOv8n y la confirmación física de presencia del fruto. La Figura 47 describe el flujo del módulo de detección, desde la captura de la imagen por la cámara hasta la ejecución del modelo YOLOv8n y la confirmación de presencia del rocoto mediante los sensores infrarrojos FC-51.

Figura 47:

Flujo del módulo de detección



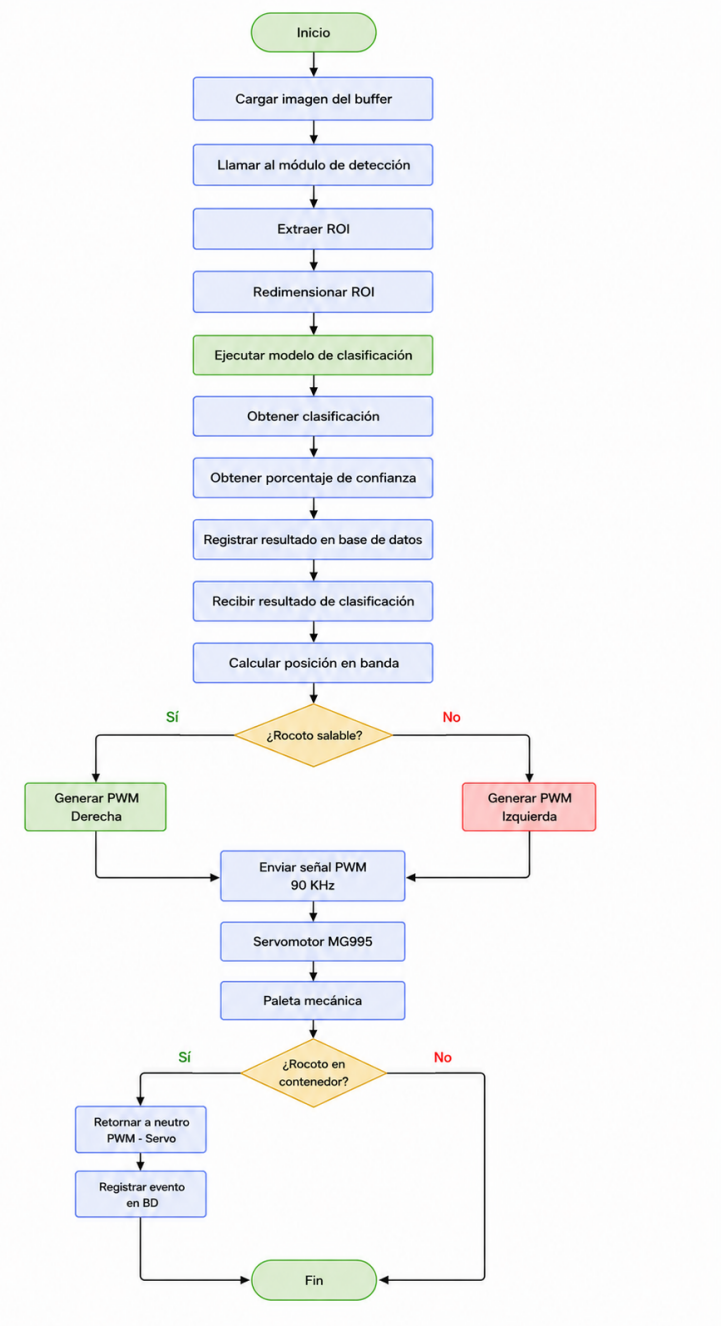
Nota: Elaboración propia (2026)

4.3.3.4. Módulo de clasificación de rocoto

La Figura 48 describe el flujo del módulo de clasificación, desde la inferencia del modelo ResNet18 hasta el accionamiento de los servomotores MG995 que desvían el rocoto hacia el canal correspondiente según su condición: saludable o defectuoso.

Figura 48:

Flujo del módulo de clasificación



Nota. Elaboración propia (2025).

4.3.4. Diagramas de conexiones eléctricas del sistema ROCOBOT

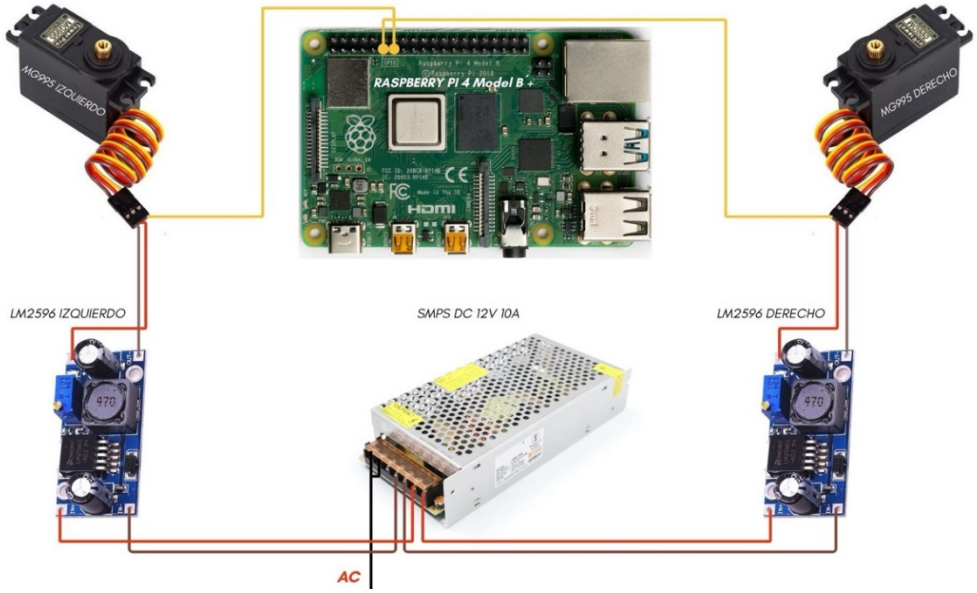
Raspberry Pi 4B como unidad de procesamiento central: el módulo de clasificación (dos servomotores MG995 + dos convertidores LM2596), el módulo de detección de posición (dos sensores FC-51 + dos convertidores LM2596), el módulo de control de la faja transportadora (motorreductor JGB37-520 + driver L298N) y el módulo de alimentación (fuente SMPS DC 12V 10A). A continuación, se describen las conexiones de cada módulo.El sistema ROCOBOT integra cuatro módulos electrónicos interconectados con la

4.3.4.1. Diagrama de conexión de servomotores MG995 y convertidores LM2596

El diagrama de clasificación del sistema ROCOBOT, compuesto por dos servomotores MG995 y dos convertidores DC-DC LM2596 independientes, uno por cada servo. El servo derecho (clasificador de rocotos saludables) recibe su señal PWM del GPIO17 (Pin 11) de la Raspberry Pi, mientras que el servo izquierdo (clasificador de rocotos defectuosos) recibe la señal del GPIO27 (Pin 13). Ambos servomotores son alimentados a 5V desde sus respectivos convertidores LM2596, cuya entrada proviene de la fuente SMPS a 12V. Esta configuración independiente garantiza que los picos de corriente del MG995 (hasta 2.5A en carga) no afecten la estabilidad de la Raspberry Pi 4B.

Figura 49:

Diagrama de conexiones de los servomotores MG995 con convertidores LM2596



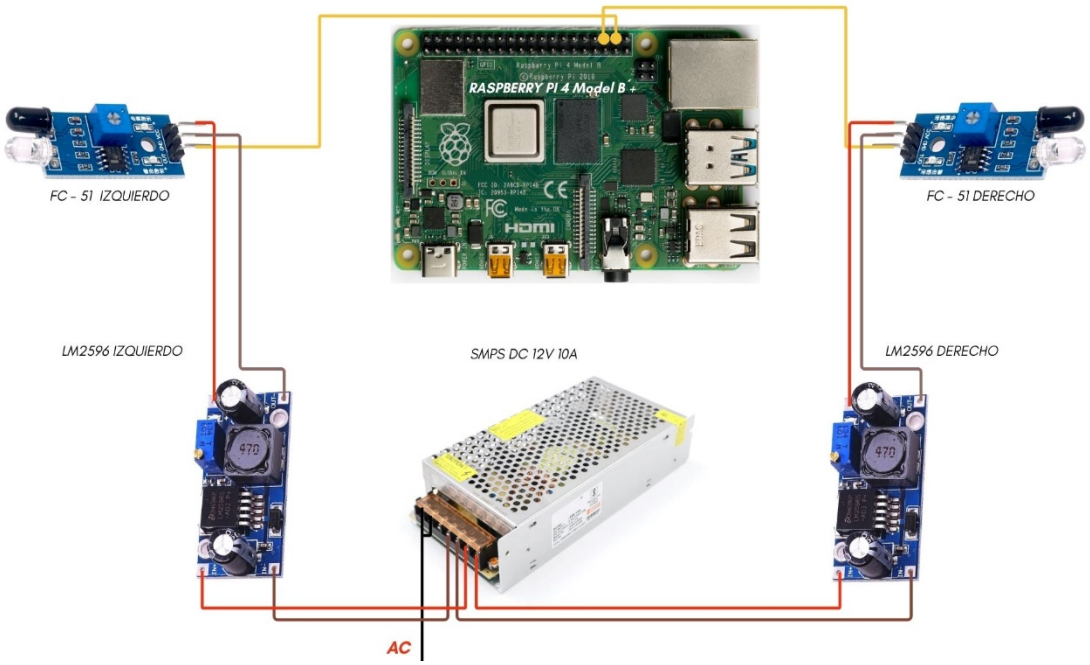
Nota. Elaboración propia (2025).

4.3.4.2. Diagrama de conexión de los sensores FC-51 y convertidores LM2596

El diagrama de detección de posición, compuesto por dos sensores infrarrojos FC-51 (izquierdo y derecho) cada uno con su propio convertidor LM2596 para alimentación a 5V. El sensor FC-51 derecho emite su señal digital D0 al GPIO16 (Pin 36) de la Raspberry Pi, mientras que el sensor FC-51 izquierdo se conecta al GPIO20 (Pin 38), permitiendo detectar la presencia del rocoto en los dos puntos de clasificación de la faja transportadora. El potenciómetro de cada sensor se ajusta para detección a aproximadamente 1 cm de distancia del objeto. La salida analógica A0 de ambos sensores no se utiliza en el sistema.

Figura 50:

Diagrama de conexiones de los sensores FC-51 y con convertidores LM2596



Nota. Elaboración propia (2025).

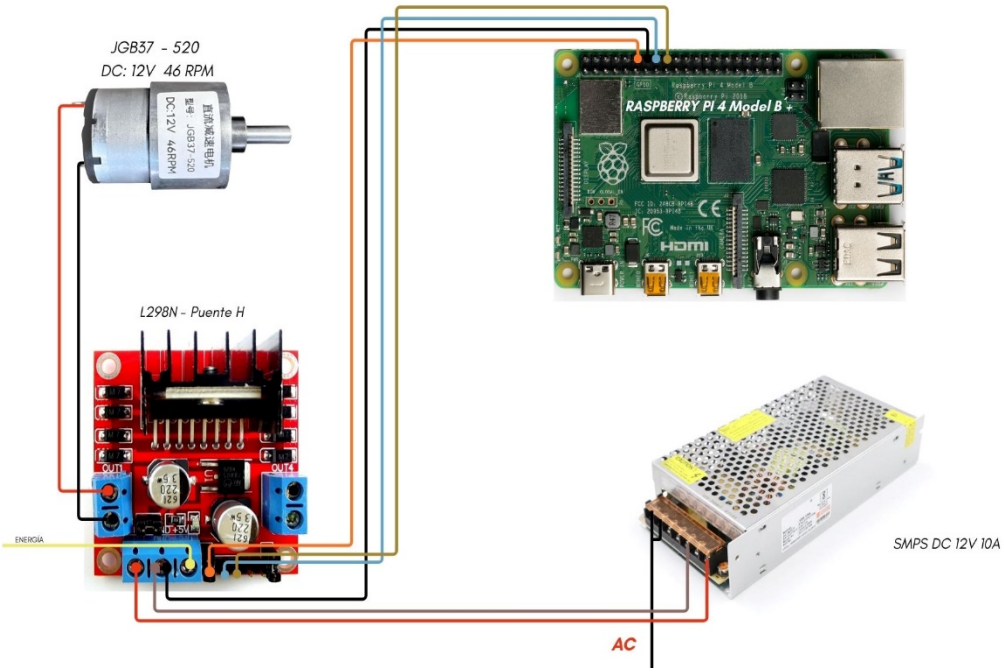
4.3.4.3. Diagrama de conexión del motoreductor JGB37-520 y driver L298N

En el diagrama de ede control de la faja transportadora, el driver L298N (puente H doble) regula la velocidad y dirección del motorreductor JGB37-520 (DC 12V, 46 RPM) mediante tres señales de control desde la Raspberry Pi 4B: el pin ENA (GPIO18) controla la velocidad mediante PWM, mientras que IN1 (GPIO23) e IN2 (GPIO24) determinan el sentido de giro. El motor recibe alimentación directamente a 12V desde la

fuente SMPS a través del L298N. El pin 5V del L298N no se conecta externamente; se habilita el regulador interno mediante el jumper del módulo.

Figura 51:

Diagrama de conexiones del motoreductor JGB37-520 DC12V y driver L298N



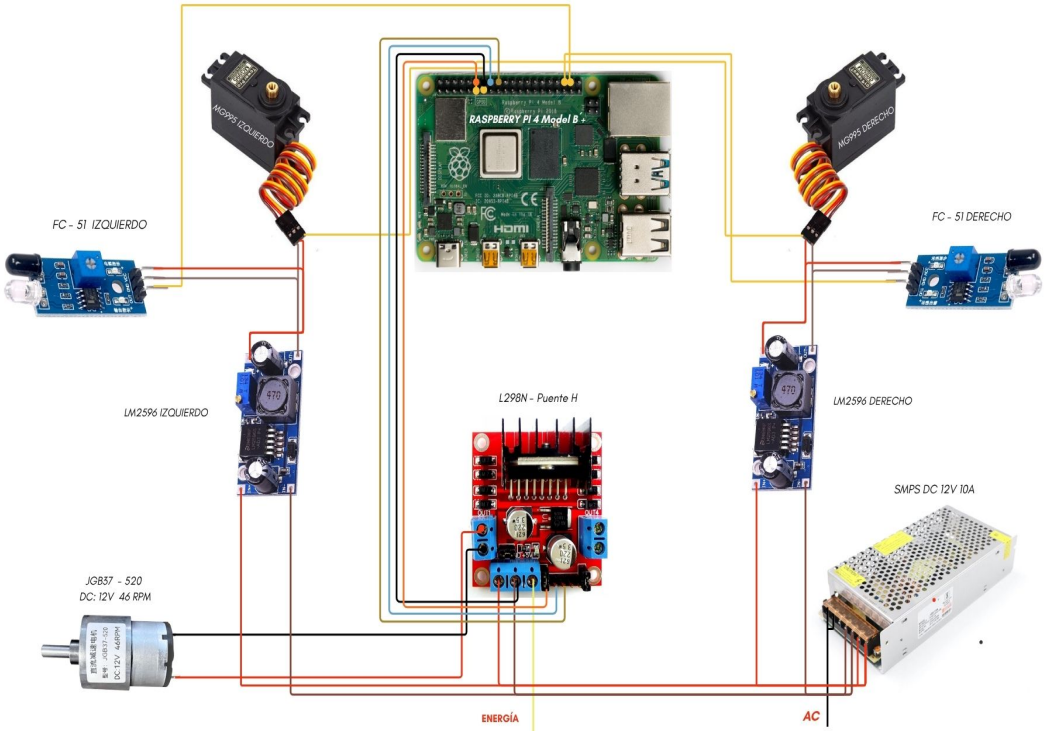
Nota. Elaboración propia (2025).

4.3.4.4. Diagrama general de conexiones del sistema

El diagrama de integración completa del sistema ROCOBOT, presenta la fuente SMPS DC 12V 10A constituye la fuente de alimentación central, distribuyendo energía a todos los módulos del sistema. Los cuatro convertidores LM2596 (dos del módulo de clasificación y dos del módulo de detección) regulan la tensión de 12V a 5V para alimentar los servomotores MG995 y los sensores FC-51, protegiéndolos de variaciones en la fuente principal y desacoplando su alimentación de la Raspberry Pi 4B. El código fuente completo del sistema embebido de control de hardware se presenta en el Anexo A.4.

Figura 52:

Diagrama general de conexiones del sistema ROCOBOT



Nota. Elaboración propia (2025).

4.3.5. Diseño de la estructura física

El prototipo ROCOBOT fue diseñado bajo el principio de modularidad, integrando tres zonas funcionales en una estructura lineal: la zona de alimentación (tolva de ingreso de rocotos), la zona de inspección (cámara de visión artificial con tira de luz LED) y la zona de clasificación (faja transportadora con rodillos y servomotores de desvío). La estructura fue fabricada en MDF de 4 mm para garantizar rigidez, ligereza y facilidad de mecanizado. A continuación, se presentan las vistas del diseño 3D, la planta 2D y la vista isométrica del prototipo. Los planos completos, incluyendo las vistas 2D acotadas y el archivo editable en AutoCAD, se presentan en el Anexo G.2.

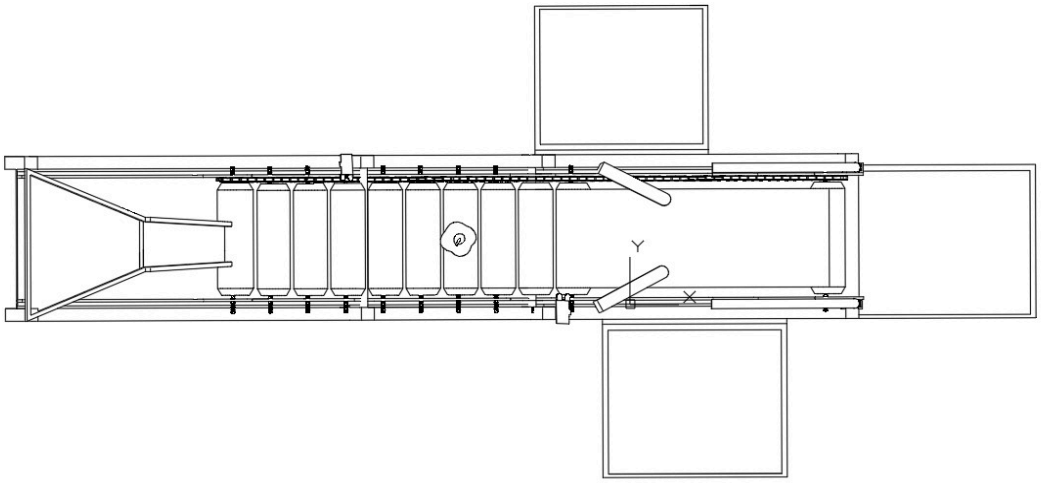
4.3.5.1. Diseño 2D Faja Transportadora

La Figura 53 muestra la vista superior del prototipo ROCOBOT en plano 2D. Se aprecia la distribución longitudinal del sistema: la tolva de alimentación en el extremo izquierdo con su guía cónica para centrar los rocotos sobre la faja, seguida de la zona de rodillos de transporte, la cámara de inspección central y la zona de clasificación con los dos canales de desvío simétricos. Las dos cajas de recolección se ubican

perpendicularmente al eje de la faja, una a cada lado, permitiendo la separación física de los rocotos según su clasificación. El plano fue elaborado en AutoCAD para garantizar la precisión dimensional de los componentes fabricados en MDF de 4 mm.

Figura 53:

Vista superior del prototipo ROCOBOT — diseño 2D en AutoCAD

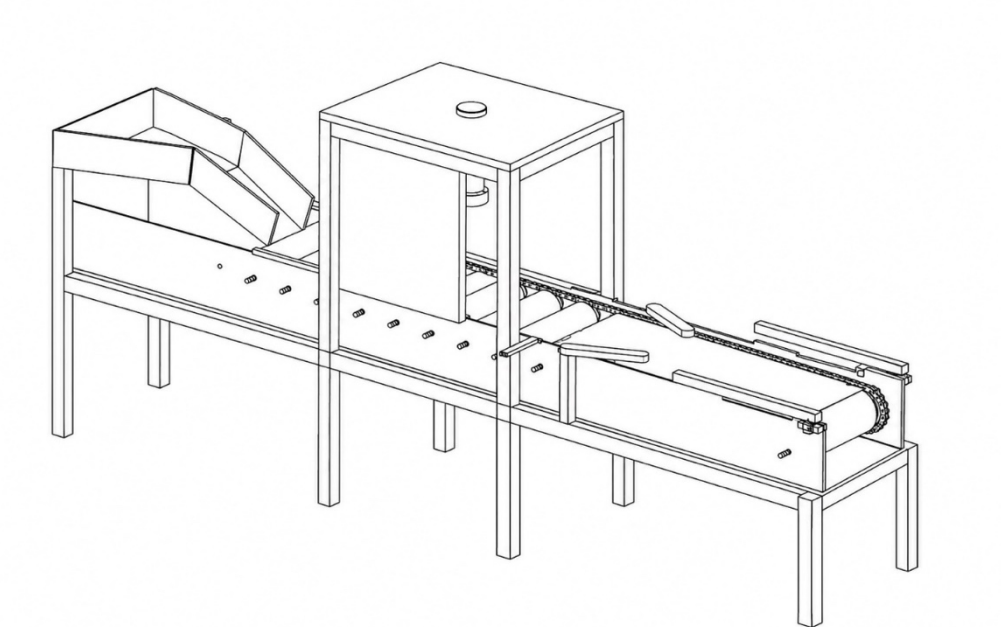


Nota. Elaboración propia (2025).

La Figura 54 presenta la vista isométrica del modelo 3D del prototipo ROCOBOT, permitiendo visualizar la integración mecánica de todos los componentes del sistema. Se distingue la estructura de soporte fabricada en MDF, la faja transportadora con rodillos accionada por el motorreductor JGB37-520 a través de la cadena de transmisión visible en el lateral derecho, la cámara de inspección montada sobre la estructura central y las dos paletas de clasificación controladas por los servomotores MG995. El diseño modular facilita el mantenimiento y la sustitución de componentes individuales sin afectar el funcionamiento del sistema completo.

Figura 54:

Vista isométrica 3D del prototipo ROCOBOT — modelo CAD



Nota. Elaboración propia (2025).

4.3.5.2. Vista General de la cinta transportadora

La Figura 55 presenta el render 3D del prototipo ROCOBOT con sus componentes físicos integrados. La tolva de alimentación (extremo izquierdo) permite el ingreso controlado de los rocotos sobre la faja transportadora con rodillos, que los desplaza hacia la zona de inspección. La cámara de inspección —alojada en una caja cerrada de cartón espuma con tira de luz LED en su interior— garantiza condiciones de iluminación controladas para la captura de imágenes. Una vez clasificado el rocoto por el modelo de Deep Learning, los dos servomotores MG995 accionan paletas de desvío que direccionan el fruto hacia uno de los dos canales de salida: el canal izquierdo (rojo) para rocotos defectuosos y el canal derecho (madera) para rocotos saludables, cayendo en las respectivas cajas de recolección.

Figura 55:

Vista general 3D del prototipo ROCOBOT — render de diseño con componentes físicos



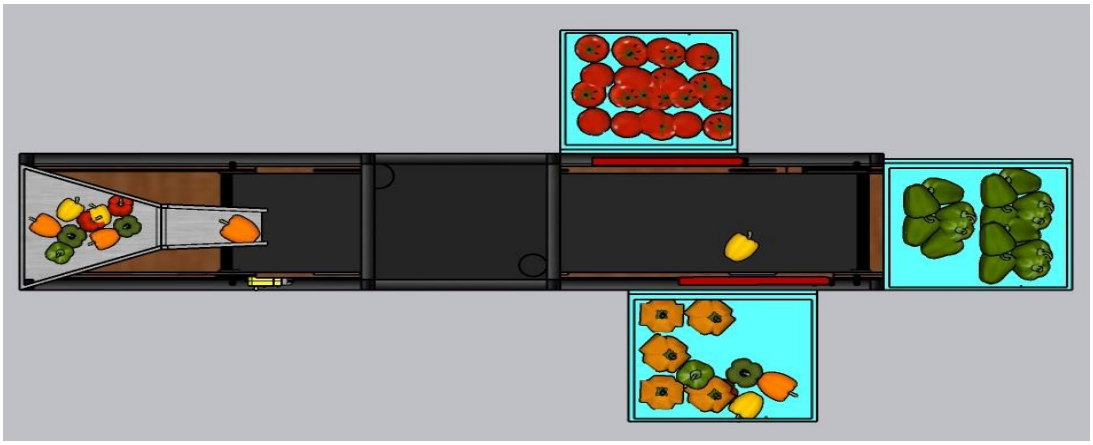
Nota. Elaboración propia (2025).

4.3.5.3. Vista Superior 3D del Prototipo

A continuación se muestra la vista superior del robot Rocobot, donde se aprecia la distribución completa de los componentes principales del sistema. En la parte izquierda se observa el embudo de alimentación, por donde se introducen los rocotos hacia la cinta transportadora. A lo largo de la cinta, los frutos pasan por la zona de inspección, donde se encuentran la cámara y las luces LED encargadas de capturar las imágenes para el proceso de detección y clasificación. En los laterales se ubican los mecanismos clasificadores, que se abren y cierran para redirigir los rocotos hacia los contenedores de destino, ubicados a ambos lados del sistema. Cada compartimiento almacena los frutos clasificados según su condición: saludables, defectuosos o descartados.

Figura 56:

Vista Superior del Rocobot



Nota. Elaboración propia (2025).

4.3.6. Construcción del prototipo final

Una vez validado el diseño en AutoCAD y el modelo 3D, se procedió a la construcción física del prototipo ROCOBOT siguiendo las especificaciones establecidas en los apartados anteriores. Se presenta una fotografía del prototipo ROCOBOT construido en vista lateral y horizontal, donde se observa la tolva de alimentación con rocotos y pimientos sobre la mesa de trabajo, la caja de inspección cerrada con iluminación interna y la faja transportadora con sus canales de salida hacia las cajas de recolección. Se aprecia la tolva de alimentación con los rocotos de prueba, la fuente de alimentación SMPS y la placa Raspberry Pi 4B montadas sobre la estructura, y la faja transportadora con los rodillos y el sistema de desvío en la parte inferior.

La estructura mecánica fue diseñada en AutoCAD y fabricada mediante corte láser con material MDF. La banda transportadora de cuero sintético PVC (1.6m × 16 cm). La cámara Raspberry Pi Module 3 Wide (12 MP, campo de visión 120°, autofocus motorizado) fue instalada en la parte superior de la caja de espuma, alineada verticalmente sobre la faja. Las tiras LED proporcionan iluminación homogénea y constante, eliminando la variabilidad de la iluminación ambiental que afectaría la consistencia del modelo de IA.

A diferencia de los antecedentes revisados que utilizan PC con GPU (Mohi-Alden et al., 2023), cámaras industriales RGB-D de alto costo (Dairath et al., 2023) o sistemas sin sincronización entre faja y pipeline de IA (Kamboj y Diwan, 2019; Sanwar y Ahmed, 2023), ROCOBOT integra hardware embebido de bajo costo (Raspberry Pi 4B) con una faja transportadora sincronizada directamente con el pipeline de IA y una caja de inspección con

iluminación controlada, condición no reportada en ninguno de los antecedentes revisados para sistemas con Raspberry Pi. La Figura 57 presentan el resultado final de la construcción.

Figura 57:

Prototipo ROCOBOT construido — vista frontal y vista lateral



Nota. Fotografías del prototipo ROCOBOT. Elaboración propia (2026).

4.4. Desarrollo del aplicativo web para la supervisión remota

4.4.1. Selección y justificación del sistema web

En la revisión de los antecedentes de la literatura no se encontró ningún sistema robótico de clasificación de frutos que reporte el desarrollo de un aplicativo web integrado para la supervisión remota del proceso. La Tabla 21 compara los antecedentes revisados, evaluando la presencia de aplicativo web, monitoreo en tiempo real y trazabilidad de resultados en base de datos.

Tabla 21:

Comparación del aplicativo web ROCOBOT vs. antecedentes de la literatura

Sistema / Referencia	App web	Tiempo real	Trazab. BD	Observación
Kamboj & Diwan (2019)	No	No	No	Solo interfaz HMI local (PLC)
Ismail & Malik (2022)	No	No	No	Sin app web ni trazabilidad
Dairath et al. (2023)	No	No	No	Sin app web ni reportes
Mohi-Alden et al. (2023)	No	No	No	Sin supervisión remota
Moya et al. (2025)	No	No	No	Sin app web integrada
ROCOBOT (2026)	Sí	Sí	Sí	FastAPI + React, producción en Vercel + Railway

Nota. BD = Trazabilidad de resultados en base de datos. Sistema propuesto. Elaboración propia (2026).

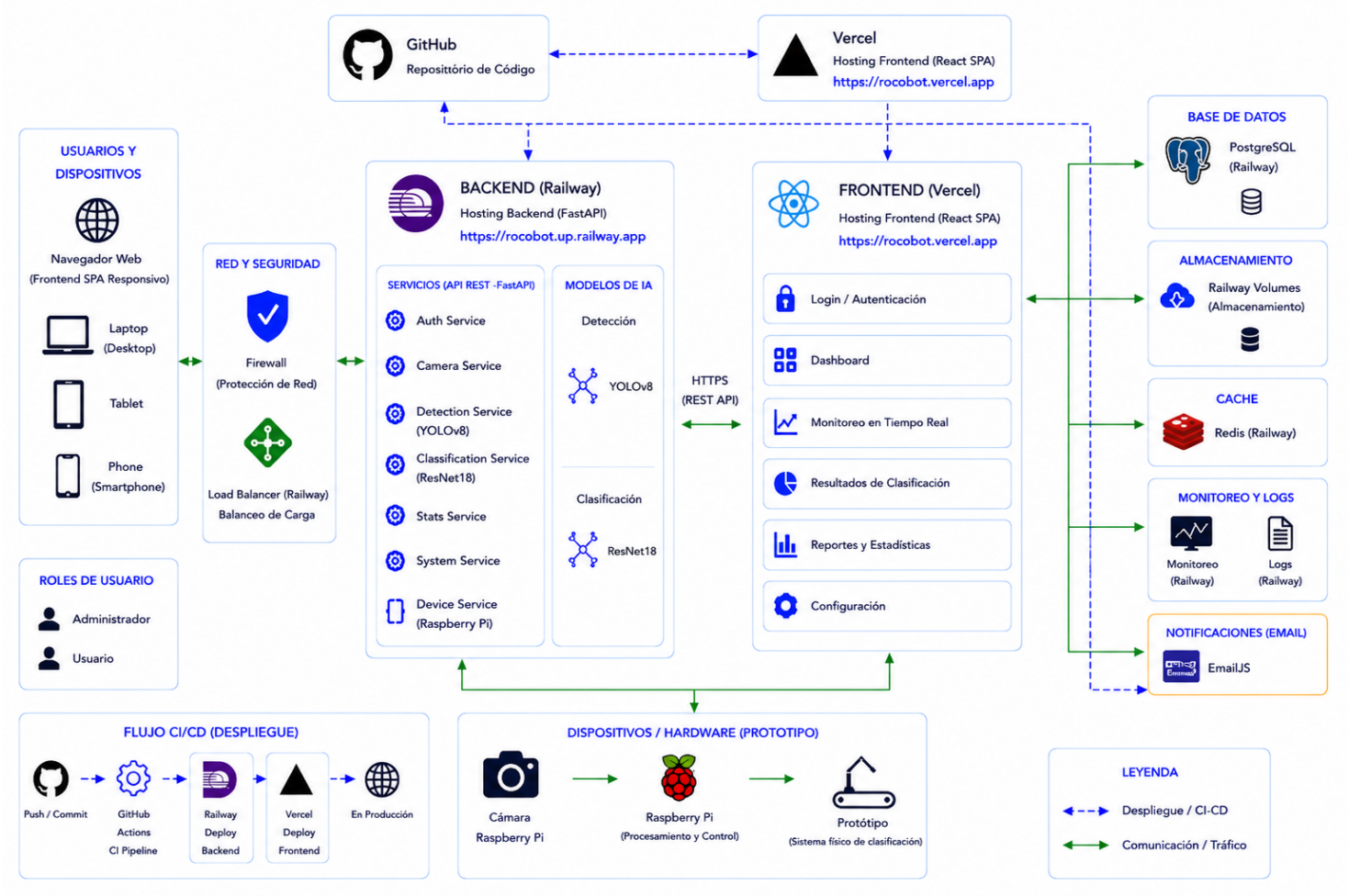
Del análisis se evidencia que ninguno de los antecedentes revisados reporta el desarrollo de un aplicativo web integrado al sistema robótico de clasificación de frutos. Kamboj y Diwan (2019) y Sanwar y Ahmed (2023) emplean interfaces HMI locales sin acceso remoto, mientras que Ismail y Malik (2022), Dairath et al. (2023) y Mohi-Alden et al. (2023) no reportan ningún mecanismo de supervisión remota ni trazabilidad de resultados en base de datos. Esta ausencia en la literatura identifica una brecha tecnológica relevante para la industria agroalimentaria, donde el productor requiere acceso a los datos de clasificación desde cualquier dispositivo sin estar presente físicamente en la planta.

4.4.2. Arquitectura del sistema

La arquitectura del sistema ROCOBOT se diseñó como un monolito modular en el backend, aplicando los principios de Clean Architecture (Martin, 2017), y un modelo de control de acceso basado en roles (RBAC) para la gestión de permisos. La Figura 56 presenta el diagrama general de la arquitectura del sistema, que constituye la base sobre la cual se desarrollaron el backend, el frontend, la base de datos y el flujo de despliegue descritos en los apartados siguientes.

Figura 58:

Arquitectura general del sistema ROCOBOT.



Nota: Elaboración propia (2026)

La decisión de implementar un monolito modular responde a que el sistema ROCOBOT, al tratarse de un proyecto académico con tráfico moderado y un equipo de desarrollo reducido, no requiere el escalado independiente por módulo que justificaría la complejidad operativa de servicios distribuidos (Fowler, 2015). El monolito modular conserva los beneficios de la separación de responsabilidades, cada módulo puede probarse y modificarse de forma aislada, sin asumir ese costo.

Se aplicó Clean Architecture porque el sistema integra dos dominios que cambian a ritmos distintos: la lógica de inferencia de los modelos de IA, que se reentrena y compara con frecuencia durante la investigación, y la persistencia de datos, que debe permanecer estable. Al aislar estas responsabilidades en capas independientes del API, servicios, infraestructura, modificar el modelo de clasificación o el motor de base de datos no obliga a modificar el resto del sistema (Martin, 2017).

RBAC se adoptó porque el sistema distingue dos tipos de usuario con responsabilidades claramente diferenciadas: el administrador, que gestiona las cuentas del sistema, registro, habilitación y deshabilitación de usuarios y asignación de roles, y el usuario, que corresponde a cada cliente con su propia cuenta independiente para supervisar el proceso de clasificación, consultar sus reportes históricos y visualizar el monitoreo en tiempo real de sus sesiones. Asignar permisos a roles predefinidos en lugar de a usuarios individuales facilita la gestión de cualquier número de cuentas sin reconfigurar permisos uno por uno (Sandhu et al., 1996).

PostgreSQL se utilizó porque los datos del sistema son inherentemente relacionales: cada resultado de clasificación pertenece a una ejecución, cada ejecución fue realizada por un usuario con un rol determinado, y los reportes estadísticos requieren agregaciones que se resuelven de forma nativa mediante SQL, garantizando la integridad referencial entre las tablas. Redis se incorporó como caché porque el panel de monitoreo en tiempo real requiere lecturas frecuentes de baja latencia que sería ineficiente resolver consultando directamente la base de datos relacional en cada actualización del dashboard. EmailJS se integró desde el frontend para el envío de correos transaccionales, recuperación de contraseña y bienvenida al nuevo usuario, sin requerir infraestructura propia de servidor de correo. Vercel se eligió para el despliegue del frontend por su CDN global y su integración nativa con GitHub Actions. Railway se eligió para el backend porque permite alojar el backend y PostgreSQL bajo el mismo proveedor con comunicación por red privada interna, evitando exponer la base de datos directamente a internet.

La comunicación entre el frontend y el backend se realiza exclusivamente mediante HTTPS a través de la API REST. El prototipo físico, a través de la Raspberry Pi, consume los mismos servicios REST del backend, lo que permite un diseño homogéneo y desacoplado entre el software y el hardware del sistema.

4.4.3. Desarrollo del sistema web

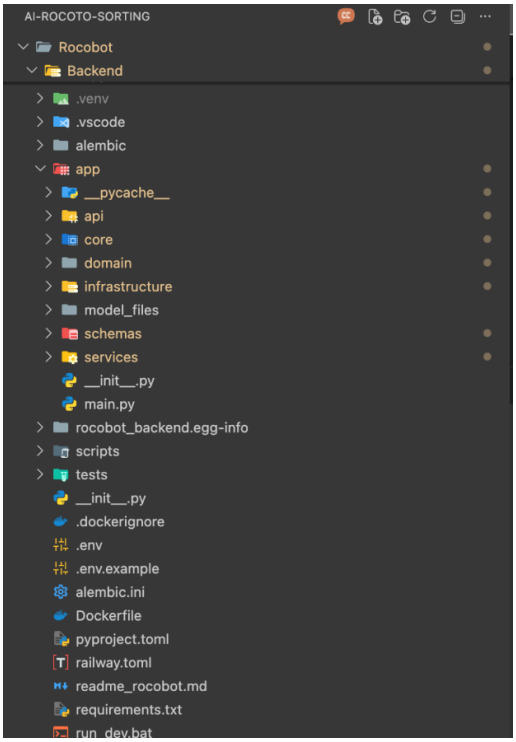
4.4.3.1. Backend (FastAPI + Railway)

El backend del sistema ROCOBOT fue desarrollado utilizando FastAPI 0.121.2, un framework de Python de alto rendimiento que destaca por su soporte nativo de programación asíncrona, validación automática de datos mediante Pydantic y documentación interactiva mediante OpenAPI (Swagger UI). El servicio se encuentra desplegado en Railway bajo la URL https://rocobot.up.railway.app.

La arquitectura interna implementa cuatro patrones de diseño: el patrón repositorio para la abstracción del acceso a datos, la capa de servicios para la separación de la lógica de negocio, la inyección de dependencias mediante el mecanismo Depends de FastAPI, y el patrón fábrica para la selección dinámica de los modelos clasificadores. La Figura 59 muestra la estructura de carpetas del módulo Backend del proyecto.

Figura 59:

Estructura de carpetas del Backend ROCOBOT



Nota. Elaboración propia (2025).

Se muestra la organización interna del módulo Backend, compuesto por las carpetas app (lógica principal del sistema), alembic (migraciones de base de datos), scripts y tests, junto con los archivos de configuración Dockerfile, railway.toml, alembic.ini y requirements.txt. Las dependencias principales del backend son: FastAPI 0.121.2, SQLAlchemy 2.0.30, PyJWT 2.11.0, Pydantic 2.12.4, PyTorch 2.9.1 y OpenCV 4.12.0 para el procesamiento de imágenes y la ejecución de los modelos de IA. El código fuente completo del backend se presenta en el Anexo A.2.

El pipeline de inferencia implementado en el backend sigue la secuencia: captura de imagen → ejecución de YOLOv8 como detector → extracción de la región de interés (ROI) si se detecta un rocoto con confianza mayor o igual a 0.70 → ejecución del modelo CNN (ResNet18 o EfficientNetB0) sobre el ROI → emisión del resultado (saludable/defectuoso) con su nivel de confianza por frame.

Mientras el rocoto pasa por la zona de inspección, la cámara captura varios frames de él en distintos momentos de su rotación. Cada frame se procesa por separado: se detecta el rocoto en la imagen, se recorta esa región y se pasa por el modelo clasificador, que devuelve la clase (saludable o defectuoso) y un valor de confianza

entre 0 y 1. Con todos los resultados por frame, el sistema decide el resultado final del rocoto en dos pasos, aplicados en orden, deteniéndose en el primero que se cumpla:

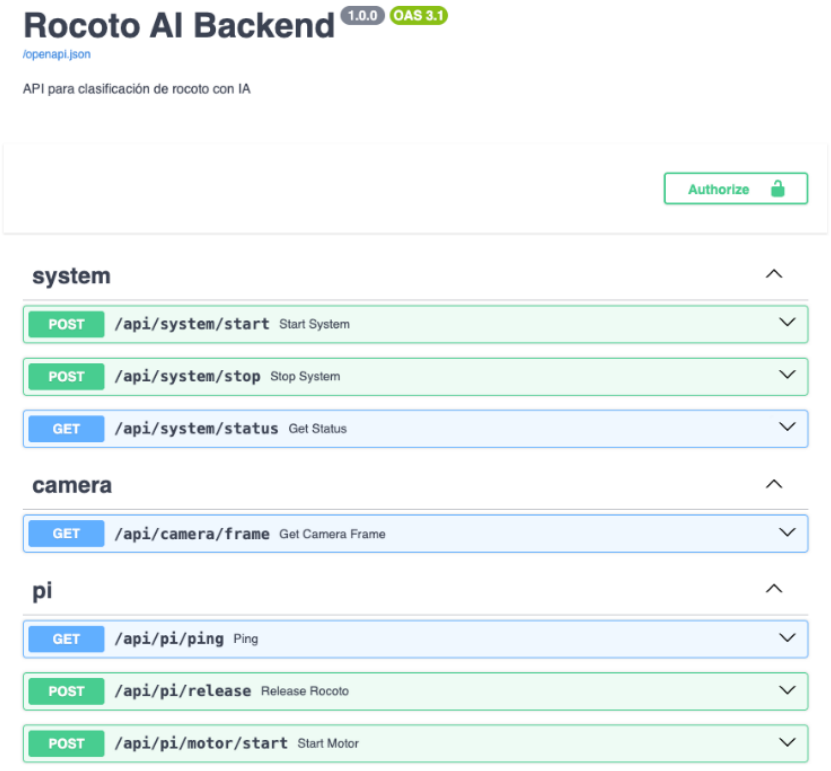
Si al menos 3 frames distintos fueron clasificados como defectuosos, se toman los 3 con mayor confianza y se promedian; si ese promedio llega a 0.85 o más, el rocoto se marca como defectuoso. Este paso permite atrapar casos de defecto parcial que solo se ven desde algunos ángulos, dado que la coincidencia de varios frames independientes en un defecto con confianza alta es una señal más confiable que un solo frame aislado.

Si no se cumple lo anterior, se suma toda la confianza acumulada de los frames clasificados como "defectuoso" y toda la de los clasificados como "saludable", y se calcula la proporción del total que corresponde a "defectuoso". Si esa proporción es 0.5 o más, el resultado final es defectuoso; en caso contrario, es saludable. Este cálculo pesa cada frame según la confianza del modelo, evitando que clasificaciones débiles empaten o superen a una clasificación mucho más segura.

La Figura 60 presenta la documentación interactiva del backend generada automáticamente por FastAPI mediante Swagger UI.

Figura 60:

Documentación interactiva Swagger UI del backend ROCOBOT

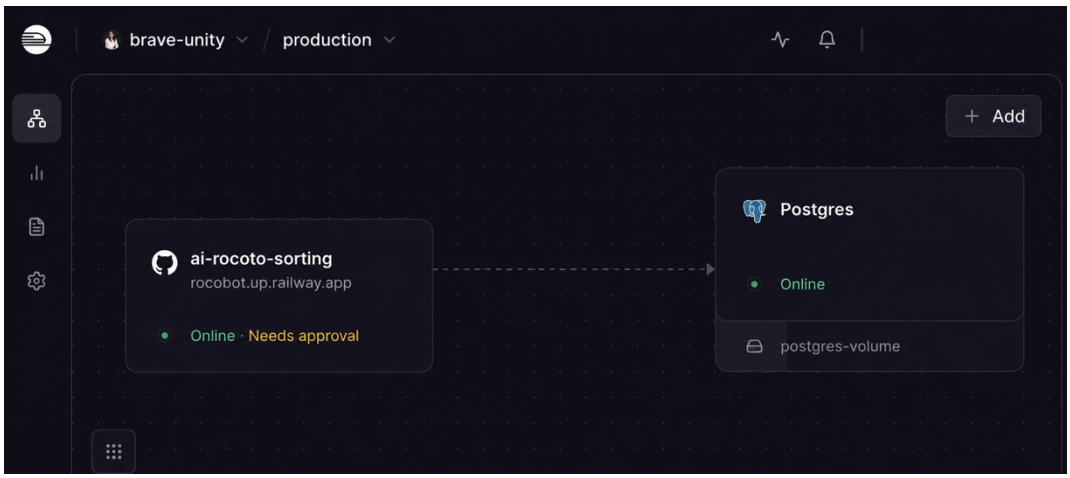


Nota. Elaboración propia (2025).

Se muestra los grupos de endpoints disponibles en la API REST: el módulo system con los endpoints de inicio, parada y consulta de estado del proceso de clasificación; el módulo camera con el endpoint de captura de fotogramas; y el módulo pi con los endpoints de comunicación con el prototipo físico, incluyendo ping, liberación del rocoto y control del motor de la faja transportadora. La Figura 61 muestra el entorno de producción del backend en Railway.

Figura 61:

Entorno de producción del backend ROCOBOT en Railway



Nota. Elaboración propia (2025).

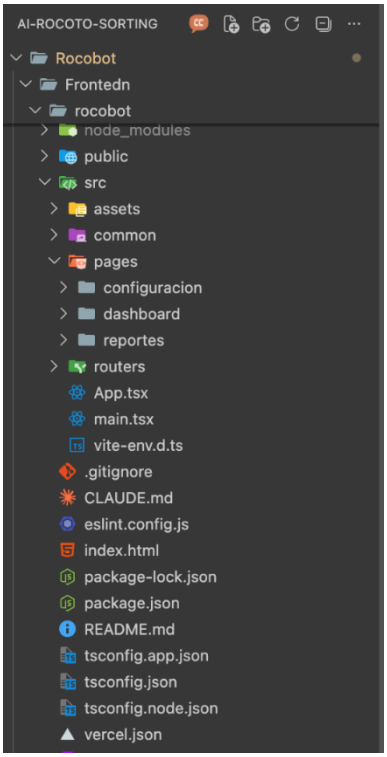
Acá se muestra el proyecto ai-rocoto-sorting desplegado en Railway bajo el entorno production, con el servicio del backend (rocobot.up.railway.app) conectado mediante red privada interna al servicio PostgreSQL con su volumen de almacenamiento (postgres-volume). Ambos servicios aparecen con estado Online, confirmando el correcto funcionamiento del entorno de producción.

4.4.3.2. Frontend Web (React SPA + Vercel)

El frontend del sistema ROCOBOT fue desarrollado como una Single Page Application utilizando React 19 con TypeScript, Vite 7.2 como herramienta de construcción y Fluent UI v9 como biblioteca de componentes. La aplicación se encuentra desplegada en Vercel bajo la URL https://rocobot.vercel.app. La comunicación con el backend se realiza mediante la biblioteca Axios, que intercepta automáticamente los tokens JWT para cada solicitud autenticada. La Figura 62 muestra la estructura de carpetas del módulo Frontend del proyecto.

Figura 62:

Estructura de carpetas del Frontend ROCOBOT



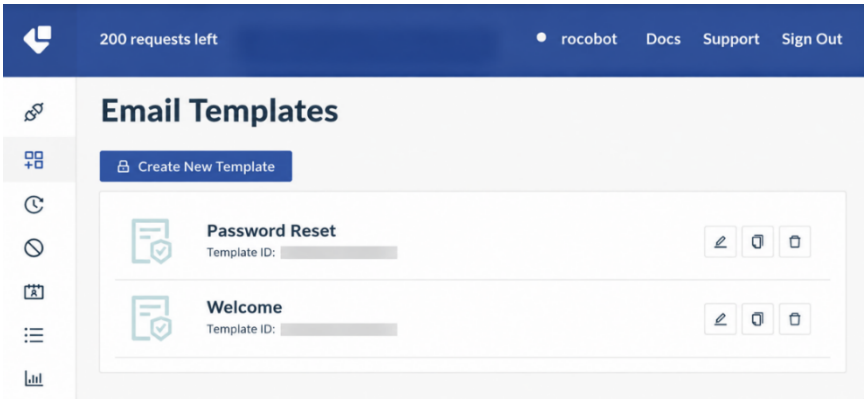
Nota. Elaboración propia (2025).

La organización modular del frontend por dominio funcional. El directorio src/common agrupa los módulos reutilizables: auth (contexto de autenticación), components (componentes visuales reutilizables), hooks (lógica de negocio encapsulada), interface (tipos TypeScript), services (servicios HTTP y EmailJS) y utils (utilidades). El directorio src/pages contiene las páginas principales: dashboard (panel de monitoreo en tiempo real), reportes (cuadro de mando con gráficos Recharts) y configuracion. El directorio src/routers gestiona el enrutamiento protegido mediante ProtectedRoutes, que redirigen al login si no existe un token JWT válido.

Para el envío de correos transaccionales se integró EmailJS (@emailjs/browser v4.4.1) directamente desde el frontend, sin requerir infraestructura propia de servidor de correo. EmailJS gestiona dos plantillas: Password Reset para la recuperación de contraseña, y Welcome para el correo de bienvenida al registrar una nueva cuenta. La Figura 63 muestra las plantillas configuradas en el panel de EmailJS.

Figura 63:

Plantillas de correo transaccional configuradas en EmailJS para ROCOBOT

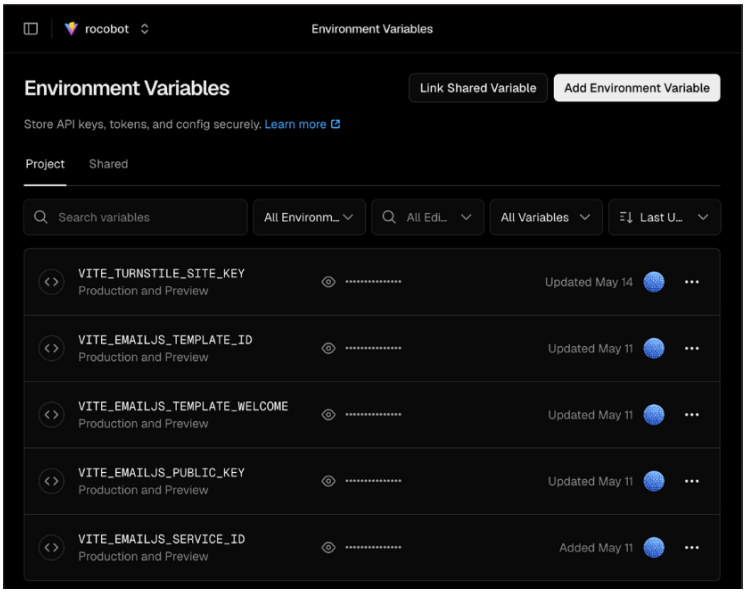


Nota. Elaboración propia (2025).

Las variables de entorno del frontend , incluyendo las claves de EmailJS y la clave de Cloudflare Turnstile, se gestionan de forma segura mediante el panel de Vercel, sin exponerlas en el código fuente del repositorio. La Figura 64 muestra el panel de variables de entorno del proyecto en Vercel.

Figura 64:

Variables de entorno del frontend ROCOBOT configuradas en Vercel



Nota. Elaboración propia (2025).

Las variables de entorno del proyecto ROCOBOT configuradas en Vercel para los entornos de producción y preview. Las variables VITE_EMAILJS almacenan las credenciales del servicio de correo transaccional, mientras que

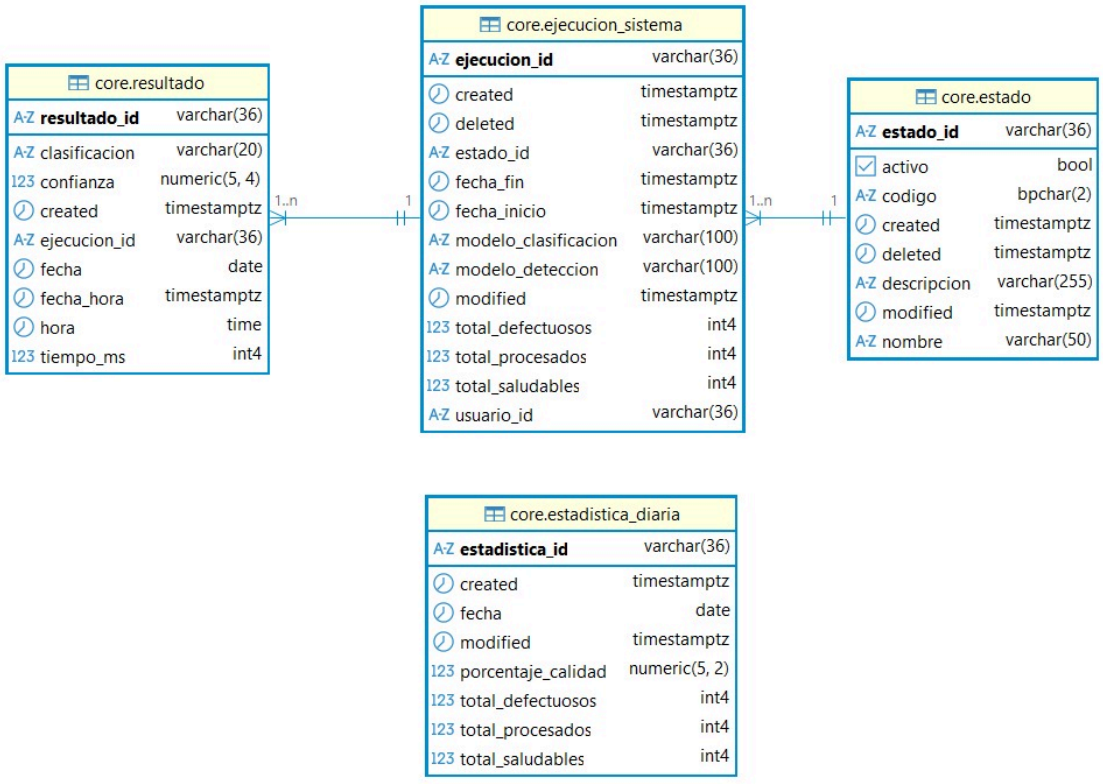
VITE_TURNSTILE_SITE_KEY almacena la clave pública del servicio Cloudflare Turnstile utilizado en el formulario de registro para prevención de bots. El código fuente completo del frontend se presenta en el Anexo A.3. El manual de usuario del aplicativo web se presenta en el Anexo D.

4.4.3.3. Base de Datos

La base de datos del sistema ROCOBOT utiliza PostgreSQL como motor relacional, desplegado en Railway con respaldo automático diario. El diseño del esquema fue realizado aplicando el modelo entidad-relación (ER) y los principios de normalización hasta la tercera forma normal (3FN), garantizando la integridad referencial y evitando la redundancia de datos. El esquema se organiza en tres módulos funcionales: auth (autenticación y roles RBAC), menu (navegación y permisos CRUD por rol) y core (producción y resultados de clasificación). La Figura 65 presenta el diagrama entidad-relación del esquema core, que constituye el núcleo operativo del sistema.

Figura 65:

Diagrama entidad-relación del esquema core de la base de datos ROCOBOT



Nota. Elaboración propia (2025).

Las cuatro entidades del esquema core y sus relaciones. La entidad ejecucion_sistema registra cada sesión de operación del prototipo identificando los modelos de IA utilizados (modelo_deteccion y modelo_clasificacion), permitiendo trazabilidad de qué versión de modelo generó cada conjunto de resultados. Por cada ejecución se generan múltiples registros en la entidad resultado —relación 1:N—, uno por cada rocoto clasificado, almacenando su fecha, hora, clasificación asignada y nivel de confianza (campo confianza, tipo numeric(5,4), rango 0.0–1.0). La entidad estadistica_diaria agrega los resultados por jornada calculando el porcentaje_calidad diario. La entidad estado actúa como catálogo de estados del sistema. El esquema auth implementa el modelo RBAC: separa los datos personales del usuario (entidad Persona) de sus credenciales de acceso (entidad Usuario) mediante una relación 1:N, y gestiona los roles a través de la tabla usuarioRol que implementa una relación N:M entre usuarios y roles. El administrador gestiona las cuentas del sistema —registro, habilitación y deshabilitación de usuarios y asignación de roles—, mientras que cada usuario cliente tiene su propia cuenta independiente para supervisar sus sesiones de clasificación y consultar sus reportes históricos. El esquema menu materializa los permisos RBAC: la tabla rol_menu asocia cada entrada de menú con un rol específico, determinando qué pantallas puede ver cada rol y qué operaciones CRUD tiene permitidas, sin codificar los permisos directamente en el frontend. La documentación técnica completa del esquema, incluyendo los diagramas ER de los esquemas auth y menu, se presenta en el Anexo C, Diccionario de Base de Datos.

4.4.3.4. CI/CD y Despliegue

El sistema ROCOBOT implementa un flujo de integración y despliegue continuos (CI/CD) configurado con GitHub Actions, integrado con Railway para el backend y Vercel para el frontend. El flujo se activa automáticamente con cada push a la rama principal y se desarrolla en cuatro etapas: (1) GitHub Actions ejecuta el pipeline de integración continua —instalación de dependencias, pruebas unitarias, análisis estático del código y generación de artefactos—; (2) Railway construye la imagen Docker del backend, ejecuta las migraciones pendientes mediante Alembic y despliega la nueva versión sin tiempo de inactividad (zero-downtime deployment); (3) Vercel genera la build de producción del frontend y la distribuye en su CDN global de forma instantánea; y (4) GitHub Actions ejecuta pruebas de humo (smoke tests) sobre los endpoints críticos para confirmar la correcta operación de la nueva versión. Este diseño

garantiza que cualquier mejora sea desplegada en producción en un tiempo estimado de tres a cinco minutos desde el commit, con trazabilidad completa de cada versión en el historial de GitHub Actions.

4.4.4. Funcionamiento del sistema en tiempo real

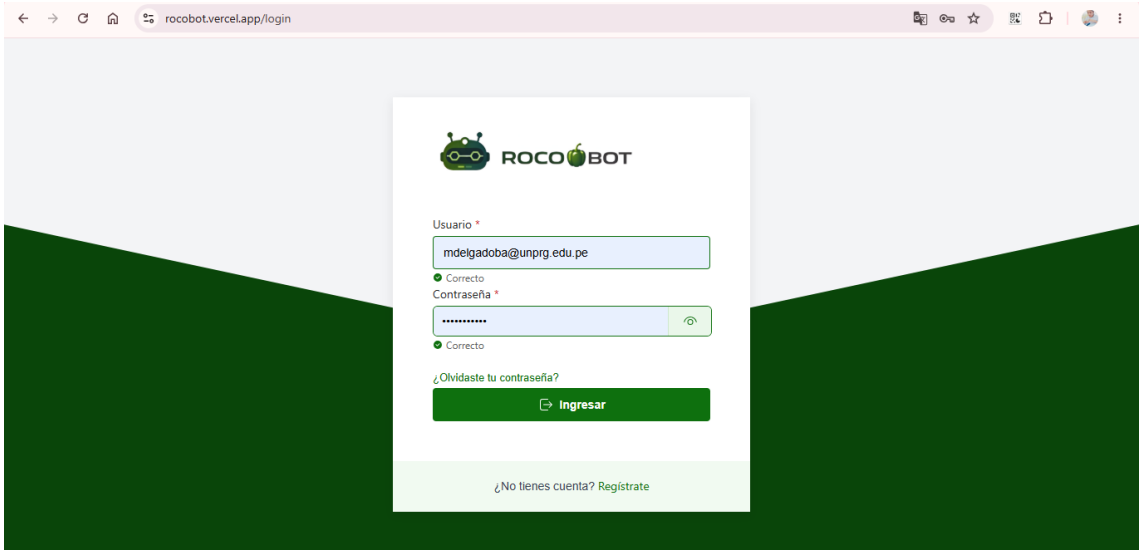
Una vez desarrollado y desplegado, el aplicativo ROCOBOT Web es accesible desde cualquier navegador moderno mediante la URL https://rocobot.vercel.app. A continuación se presentan las capturas reales del sistema en operación, correspondientes a las pantallas principales del aplicativo.

4.4.4.1. Acceso y registro

A continuación se muestra la pantalla de inicio de sesión del aplicativo web ROCOBOT, accesible desde la URL https://rocobot.vercel.app/login. El sistema valida el formato del correo electrónico y la contraseña antes de enviar la solicitud al backend, reduciendo peticiones innecesarias. Una vez autenticado, el sistema genera un token JWT que se almacena en el cliente y se adjunta automáticamente a cada solicitud subsiguiente mediante el interceptor Axios. Tal como se muestra en la figura 66.

Figura 66:

Pantalla de inicio de sesión del aplicativo web ROCOBOT



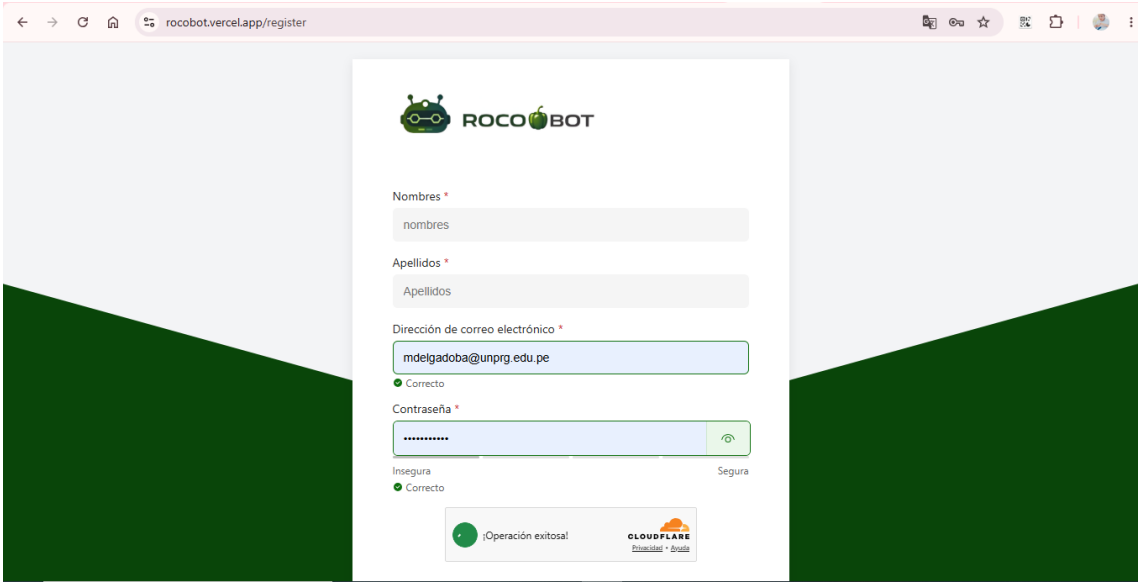
Nota. Elaboración propia (2026).

La Figura 67 muestra la pantalla de registro de nueva cuenta, que solicita nombres, apellidos, correo electrónico y contraseña. El formulario incorpora el widget

de verificación Cloudflare Turnstile —visible en la parte inferior— que confirma que el solicitante es un ser humano antes de permitir la creación de la cuenta, previniendo el registro automatizado por bots. Al completar el registro exitosamente, EmailJS envía automáticamente un correo de bienvenida al usuario mediante la plantilla Welcome.

Figura 67:

Pantalla de registro de cuenta del aplicativo web ROCOBOT



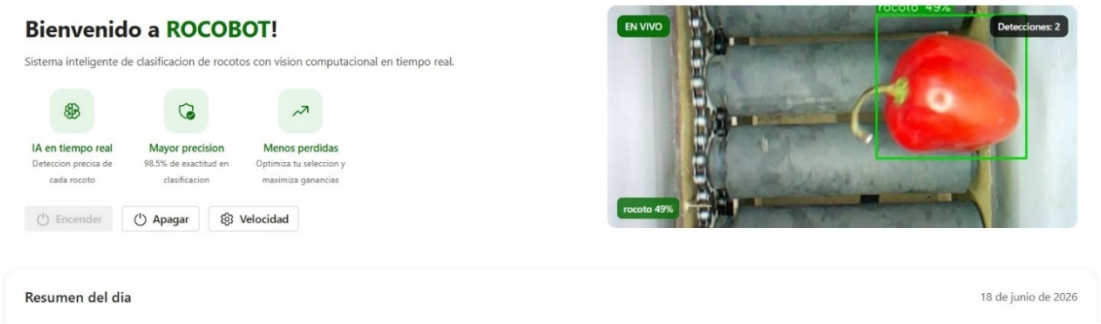
Nota. Elaboración propia (2026).

4.4.4.2. Panel principal y monitoreo en tiempo real

El panel principal durante una sesión real de operación con el indicador EN VIVO activo. La transmisión de video de la cámara Raspberry Pi captura un rocoto sobre la faja transportadora del prototipo físico, con el recuadro verde del modelo YOLOv8n superpuesto y su nivel de confianza (rocoto 49%). El panel muestra el resumen del día con la fecha actual y los controles de encendido, apagado y velocidad de la faja disponibles para el usuario, como se muestra n la figura 68.

Figura 68:

Panel principal del aplicativo web— detección de rocoto en tiempo real



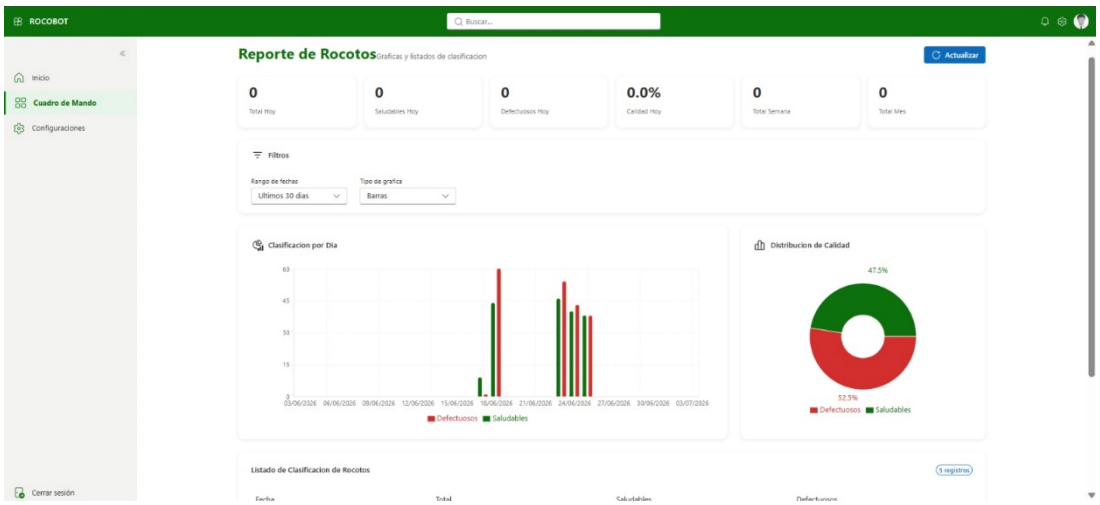
Nota. Elaboración propia (2026).

4.4.4.3. Cuadro de mando

El Cuadro de Mando ofrece el análisis histórico de los datos de clasificación para la toma de decisiones del productor. Permite filtrar por rango de fechas (7 días, 30 días, 3 meses o personalizado) y visualizar la evolución temporal mediante gráficas de barras o líneas, así como la distribución de calidad del período seleccionado. La base de datos PostgreSQL registra cada decisión individual de clasificación con su timestamp, nivel de confianza y estado del rocoto, garantizando la trazabilidad completa del proceso. El acceso al aplicativo en producción se presenta en el Anexo B.

Figura 69:

Cuadro de mando del aplicativo web — reporte histórico de clasificación



Nota. Elaboración propia (2026).

4.5. Validación mediante pruebas piloto del sistema

4.5.1. Antecedentes de validación en sistemas robóticos de clasificación

La revisión de la literatura no permitió identificar un antecedente que integre, de manera conjunta, el desarrollo de un sistema robótico completo y su validación mediante pruebas piloto en condiciones reales de operación. Sin embargo, se identificaron tres antecedentes que sí reportan una prueba piloto física de un sistema integrado sobre un lote de muestras reales —criterio que los distingue de estudios centrados en el desarrollo aislado de un prototipo, un modelo de inteligencia artificial o un aplicativo software, los cuales se abordan en los Objetivos 2, 3 y 4, respectivamente—: Abdal et al. (2024), con 100 pruebas sobre objetos físicos; Ismail y Malik (2022), con pruebas en tiempo real sobre 30 manzanas y 32 plátanos; y Tan et al. (2025), con pruebas de hardware sobre 60 manzanas. A continuación, se detalla cada uno de estos antecedentes, cuyas condiciones de prueba y métricas reportadas se resumen en la Tabla 22.

Tabla 22:

Antecedentes de validación de sistemas robóticos de clasificación de frutos

N°	Autor/Año	Producto	N° pruebas	Métrica reportada	Hardware
1	Abdal et al. (2024)	Objetos genéricos	100	89% precisión	Raspberry Pi 4B
2	Ismail & Malik (2022)	Manzana/Plátano	62 (30+32)	96.7% / 93.8% precisión	Raspberry Pi
3	Tan et al. (2025)	Manzana	60	97% (un lado) / 100% (múltiples lados)	Raspberry Pi 4B

Nota. NR = No reportado. Elaboración propia (2026).

Se desprende que los tres antecedentes reportan validaciones piloto con métricas de precisión entre 89% y 100%, obtenidas sobre lotes físicos de entre 60 y 100 muestras, y ninguno documenta la trazabilidad individual de cada clasificación con su nivel de confianza y tiempo de procesamiento registrado automáticamente en una base de datos, característica propia del sistema ROCOBOT. Tan et al. (2025) alcanzó 100% de precisión mediante detección de múltiples lados con rotación controlada de la manzana sobre una plataforma giratoria, condición distinta al flujo continuo en banda transportadora empleado en

ROCOBOT. Con base en este análisis, se ejecutaron 150 pruebas piloto con rocotos (Capsicum pubescens) sobre el sistema robótico ROCOBOT integrado en condiciones controladas, superando en volumen de pruebas a los tres antecedentes revisados. Cada ejecución registró automáticamente la clasificación asignada, el nivel de confianza del modelo y el tiempo de procesamiento en milisegundos, almacenados en tiempo real mediante el aplicativo web. Los resultados completos de las 150 ejecuciones se presentan en el Anexo F.

4.5.2. Integración del sistema ROCOBOT

Las pruebas piloto validaron el funcionamiento del sistema ROCOBOT como un sistema integrado de cuatro componentes operando en conjunto: el prototipo físico con la faja transportadora y los servomotores, el pipeline de detección YOLOv8n y clasificación ResNet18 ejecutado sobre la Raspberry Pi 4B, el aplicativo web ROCOBOT Web para la supervisión remota, y el registro automático de resultados en la base de datos PostgreSQL. Durante cada ejecución, el rocoto ingresa por la tolva de alimentación sobre la faja transportadora, el sensor infrarrojo FC-51 detecta su presencia y activa la captura, la cámara Raspberry Pi Module 3 Wide captura la imagen, el modelo YOLOv8n detecta y localiza el rocoto, el modelo ResNet18 lo clasifica como saludable o defectuoso con su nivel de confianza, el resultado es enviado al backend FastAPI que lo almacena en PostgreSQL y lo transmite al frontend React en tiempo real, el servomotor MG995 correspondiente es accionado para desviar el rocoto hacia el canal de salida correcto, y el aplicativo web actualiza los indicadores del panel principal con el nuevo registro. Esta cadena completa se ejecutó de forma ininterrumpida a lo largo de las tres sesiones de prueba realizadas el 23, 24 y 25 de junio de 2026, sin caídas del sistema ni pérdida de registros.

4.5.3. Resultados de las pruebas piloto

Sobre esta base operativa, se analizaron en detalle los 150 registros generados por el sistema para caracterizar su comportamiento en términos de clasificación, confianza y tiempos de procesamiento. La Figura 70 presenta una muestra representativa de 10 de estas ejecuciones.

Figura 70:

Muestra representativa de 10 ejecuciones — pruebas piloto sistema ROCOBOT (n=150)

resultado_id	ejecucion_id	fecha	hora	fecha_hora	clasificacion	confianza	created	tiempo_ms
03e09b9d-3a5d-4106-8f5c	a0aceb27-a953-402e-97a	24/06/26	18:14.2	18:14.2	saludable	1	18:14.2	26788
0517f94c-9864-48b0-ba07	134e99e3-cd4e-4d73-8ef	23/06/26	14:46.3	14:46.3	saludable	1	14:46.3	4212
0d6cda00-f74e-42ba-919f-	5ea24d6d-9a02-45c8-abb	25/06/26	59:55.4	59:55.4	saludable	1	59:55.4	4731
0db0c67e-0cda-49f1-b2e7	a0aceb27-a953-402e-97a	24/06/26	28:22.9	28:22.9	defectuoso	1	28:22.9	4129
10bccc4b-963b-4967-a9cb	a0aceb27-a953-402e-97a	24/06/26	00:34.6	00:34.6	defectuoso	1	00:34.6	13010
113f5835-7e8d-4161-9140	a0aceb27-a953-402e-97a	24/06/26	05:30.9	05:30.9	saludable	1	05:30.9	4082
14836568-c7aa-4849-834e	44b3bf5c-c61c-41cf-ae9a	25/06/26	08:43.7	08:43.7	saludable	1	08:43.7	3851
1c960eed-8947-427e-a2f5	5b4a18ed-053c-4224-908	25/06/26	16:25.7	16:25.7	saludable	1	16:25.7	3494
1ec8e1a9-2903-4b0b-aeb5	a0aceb27-a953-402e-97a	24/06/26	13:54.9	13:54.9	saludable	1	13:54.9	11765
230d7955-e47f-4ac0-9c0c	1ed07426-8885-43a3-9e9	23/06/26	17:03.7	17:03.7	defectuoso	1	17:03.7	4189
23d97cf7-11a4-475f-af26-(	5b4a18ed-053c-4224-908	25/06/26	30:36.5	30:36.5	defectuoso	1	30:36.5	2812
2f4a2e53-c9a3-4a24-b087	5ea24d6d-9a02-45c8-abb	25/06/26	06:57.2	06:57.2	saludable	1	06:57.2	6389
3066f456-f012-47c3-82bf-1	ff481682-6d0a-42ce-b069	24/06/26	48:15.9	48:15.9	defectuoso	1	48:15.9	4381
32ba4a2e-b660-4132-936c	a0aceb27-a953-402e-97a	24/06/26	26:11.3	26:11.3	saludable	1	26:11.3	5028
33d7c28b-821c-416b-a011	5ea24d6d-9a02-45c8-abb	25/06/26	45:12.3	45:12.3	saludable	1	45:12.3	4718
38466309-b107-46c3-a75a	5ea24d6d-9a02-45c8-abb	25/06/26	49:49.6	49:49.6	saludable	1	49:49.6	2340
3af7f4e5-a641-4a04-8939-	a0aceb27-a953-402e-97a	24/06/26	23:51.0	23:51.0	saludable	1	23:51.0	4308
4213c06d-86c6-441a-a71c	1ed07426-8885-43a3-9e9	23/06/26	11:48.6	11:48.6	saludable	1	11:48.6	5383
473aa0fb-e257-4066-b7fb-	134e99e3-cd4e-4d73-8ef	23/06/26	16:09.7	16:09.7	defectuoso	1	16:09.7	6430

Nota. Los registros completos de ejecuciones se presentan en el Anexo F. Elaboración propia (2026).

A partir del conjunto completo de las 150 ejecuciones, la Tabla 23 consolida los indicadores de rendimiento operativo del sistema.

Tabla 23:

Indicadores de rendimiento operativo del sistema ROCOBOT — pruebas piloto (n=150)

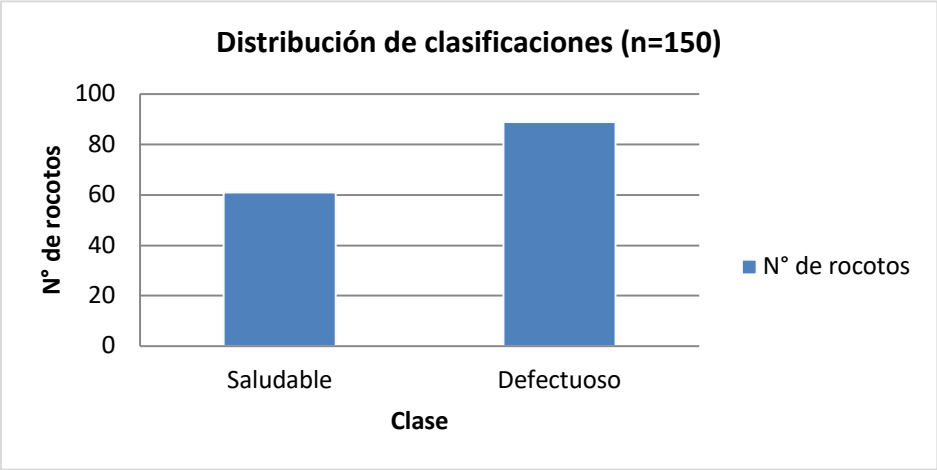
Indicador	Valor
Total de ejecuciones	150
Rocotos saludables clasificados	61 (40.7%)
Rocotos defectuosos clasificados	89 (59.3%)
Confianza promedio del modelo	97.07%
Confianza promedio — clase saludable	98.93%
Confianza promedio — clase defectuoso	95.80%
Ejecuciones con confianza ≥ 90%	143 (95.3%)
Tiempo de procesamiento promedio	15,400 ms
Tiempo de procesamiento mínimo	2,340 ms
Tiempo de procesamiento máximo	51,870 ms
Mediana del tiempo de procesamiento	11,278 ms

Nota. Pruebas ejecutadas entre el 23 y 25 de junio de 2026. Elaboración propia (2026).

Estos indicadores se presentan de forma gráfica en las Figuras 71 a 74, elaboradas a partir del análisis de la base de datos en Excel. La Figura 71 presenta la distribución de las 150 clasificaciones asignadas por el modelo durante las pruebas piloto.

Figura 71:

Distribución de clasificaciones obtenidas en las pruebas piloto (n=150)

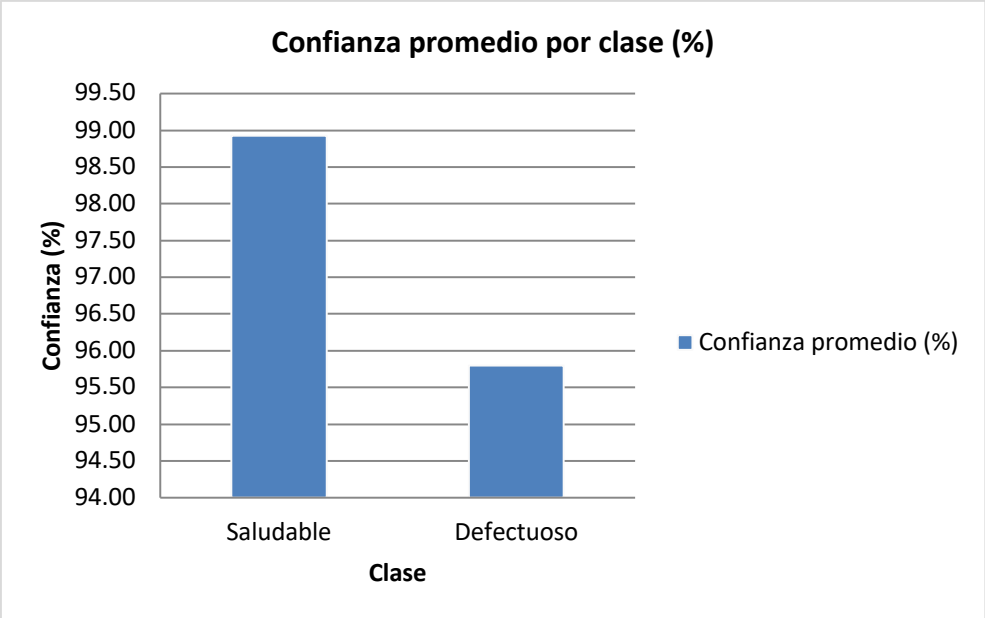


Nota. Elaboración propia (2026).

Del total de 150 rocotos procesados, el modelo clasificó 89 como defectuosos (59.3%) y 61 como saludables (40.7%). Esta distribución corresponde a lo que el modelo predijo, no a la composición real del lote — distinción que se retoma en la sección de validación contra el criterio experto. La Figura 72 presenta la confianza promedio del modelo diferenciada por clase de clasificación.

Figura 72:

Confianza promedio del modelo por clase — pruebas piloto (n=150)

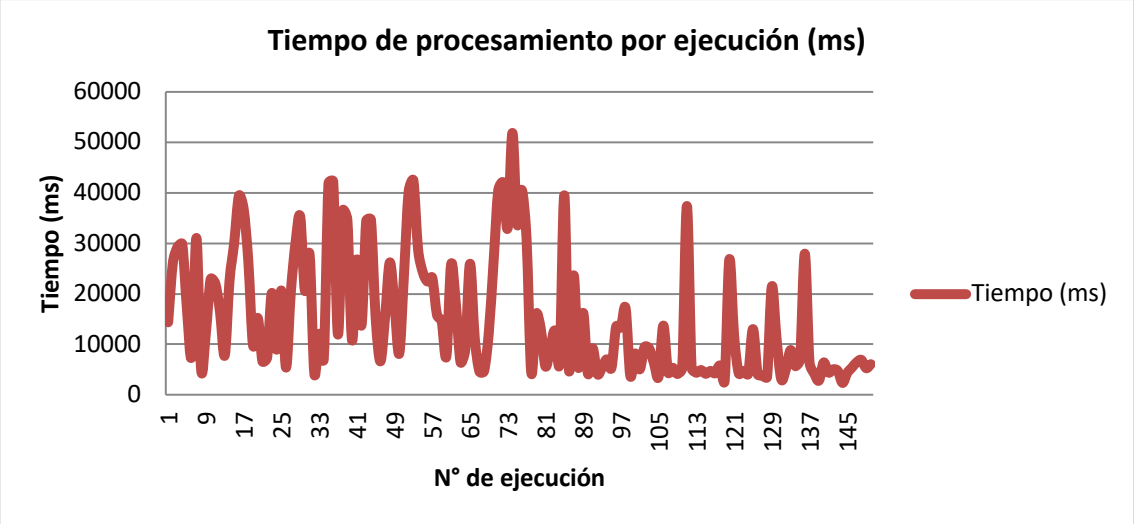


Nota. Elaboración propia (2026).

La clase saludable obtuvo una confianza promedio de 98.93% y la clase defectuoso de 95.80%, con una confianza global de 97.07%. El modelo alcanzó confianza igual o superior al 90% en 143 de las 150 ejecuciones (95.3%), evidenciando alta consistencia interna en sus predicciones. Esta confianza es una métrica que el propio modelo reporta sobre su certeza, distinta del accuracy real verificado contra el criterio del experto que se presenta más adelante. La Figura 73 presenta el tiempo de procesamiento registrado en cada una de las 150 ejecuciones.

Figura 73:

Tiempo de procesamiento por ejecución — pruebas piloto (n=150)



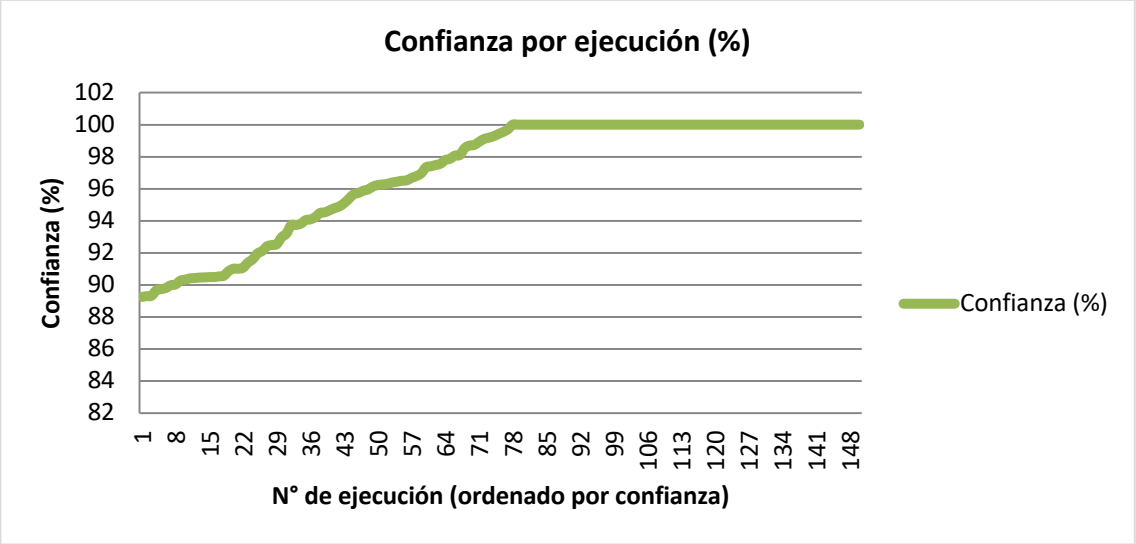
Nota. Elaboración propia (2026).

El tiempo promedio de procesamiento fue de 15,400 ms, con una mediana de 11,278 ms, un mínimo de 2,340 ms y un máximo de 51,870 ms. El 48.7% de las ejecuciones se completó en menos de 10 segundos. La variabilidad en los tiempos responde principalmente al tamaño morfológico del rocoto y a su interacción con el mecanismo físico del prototipo: los frutos de menor tamaño presentan mayor probabilidad de atascarse en los canales de separación, prolongando su tiempo de procesamiento, mientras que los frutos de mayor volumen se desplazan con facilidad a través del mecanismo, siendo procesados en menor tiempo. Este comportamiento es consistente con las limitaciones esperables en prototipos de bajo costo con hardware embebido, similares a las reportadas por Ismail y Malik (2022).

Finalmente, la Figura 74 presenta el nivel de confianza registrado en cada ejecución, ordenado de menor a mayor para visualizar su rango de variación.

Figura 74:

Confianza por ejecución — pruebas piloto (n=150)



Nota. Elaboración propia (2026).

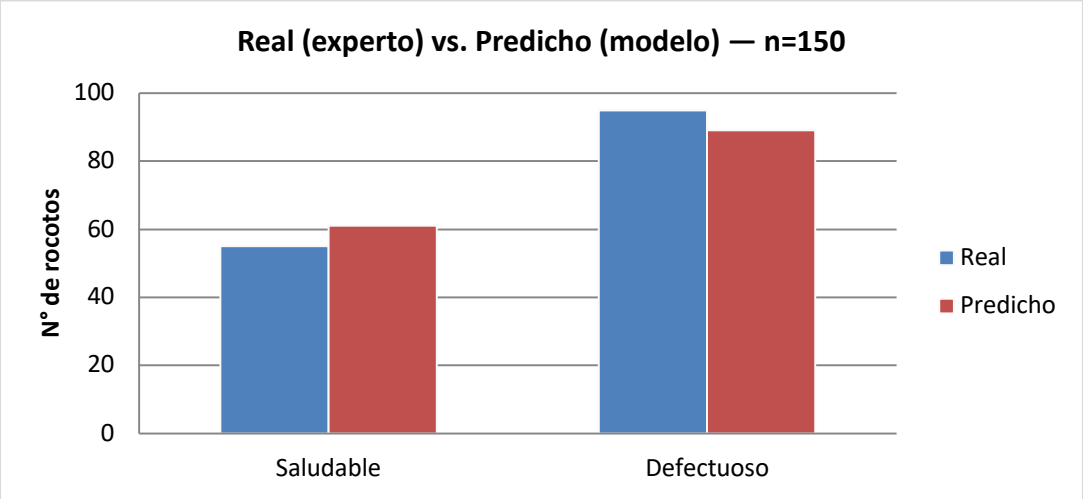
El 83.3% de las ejecuciones (125 de 150) registró una confianza igual o superior a 92%, y ninguna ejecución registró una confianza inferior a 89.3% (el valor mínimo observado), lo que confirma que el pipeline YOLOv8n + ResNet18 opera de forma consistente en condiciones reales de funcionamiento sobre hardware embebido Raspberry Pi 4B, sin casos de baja confianza extrema. Sin embargo, esta y las métricas anteriores describen el comportamiento del modelo desde su propia perspectiva, y no constituyen por sí solas una validación de su exactitud, ya que la confianza es una medida que el modelo reporta sobre su propia certeza, no una verificación externa de acierto. Para evaluar el rendimiento real del pipeline completo en condiciones operativas era necesario contrastar estos resultados contra un criterio independiente.

4.5.4. Validación contra el criterio experto

Con ese fin, los 150 rocotos evaluados fueron adicionalmente verificados por el experto propietario del producto, quien determinó el estado real de cada lote de forma independiente a la clasificación del modelo. La Figura 75 compara ambos conjuntos de totales.

Figura 75:

Comparación de clasificación real vs. predicha — pruebas piloto (n=150)

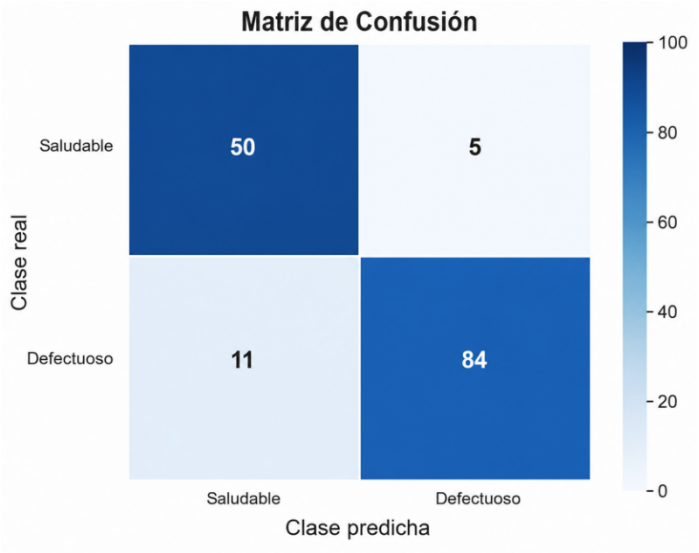


Nota. Elaboración propia (2026).

El experto verificó 55 rocotos saludables y 95 defectuosos, frente a los 61 saludables y 89 defectuosos clasificados por el modelo. El sistema sobreestimó la clase saludable en 6 unidades y subestimó la clase defectuoso en la misma magnitud, indicando una tendencia a aceptar como saludables algunos rocotos que en realidad presentaban defectos. A partir de estos totales y del número de aciertos confirmado por el experto (134 de 150), se construyó la matriz de confusión del sistema, presentada en la Figura 76.

Figura 76:

Matriz de confusión de las pruebas piloto (n=150)



Nota. Elaboración propia (2026).

Matriz derivada de los totales confirmados por el experto (real 55/95, predicho 61/89, aciertos 134/150), única solución matemáticamente consistente con dichos totales, dado que la verificación experta se realizó sobre los conteos totales al cierre de las sesiones de prueba, no mediante trazabilidad individual por rocoto.

La matriz muestra 50 verdaderos positivos, 5 falsos negativos, 11 falsos positivos y 84 verdaderos negativos, resultando el sistema más propenso a aceptar producto defectuoso (11 casos, 11.58% de los defectuosos reales) que a rechazar producto saludable (5 casos, 9.09% de los saludables reales) —comportamiento inverso al observado en el modelo ResNet18 sobre el conjunto de prueba de laboratorio (FN=0, Tabla 16), lo que evidencia que las condiciones reales de operación introducen una fuente de error que el conjunto de entrenamiento no representó completamente. A partir de esta matriz, la Tabla 24 resume las métricas de clasificación resultantes.

Tabla 24:

Métricas de validación de las pruebas piloto — verificación experta (n=150)

Clase	Precision	Recall	F1-score	Support
defectuosos	94.38%	88.42%	91.32%	95
saludables	81.97%	90.91%	86.21%	55
accuracy			89.33%	150
macro avg	88.17%	89.67%	88.76%	150

Nota. Elaboración propia (2026).

El sistema ROCOBOT alcanza así un accuracy de 89.33% (134 de 150 rocotos correctamente clasificados) frente al criterio del experto, con un F1-score de 86.21% para la clase saludable y 91.32% para la clase defectuoso.

4.5.5. Análisis comparativo de la validación piloto

Habiendo establecido el desempeño real del sistema, corresponde contrastarlo con los antecedentes presentados al inicio de este apartado, para determinar su posición relativa frente al estado del arte en validación piloto de sistemas integrados. La Tabla 25 resume esta comparación.

Tabla 25:

Comparación del accuracy de las pruebas piloto vs. Antecedentes

Autor/Año	Producto	N° pruebas	Métrica reportada	Valor
Abdal et al. (2024)	Objetos genéricos	100	Precisión	89.00%
Ismail & Malik (2022)	Manzana/Plátano	62	Precisión	96.70% / 93.80%
Tan et al. (2025)	Manzana	60	Precisión	97.00% / 100.00%
ROCOBOT (pruebas piloto)	Rocoto (Capsicum pubescens)	150	Accuracy	89.33%

Nota. Elaboración propia (2026).

El accuracy de 89.33% obtenido por ROCOBOT en campo resulta prácticamente equivalente al 89% de precisión reportado por Abdal et al. (2024) con Raspberry Pi 4B en clasificación por color y tamaño —el antecedente más cercano en hardware y en número de pruebas ejecutadas—, aunque inferior al 96.7%/93.8% de Ismail y Malik (2022) y al 97%/100% de Tan et al. (2025). Respecto a este último, cabe señalar que su 100% de precisión se alcanzó con detección de múltiples lados mediante rotación controlada de la manzana sobre una plataforma giratoria y sobre solo 60 muestras, condiciones más controladas que el flujo continuo en banda transportadora de 150 rocotos empleado en ROCOBOT; con detección de un solo lado —condición más comparable a la de ROCOBOT—, Tan et al. (2025) obtuvo 97%, aún superior al 89.33% de ROCOBOT, lo que sugiere una oportunidad de mejora mediante mecanismos de captura multi-ángulo. Cabe destacar que ninguno de estos antecedentes reporta la matriz de confusión completa de su validación en campo, limitándose a una cifra agregada de precisión; ROCOBOT es, entre los antecedentes revisados, el único sistema que reporta el desglose completo de verdaderos y falsos positivos/negativos verificado contra un experto humano en condiciones reales, además de superar en volumen de pruebas (150) a los tres antecedentes revisados.

En conjunto, las pruebas piloto ejecutadas con 150 rocotos (Capsicum pubescens) confirman el funcionamiento integrado del sistema ROCOBOT en condiciones operativas reales: el prototipo físico, el pipeline YOLOv8n + ResNet18, el aplicativo web ROCOBOT Web y la base de datos PostgreSQL operaron de forma ininterrumpida y coordinada durante

las tres sesiones de prueba, con una confianza promedio de 97.07% y un accuracy real, verificado por el experto, de 89.33%. La brecha entre este valor y el 99.64% obtenido en el conjunto de prueba de laboratorio (Tabla 16), así como frente al 97%-100% de Tan et al. (2025), evidencia la diferencia esperable entre validación controlada y despliegue en campo, y respalda la recomendación de ampliar el dataset con mayor variabilidad de condiciones reales de captura, así como de evaluar mecanismos de captura multi-ángulo. Se validó así, mediante pruebas piloto con rocotos, el funcionamiento integrado del sistema robótico ROCOBOT, evaluando el rendimiento del pipeline completo de detección y clasificación automática en condiciones operativas reales. Los registros completos de las 150 ejecuciones se presentan en el Anexo F.

5. DISCUSIÓN DE RESULTADOS

Se construyeron dos datasets de imágenes de rocotos (*Capsicum pubescens*) recolectados con una frecuencia aproximada mensual entre agosto de 2023 y diciembre de 2025 en el establecimiento del proveedor distribuidor ubicado en el distrito de José Leonardo Ortiz, Chiclayo, con rocotos provenientes de la región Amazonas, Perú: un dataset de detección de 5,421 imágenes y un dataset de clasificación balanceado de 2,732 imágenes (1,366 por clase). El dataset de detección parte de un conjunto base de 2,725 imágenes, sobre el cual se aplicaron técnicas de *data augmentation* (rotaciones, cambios de brillo y volteo horizontal) exclusivamente sobre el subconjunto de entrenamiento, manteniendo los subconjuntos de validación y prueba intactos con imágenes originales sin aumentar, con el fin de evitar una estimación optimista y no representativa del desempeño real del modelo ante datos no vistos (*data leakage*); adicionalmente, se incorporaron 800 imágenes negativas para fortalecer la capacidad discriminativa del modelo ante fondos complejos. El dataset de clasificación, en cambio, no fue sometido a *data augmentation*, dado que su objetivo es evaluar la capacidad del modelo para discriminar entre clases a partir de imágenes representativas del entorno real de operación. Respecto al tamaño del dataset de detección, Abarquez et al. (2024) emplearon 5,000 imágenes de pimientos, Fan et al. (2023) utilizaron 4,694 imágenes de malezas, Agegnehu et al. (2023) trabajaron con 3,193 imágenes de pimiento, Moya et al. (2025) con 2,145 imágenes de tomate, Hou et al. (2024) con 470 imágenes de tomate, Bazame et al. (2023) con 387 imágenes de café y Ibrahim et al. (2023) con 300 imágenes de chile, siendo el dataset de detección ROCOBOT el de mayor volumen entre todos los antecedentes que emplean arquitecturas YOLO, incluso considerando únicamente el conjunto base previo al aumento de datos (2,725 imágenes). Respecto al dataset de clasificación, Dairath et al. (2023) emplearon 16,643 imágenes, Oliullah et al. (2025) utilizaron 16,022 imágenes, Ismail y Malik (2022) trabajaron con 9,091 imágenes e Hyeon et al. (2021) con aproximadamente 1,000 imágenes de pimiento, siendo el dataset de clasificación ROCOBOT de tamaño moderado pero el único construido específicamente para *Capsicum pubescens* en el contexto de un sistema robótico físico.

Se entrenaron y compararon ocho modelos de redes neuronales convolucionales, seis de detección (YOLOv5s, YOLOv5m, YOLOv5l, YOLOv8n, YOLOv8s, YOLOv11n) y dos de clasificación (ResNet18, EfficientNetB0). En detección, YOLOv8n obtuvo el mayor mAP@0.5:0.95 = 81.30% y fue seleccionado como modelo del sistema; todos los modelos propuestos superan el mAP@0.5 de 90.4% reportado por Ouyang et al. (2022) con YOLOv5

para tomate, y la precisión de 96.20% de YOLOv8n supera el 95.9% de Abarquez et al. (2024) con YOLOv8 para pimientos de la misma familia botánica *Capsicum*. En clasificación, ResNet18 alcanzó 99.64% de accuracy con 1 FP (0.72%)**,** resultado comparable al 99.61% de Hyeon et al. (2021) con ResNet50 para pimientos mediante fine-tuning, superior al 99.2% de Ismail y Malik (2022) con EfficientNet para frutas, y superior al 96.75% de Dairath et al. (2023) con EfficientNet sobre un dataset siete veces mayor en hardware embebido Raspberry Pi, demostrando la efectividad de la estrategia de fine-tuning por fases para dominios agrícolas con datasets de tamaño moderado.

Se diseñó, construyó e implementó el primer prototipo robótico documentado para la clasificación automática de *Capsicum pubescens*, integrando una faja transportadora con rodillos, cámara Raspberry Pi Module 3 Wide (12 MP), tira LED en caja cerrada, tres servomotores MG995 y procesamiento embebido en Raspberry Pi 4B. En comparación con los antecedentes, Hou et al. (2024) emplearon Raspberry Pi 4B con cinta transportadora y brazo robótico para tomates; Dairath et al. (2023) utilizaron Raspberry Pi con brazo robótico de 4 grados de libertad y servomotores para frutos mixtos; Sanwar y Ahmed (2023) implementaron faja transportadora con rodillos controlada por Arduino Uno para objetos genéricos; y Abdal et al. (2024) emplearon Raspberry Pi 4B con cinta transportadora y servomotor para clasificación por color y tamaño. ROCOBOT es el único prototipo entre los antecedentes revisados que integra simultáneamente faja transportadora con rodillos, procesamiento embebido con Deep Learning y tres mecanismos de clasificación física independientes para *Capsicum pubescens*. El prototipo fue dimensionado para un rango de diámetro de fruto de 4.0 a 8.4 cm (Anexo G.1), valor que cubre el máximo diámetro reportado experimentalmente para la especie (8.37 cm; Huacachino Zavala & Ripa Huacachino, 2023). Si bien el ecotipo de la selva central es reconocido como el de mayor tamaño en el país (Yapo & Pacheco, 2023; Oliva et al., 2019), no se dispone de un valor de diámetro específico para dicho ecotipo en la literatura consultada, quedando su verificación dimensional como línea de trabajo futura. Para frutos en el extremo inferior del rango de diseño, en cambio, las pruebas piloto confirmaron una mayor probabilidad de atasco en los canales de separación, constituyendo la principal limitación identificada del sistema.

Se desarrolló el aplicativo web ROCOBOT Web (*https://rocobot.vercel.app*), con backend FastAPI + Railway y frontend React 19 + Vercel, que permite la supervisión remota, el monitoreo en tiempo real con transmisión de video y detecciones activas, y la consulta de reportes históricos con trazabilidad completa en base de datos PostgreSQL. Ouyang et al.

(2022) incorporaron una interfaz gráfica para supervisión y control del prototipo robótico para tomate, siendo el único antecedente que reporta un componente de monitoreo; sin embargo, no incluye supervisión remota, transmisión de video en tiempo real ni trazabilidad en base de datos. Ninguno de los demás antecedentes revisados, tales como Hou et al. (2024), Dairath et al. (2023), Abdal et al. (2024) e Ismail y Malik (2022), reporta un componente equivalente para sistemas de clasificación de frutos con hardware embebido, siendo este el aporte diferenciador del sistema ROCOBOT respecto al estado del arte revisado.

Las pruebas piloto ejecutadas con el sistema ROCOBOT integrado sobre 150 rocotos (*Capsicum pubescens*) en condiciones controladas alcanzaron un accuracy real de 89.33% (134 de 150 clasificaciones correctas), verificado contra el criterio del experto, con una confianza promedio del modelo de 97.07% y el 95.3% de las ejecuciones con confianza igual o superior al 90%. Este accuracy es inferior al 99.64% obtenido por ResNet18 y al 96.20% de precisión de YOLOv8n en el conjunto de prueba de laboratorio (Tabla 16), diferencia explicable porque esta cifra ya no evalúa un modelo de forma aislada bajo condiciones controladas de validación, sino el desempeño del pipeline completo integrado —detección YOLOv8n, clasificación ResNet18, mecanismo físico de separación, hardware embebido Raspberry Pi 4B y condiciones reales y variables del producto— operando de forma continua en campo. Bajo ese criterio, el 89.33% obtenido constituye un resultado favorable, consistente con la brecha esperable entre validación en laboratorio y despliegue real reportada en la literatura. Abdal et al. (2024) ejecutaron 100 pruebas con Raspberry Pi 4B sobre cinta transportadora obteniendo 89.00% de precisión; Ismail y Malik (2022) validaron su prototipo con Raspberry Pi alcanzando precisiones en tiempo real de 96.7% y 93.8% para manzanas y plátanos respectivamente. El accuracy de 89.33% de ROCOBOT resulta prácticamente equivalente al 89.00% de Abdal et al. (2024) —el antecedente más cercano en hardware y número de pruebas ejecutadas— y, aunque inferior a Ismail y Malik (2022), valida el rendimiento operativo del pipeline integrado YOLOv8n + ResNet18 en condiciones reales de funcionamiento con hardware embebido de bajo costo.

6. CONCLUSIONES

Se desarrolló el prototipo de sistema robótico ROCOBOT para la detección y clasificación automática del estado del rocoto (*Capsicum pubescens*) mediante técnicas de Deep Learning integradas con visión artificial, procesamiento embebido y un aplicativo web para la supervisión remota del proceso, logrando la integración funcional del pipeline de inteligencia artificial, el prototipo físico y la plataforma de monitoreo en un sistema único y operativo.

Se construyeron dos datasets de imágenes de rocotos (*Capsicum pubescens*) recolectados con una frecuencia aproximada mensual durante dos años (agosto 2023 – diciembre 2025) en condiciones reales de distribución agroindustrial en la región Amazonas: un dataset de detección de 5,421 imágenes, obtenido a partir de un conjunto base de 2,725 imágenes originales mediante filtro de similitud, con etiquetado en formato YOLO, *data augmentation* aplicado exclusivamente al subconjunto de entrenamiento e incorporación de 800 imágenes negativas; y un dataset de clasificación balanceado de 2,732 imágenes (1,366 por clase).

Se entrenaron y compararon seis modelos de detección (YOLOv5s, YOLOv5m, YOLOv5l, YOLOv8s, YOLOv11n y YOLOv8n) y dos modelos de clasificación (ResNet18 y EfficientNetB0). De los modelos de detección, YOLOv5s (mAP@0.5:0.95 = 80.16%), YOLOv5m (79.87%), YOLOv5l (80.40%), YOLOv8s (79.30%), YOLOv11n (79.40%) y YOLOv8n (81.30%), YOLOv8n obtuvo el mayor mAP@0.5:0.95 y fue seleccionado como modelo de detección del sistema ROCOBOT. En la etapa de clasificación, ResNet18 superó a EfficientNetB0 en todas las métricas evaluadas (99.64% vs 95.68% de accuracy), siendo determinante la menor tasa de falsos positivos de ResNet18 (FP = 1, 0.72%) frente a los 2 falsos positivos de EfficientNetB0 (1.44%), garantizando que ROCOBOT comete significativamente menos errores del tipo más crítico para el control de calidad: dejar pasar un rocoto defectuoso como saludable**.** La estrategia de fine-tuning por fases demostró ser efectiva para adaptar ambas arquitecturas al dominio del rocoto, alcanzando val_acc = 99.82% y AUC = 0.9999 en validación en solo 20 épocas totales para ResNet18.

Se diseñó, construyó e implementó el prototipo robótico ROCOBOT integrando una faja transportadora con rodillos, cámara Raspberry Pi Module 3 Wide (12 MP), tira LED en caja cerrada, tres servomotores MG995 de clasificación física y procesamiento embebido en Raspberry Pi 4B, logrando la integración funcional de los componentes mecánicos, electrónicos y de cómputo del sistema en un único prototipo operativo.

Se desarrolló el aplicativo web ROCOBOT Web (*https://rocobot.vercel.app*), con backend FastAPI + Railway y frontend React 19 + Vercel, que permite la supervisión remota, el monitoreo en tiempo real con transmisión de video y detecciones activas, y la consulta de reportes históricos con trazabilidad completa en base de datos PostgreSQL.

Se validó, mediante pruebas piloto ejecutadas sobre 150 rocotos (*Capsicum pubescens*) en condiciones controladas, el funcionamiento integrado del sistema robótico ROCOBOT, obteniendo un accuracy de 89.33% frente al criterio del experto (134 de 150 clasificaciones correctas), una confianza promedio del modelo de 97.07% y una mediana de tiempo de procesamiento de 11,278 ms (~11.3 s) por rocoto, confirmando el correcto funcionamiento coordinado del pipeline de detección y clasificación automática, el prototipo físico y el aplicativo web en condiciones operativas reales.

7. RECOMENDACIONES

Emplear el sistema ROCOBOT en su conjunto como línea base para nuevos proyectos de tesis vinculados a la clasificación agroindustrial mediante visión artificial, considerando que los resultados obtenidos en la integración completa del pipeline —detección, clasificación, prototipo físico y aplicativo web, con un accuracy de 89.33% verificado en campo y funcionamiento ininterrumpido durante las tres sesiones de prueba— demuestran la viabilidad técnica de esta arquitectura como punto de partida replicable para futuras investigaciones.

Emplear las arquitecturas de detección y clasificación seleccionadas en la presente investigación (YOLOv8n con mAP@0.5:0.95 = 81.30% y ResNet18 con accuracy = 99.64%) como referencia para futuras investigaciones relacionadas con otros productos de la misma familia botánica del rocoto (*Capsicum pubescens*), tales como pimientos, ajíes y chiles, dado que ambas arquitecturas, evaluadas de forma individual, obtuvieron resultados sobresalientes y consistentes en sus respectivas tareas. Cabe precisar que los modelos entrenados en este trabajo no son directamente aplicables a otros productos, al estar ajustados específicamente a las características morfológicas del rocoto; por ello, se recomienda entrenar nuevamente ambas arquitecturas desde el enfoque de *transfer learning* sobre un dataset propio de la especie o variedad objetivo, sin requerir cambios estructurales en el diseño del pipeline.

Desarrollar una segunda versión (v2) del prototipo ROCOBOT que incorpore mejoras en los componentes mecánico, electrónico y de captura identificados durante la validación piloto, tales como un mecanismo de control de flujo en la faja transportadora, captura multi-ángulo del fruto, una estructura de rodillos con separación ajustable para procesar rango de diámetros de fruto mayores a 8.4 cm (variedades de mayor tamaño, como el rocoto de la selva central) y una estructura física más robusta, con el fin de reducir la brecha observada entre el desempeño en laboratorio y en condiciones reales de operación.

Realizar una prueba de transferencia tecnológica del aplicativo web ROCOBOT Web con productores reales del sector agroindustrial del rocoto, dado que la plataforma se encuentra completamente funcional y desplegada en producción (backend FastAPI + Railway, frontend React + Vercel, con monitoreo en tiempo real y trazabilidad en base de datos PostgreSQL); esta prueba permitiría validar la usabilidad del sistema desde la perspectiva del usuario final, recoger retroalimentación sobre los indicadores del dashboard y el cuadro de mando, y sentar las bases para una eventual adopción del sistema como herramienta de control de calidad en la cadena productiva del rocoto peruano.

8. REFERENCIAS

Abarquez, H. R. S., Agsolid, A. M. N., Oliveros, J. A., Sebastian, A. Y. F., Transfiguracion, S. L. L., Magsumbol, J.-A. V., Tubola, O. D., & Rosales, M. A. (2024). SiliAI: AI-driven robotic system for efficient chili pepper sorting using YOLOv8. *2024 IEEE International Conference on Imaging Systems and Techniques (IST)*, 1–6.https://ieeexplore.ieee.org/abstract/document/10759175

Abdal, A., Al-Daleh, A., Al-Dukhi, M., Al-Jassam, N., & AlKandari, N. (2024). *Image processing-based sorting system* (Bachelor's capstone project, American University of Kuwait). American University of Kuwait Repository. American University of Kuwait Repository

Agegnehu, Y., Salau, A. O., Abuhayi, B. M., Mussa, A. A., & Ayalew, A. M. (2023). *CPD-CCNN: Classification of pepper disease using a concatenation of convolutional neural network models*. *Scientific Reports, 13*, 15581. https://doi.org/10.1038/s41598-023-42843-2

Anderson, D. J., & Carmichael, A. (2016). *Essential Kanban condensed*. Lean Kanban University Press.

Bass, L., Clements, P., & Kazman, R. (2021). *Software architecture in practice* (4.ª ed.). Addison-Wesley.

Bazame, H. C., Molin, J. P., Althoff, D., & Martello, M. (2023). Detection of coffee fruits on tree branches using computer vision. Scientia Agricola, 80, e20220064. https://doi.org/10.1590/1678-992x-2022-0064

Biswas, S., & Simon, J. (2023). TensorFlow for deep learning applications: Advances and future trends. *Journal of Artificial Intelligence Research, 76*, 1123-1145.

Bolton, W. (2015). Mechatronics: Electronic control systems in mechanical engineering (6th ed.). Pearson.

Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer vision with the OpenCV library. O'Reilly Media.

Clement, J., Novas, N., Gazquez, J. A., Manzano-Agugliaro, F., & García-Cruz, A. (2012). *High speed intelligent classifier of tomatoes by colour, size and weight*. Spanish Journal of Agricultural Research, 10(2), 314–325. https://doi.org/10.5424/sjar/2012102-1994

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). *CRISP-DM 1.0: Step-by-step data mining guide*. SPSS Inc. https://www.the-modeling-agency.com/crisp-dm.pdf

Chollet, F. (2015). *Keras*. https://keras.io

Codd, E. F. (1970). *A relational model of data for large shared data banks*. *Communications of the ACM, 13*(6), 377–387. https://doi.org/10.1145/362384.362685

Dairath, M. H., Akram, M. W., Mehmood, M. A., Sarwar, H. U., Akram, M. Z., Omar, M. M., & Faheem, M. (2023). *Computer vision-based prototype robotic picking cum grading system for fruits*. Smart Agricultural Technology, 4, 100210. https://doi.org/10.1016/j.atech.2023.100210

Dhanush, G., Khatri, N., Kumar, S., & Shukla, P. K. (2023). A comprehensive review of machine vision systems and artificial intelligence algorithms for the detection and harvesting of agricultural produce. *Scientific African*, *21*(e01798), e01798. https://doi.org/10.1016/j.sciaf.2023.e01798

ElectroPeak. (2019). *FC-51 infrared obstacle avoidance sensor module*. https://electropeak.com/learn/

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision, 88*(2), 303–338. https://doi.org/10.1007/s11263-009-0275-4

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters, 27*(8), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

Fan, X., Chai, X., Zhou, J., & Sun, T. (2023). Deep learning-based weed detection and target spraying robot system at seedling stage of cotton field. *Computers and Electronics in Agriculture*, *214*(108317), 108317. https://doi.org/10.1016/j.compag.2023.108317

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures* [Tesis doctoral, University of California, Irvine].

Fowler, M. (2015). *Monolith first*. martinfowler.com. https://martinfowler.com/bliki/MonolithFirst.html

Gestión. (2018, September 8). *Google: Estos son los productos peruanos más buscados en el extranjero*. Gestión. https://gestion.pe/economia/son-productos-peruanos-buscados-extranjero-nndc-243853-noticia

Gonzalez, R. C., & Woods, R. E. (2018). *Digital image processing* (4th ed.). Pearson.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. https://www.deeplearningbook.org/

He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep residual learning for image recognition*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778. https://doi.org/10.1109/CVPR.2016.90

Hernández Sampieri, R., & Mendoza Torres, C. P. (2018). *Metodología de la investigación: Las rutas cuantitativa, cualitativa y mixta*. McGraw-Hill.

Hespeler, S. C., Nemati, H., & Dehghan-Niri, E. (2021). Non-destructive thermal imaging for object detection via advanced deep learning for robotic inspection and harvesting of chili peppers. Artificial Intelligence in Agriculture, 5, 102–117. https://doi.org/10.1016/j.aiia.2021.05.003

Hou, L., Liu, Z., You, J., Liu, Y., Xiang, J., Zhou, J., & Pan, Y. (2024). Tomato sorting system based on machine vision. *Electronics*, *13*(11), 2114. https://doi.org/10.3390/electronics13112114

Huacachino Zavala, D., & Ripa Huacachino, Y. (2023). *Respuesta de dos variedades de rocoto (Capsicum pubescens L.) a la aplicación de cuatro bioestimulantes orgánicos en el distrito de Colpas, provincia de Ambo, región Huánuco* [Tesis de pregrado, Universidad Nacional Daniel Alcides Carrión]. Repositorio Institucional UNDAC.

Humble, J., & Farley, D. (2010). *Continuous delivery: Reliable software releases through build, test, and deployment automation*. Addison-Wesley.

Hyeon, J., Kim, M., Lee, J., & Park, S. (2021). Deep learning-based diagnosis of diseases and pests in hot pepper plants using VGG and ResNet architectures. *Sensors, 21*(9), 3124. https://doi.org/10.3390/s21093124

Ibrahim, M. F., Zainudin, M. S., Idris, M. I., Saad, W. M., Kamarudin, M. R., Sulaiman, N. A., & Mohamed, R. (2023). Detection and Classifying of Chili Fruits Variations using YOLOv5. KZYJC, 38(4), 1363-1376. https://doi.org/10.13140/RG.2.2.24901.83685

Ismail, N., & Malik, O. A. (2022). Real-time visual inspection system for grading fruits using computer vision and deep learning techniques. *Information Processing in Agriculture*, *9*(1), 24–37. https://doi.org/10.1016/j.inpa.2021.01.005

Kalpakjian, S., & Schmid, S. (2014). *Manufacturing processes for engineering materials* (5th ed.). Pearson.

Kamboj, D. S., & Diwan, A. (2019). *Development of automatic sorting conveyor belt using PLC*. International Journal of Mechanical Engineering and Technology, 10(8), 124–132.

Khalifa, N. E., Loey, M., & Mirjalili, S. (2022). A comprehensive survey of recent trends in deep learning for digital images augmentation. *Artificial Intelligence Review*, *55*(3), 2351–2377. https://doi.org/10.1007/s10462-021-10066-4

Khort, D. O., Kutyrev, A., Smirnov, I., Andriyanov, N., Filippov, R., Chilikin, A., Astashev, M. E., Molkova, E. A., Sarimov, R. M., Matveeva, T. A., & Gudkov, S. V. (2024). Enhancing sustainable automated fruit sorting: Hyperspectral analysis and machine learning algorithms. *Sustainability*, *16*(22), 10084. https://doi.org/10.3390/su162210084

LeCun, Y., Bengio, Y., & Hinton, G. (2015). *Deep learning*. **Nature, 521**(7553), 436–444. https://doi.org/10.1038/nature14539

Li, J., Li, Z., Wang, X., & Zhang, H. (2022). Robotic harvesting of clustered berries using YOLOv4 and Deep SORT. *Biosystems Engineering, 215*, 168–181.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, **C. L.** (2014). Microsoft COCO: Common objects in context. En D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer Vision – ECCV 2014* (pp. 740–755). Springer. https://doi.org/10.1007/978-3-319-10602-1_48

Martin, R. C. (2017). *Clean architecture: A craftsman's guide to software structure and design*. Pearson.

Ministerio de Desarrollo Agrario y Riego. (2024, 6 de septiembre). *Perú cuenta con más de 350 variedades de ajíes que se producen en costa, sierra y selva*. https://www.gob.pe/institucion/midagri/noticias/1017426-midagri-peru-cuenta-con-mas-de-350-variedades-de-ajies-que-se-producen-en-costa-sierra-y-selva

Mitchell, T. M. (1997). *Machine learning*. McGraw-Hill.

Mohi-Alden, K., Omid, M., Soltani Firouz, M., & Nasiri, A. (2023). A machine vision-intelligent modelling-based technique for in-line bell pepper sorting. *Information Processing in Agriculture*, *10*(4), 491–503. https://doi.org/10.1016/j.inpa.2022.05.003

Moya, V., Guerra, M., Pazmiño, K., Abedrabbo, F., Chicaiza, F. A., & Pozo-Espín, D. (2025). *Tomato classification with YOLOv8: Enhancing automated sorting and quality assessment*. Smart Agricultural Technology, 12, 101221. https://doi.org/10.1016/j.atech.2025.101221

Noordin, N. H., Samad, R., & Abdul Malek, A. H. (2025). *Real-time FFB ripeness detection using IoT-enabled YOLOv8n on Raspberry Pi 4 edge devices for precision agriculture*.

Journal of Mechatronics, Electrical Power, and Vehicular Technology, 16(2), 305–317. https://doi.org/10.55981/j.mev.2025.1220

Oliullah, K., Islam, M. R., Babar, J. I., Quraishi, M. A. N., Rahman, M. M., Mahbub-Or-Rashid, M., & Bhuiyan, T. M. A.-U.-H. (2025). FruVeg_MultiNet: A hybrid deep learning-enabled IoT system for fresh fruit and vegetable identification with web interface and customized blind glasses for visually impaired individuals. *Journal of Agriculture and Food Research*, *19*(101623), 101623. https://doi.org/10.1016/j.jafr.2024.101623

Oliva, M., Oliva-Cruz, M., Maicelo, J., Oliva-Cruz, J., Morales, R., Salvador, W., ..., & Barrena, M. (2019). *Caracterización morfológica de ocho ecotipos de rocoto (Capsicum pubescens Ruiz & Pav.) en la región Amazonas, Perú*. INDES Revista de Investigación para el Desarrollo Sustentable**.**

Ouyang, J., Li, Y., Zhang, S., Wang, H., & Chen, Y. (2022). *Automatic tomato disease monitoring system based on a modular mobile robot and deep learning*. Computers and Electronics in Agriculture, 198**,** 107058. https://doi.org/10.1016/j.compag.2022.107058

Padilla, R., Passos, W. L., Dias, T. L. B., Netto, S. L., & da Silva, E. A. B. (2021). *A comparative analysis of object detection metrics with a companion open-source toolkit*. **Electronics, 10**(3), 279. https://doi.org/10.3390/electronics10030279

Patel, K. K., Kar, A., Jha, S. N., & Khan, M. A. (2011). *Machine vision system: A tool for quality inspection of food and agricultural products*. *Journal of Food Science and Technology, 49*(2), 123–141. https://doi.org/10.1007/s13197-011-0321-4

PostgreSQL Global Development Group. (2023). *PostgreSQL documentation*. https://www.postgresql.org/docs/

Railway. (2023). *Railway documentation*. https://docs.railway.app/

Ramírez, S. (2021). *FastAPI documentation*. https://fastapi.tiangolo.com/

Raspberry Pi Foundation. (2023). Raspberry Pi 4 Model B. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

React. (2024). *React Documentation*. https://react.dev/

Redis. (2023). *Redis documentation*. https://redis.io/docs/

Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., & Savarese, S. (2019). Generalized intersection over union: A metric and a loss for bounding box regression. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 658–666. https://doi.org/10.1109/CVPR.2019.00075

Russell, S., & Norvig, P. (2021). *Artificial intelligence: A modern approach* (4th ed.). Pearson.

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., & Youman, C. E. (1996). Role-based access control models. *Computer, 29*(2), 38–47. https://doi.org/10.1109/2.485845

Sathyanarayanan, S., & Roopashri Tantri, B. (2024). *Confusion matrix-based performance evaluation metrics*. African Journal of Biomedical Research, 27(4S), 4023–4031.https://doi.org/10.53555/AJBR.v27i4S.4345

Sanwar, S., & Ahmed, M. I. (2023). *Automated object sorting system with real-time image processing and robotic gripper mechanism control*. Journal of Engineering Advancements, **4**(3), 70–79. https://doi.org/10.38032/jea.2023.03.003

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management, 45*(4), 427–437. https://doi.org/10.1016/j.ipm.2009.03.002

STMicroelectronics. (2000). *L298 dual full-bridge driver* [Hoja de datos]. https://www.st.com/resource/en/datasheet/l298.pdf

Szeliski, R. (2010). Computer vision: Algorithms and applications. Springer.

Tan, M., & Le, Q. (2019). *EfficientNet: Rethinking model scaling for convolutional neural networks*. Proceedings of the 36th International Conference on Machine Learning (ICML). https://arxiv.org/abs/1905.11946

Tan, W. X., Lim, S. T., Koo, V. C., & Lim, W. S.(2025). *Design and development of an automated fruit quality classifier and sorter using a robotic arm.* Edelweiss Applied Science and Technology, 9(10), 32–49. https://doi.org/10.55214/2576-8484.v9i10.10340

Tang, Y., Li, X., & Wang, H. (2022). TensorFlow 2.0: Advances in deep learning frameworks. *IEEE Transactions on Artificial Intelligence, 3*(4), 290-303.

Texas Instruments. (2016). *LM2596 step-down voltage regulator datasheet*. https://www.ti.com/

TowerPro. (2014). *MG995 high torque servo motor datasheet*. https://towerpro.com.tw/

Ulrich, K. T., & Eppinger, S. D. (2016). *Product design and development* (6th ed.). McGraw-Hill.

United Nations Environment Programme. (2021). *UNEP Food Waste Index Report 2021*. United Nations. https://www.unep.org/resources/report/unep-food-waste-index-report-2021

Ultralytics. (2025). *YOLOv5 and YOLOv8 architecture*. https://docs.ultralytics.com

Vercel. (2023). *Vercel documentation*. https://vercel.com/docs

Wang, C., Wang, H., Han, Q., Zhang, Z., Kong, D., & Zou, X. (2024). Strawberry Detection and Ripeness Classification Using YOLOv8+ Model and Image Processing Method. Agriculture, 14(5), 751. https://doi.org/10.3390/agriculture14050751

Wang, Q., Wu, D., Sun, Z., Zhou, M., Cui, D., Xie, L., Hu, D., Rao, X., Jiang, H., & Ying, Y. (2023). *Design, integration, and evaluation of a robotic peach packaging system based on deep learning*. Computers and Electronics in Agriculture, 213, 108250. https://doi.org/10.1016/j.compag.2023.108250

Wang, Z., Ling, Y., Wang, X., Meng, D., Nie, L., An, G., & Wang, X. (2022). *An improved Faster R-CNN model for multi-object tomato maturity detection in complex scenarios*. Ecological Informatics, 72(101886), 101886. https://doi.org/10.1016/j.ecoinf.2022.101886

Yapo, F. M., & Pacheco, G. A. (2023). *Manual de manejo agronómico de rocoto*. Instituto Nacional de Innovación Agraria - INIA. https://repositorio.inia.gob.pe/

9. ANEXOS

Los siguientes anexos recopilan los recursos digitales del sistema ROCOBOT: código fuente, modelos de IA, dataset, pruebas piloto, diseño mecánico, base de datos y manual de usuario, disponibles en repositorios y plataformas de acceso controlado.

Anexo A. Repositorio de código fuente

El código fuente completo del sistema ROCOBOT se encuentra alojado en el siguiente repositorio de GitHub. El repositorio contiene los tres componentes principales del sistema: el backend desarrollado con FastAPI y el frontend desarrollado con React y TypeScript.

Repositorio principal del proyecto:

https://github.com/CRISTHIAN-ODL/ai-rocoto-sorting.git

Repositorio GitHub — rama: qas

A.1. Módulo de Modelos de Inteligencia Artificial (IAModels)

Contiene los archivos de entrenamiento, los pesos de los modelos entrenados (.pt para YOLO y .pth para ResNet18 y EfficientNetB0), los notebooks de Jupyter utilizados en Google Colab y los scripts de evaluación de métricas.

Ruta en el repositorio:

https://github.com/MashaDBDevep/rocobot-IAMODELS

Modelos YOLOv5 (s/m/l), YOLOv8, YOLOv11, ResNet18 y EfficientNetB0

A.2. Módulo Backend (FastAPI

Contiene el código fuente del servidor de la API REST, incluyendo los servicios de autenticación, detección, clasificación, estadísticas y comunicación con la Raspberry Pi. Implementado con FastAPI 0.121.2, SQLAlchemy 2.0.30 y PostgreSQL.

Ruta en el repositorio:

https://github.com/CRISTHIAN-ODL/ai-rocoto-sorting/tree/qas/Rocobot/Backend

API REST FastAPI — servicios: auth, detection, classification, stats, system, camera, raspberry

A.3. Módulo Frontend (React + TypeScript)

Contiene el código fuente del aplicativo web desarrollado con React 19, TypeScript, Vite 7.2 y Fluent UI v9. Incluye los módulos de dashboard, cuadro de mando, configuraciones, autenticación y componentes reutilizables.

Ruta en el repositorio:

https://github.com/CRISTHIAN-ODL/ai-rocoto-sorting/tree/qas/Rocobot/Frontedn

Aplicativo web React SPA — desplegado en Vercel

A.4. Módulo de Control de Hardware (Raspberry Pi)

Contiene el código fuente del sistema embebido que se ejecuta directamente en la Raspberry Pi 4B. Gestiona el control de los actuadores (servomotores MG995 y motoreductor JGB37-520), la lectura de los sensores de posición (FC-51) y la comunicación con el backend FastAPI para recibir los comandos de clasificación en tiempo real.

Ruta en el repositorio:

https://github.com/CRISTHIAN-ODL/Hardware_SystemClassification.git

Control de hardware Raspberry Pi 4B — actuadores, sensores y comunicación con backend

Anexo B. Aplicativo Web en Producción

El aplicativo web del sistema ROCOBOT se encuentra desplegado en producción en la plataforma Vercel y es accesible desde cualquier navegador web moderno. La dirección URL de acceso es la siguiente:

Ruta de producción:

https://rocobot.vercel.app/login

Aplicativo web ROCOBOT — autenticación requerida para acceder al sistema

Para acceder al sistema se requieren credenciales de usuario previamente registradas por el administrador del sistema. El aplicativo incluye los módulos de dashboard con monitoreo en tiempo real, cuadro de mando con reportes estadísticos y configuración del sistema de clasificación.

Anexo C. Diccionario de Base de Datos

El diccionario de base de datos del sistema ROCOBOT documenta de manera exhaustiva los tres esquemas de la base de datos PostgreSQL: auth (autenticación y usuarios), menu (navegación y permisos) y core (producción y resultados). Incluye diagramas entidad-relación, descripción de cada tabla, tipos de datos, restricciones de integridad y relaciones entre entidades.

Documento completo (Google Drive):

https://docs.google.com/document/d/1Q9UnfETFr_2zDWRxkDJmeOU0-pPxty9m/edit

Diccionario de Base de Datos ROCOBOT v1.0 — formato PDF

Tabla 26:

Resumen de los esquemas documentados en el Diccionario de Base de Datos

Esquema	N° Tablas	Entidades principales	Propósito
auth	5	Persona, Usuario, rol_Id, usuarioRol, password_reset_token	Autenticación de usuarios, roles y recuperación de contraseñas.
menu	2	menu, rol_menu	Navegación jerárquica y permisos CRUD por rol.
core	4	ejecucion_sistema, resultado, estadistica_diaria, estado	Registro de ejecuciones, resultados individuales y estadísticas diarias.

Nota. Elaboración propia (2026).

Anexo D. Manual de Usuario del Aplicativo Web

El manual de usuario del aplicativo web ROCOBOT describe de manera detallada el procedimiento de acceso al sistema, la navegación por cada módulo de la plataforma, la interpretación de los indicadores del dashboard, la consulta de reportes estadísticos y la configuración de parámetros operativos. Está dirigido a los operarios y administradores del sistema de clasificación.

Documento completo (Google Drive):

https://docs.google.com/document/d/1va7HfBYsff6Oh3S9VDLswS4q2mo_eR6U/edit

Manual de usuario ROCOBOT — formato PDF

Anexo E. Dataset de Imágenes de Rocotos

El dataset de imágenes utilizado para el entrenamiento, validación y prueba de los modelos de inteligencia artificial se encuentra disponible en la siguiente carpeta de Google Drive. Se emplearon dos datasets independientes: uno para la etapa de detección y clasificación.

Dataset de detección — 5,421 imágenes obtenidas mediante filtro de similitud y data augmentation sobre un dataset base de 2,725 imágenes. Contiene una única clase anotada (clase 0: rocoto) en formato YOLO (.txt), organizado en carpetas train, val y test.

Dataset de clasificación — 2,732 imágenes de rocotos perfectamente balanceadas en dos clases: defectuoso (clase 0) y saludable (clase 1), con 1,366 imágenes por clase.

Acceso al dataset (Google Drive):

https://drive.google.com/drive/folders/136bzYWhHJoHnJ4AC2q8QItPbdPw-hHCR?usp=sharing

Dataset ROCOBOT — 5,421 imágenes para detección y 2,732 imágenes para clasificación

Anexo F. Registro de Pruebas Piloto

El registro completo de las 150 ejecuciones realizadas durante las pruebas piloto del sistema ROCOBOT se encuentra disponible en el siguiente documento de Google Drive. Incluye, para cada ejecución, la clasificación asignada por el modelo, el nivel de confianza y el tiempo de procesamiento registrados automáticamente en tiempo real.

Documento completo (Google Drive):

https://drive.google.com/file/d/1HITXcmC_7G3xEq5Xq4Tgj62bvA8Pa9Vy/view?usp=sharing

Registro de Pruebas Piloto ROCOBOT — 150 ejecuciones (n=150)

Anexo G. Diseño Mecánico del Prototipo

El diseño mecánico del prototipo ROCOBOT comprende dos componentes complementarios: la memoria de cálculo que fundamenta los parámetros geométricos del sistema y los planos técnicos con los diagramas, flujos y vistas 2D/3D del prototipo, disponibles en los siguientes documentos de Google Drive.

G.1. Memoria de Cálculo

Documento que determina, mediante criterios de ingeniería mecánica, los parámetros geométricos del sistema de rodillos: diámetro de rodillo, piñón de transmisión, separación entre rodillos, distancia entre centros y cinemática de rotación del fruto para la captura multiangular.

Documento completo (Google Drive):

https://docs.google.com/document/d/1zJSD6YnvnKX9W3t-fg3iQwf_nxWAtn-U/edit

Memoria de cálculo del sistema de rodillos giratorios del prototipo ROCOBOT

G.2. Planos y Diseño 2D/3D

Carpeta con los diagramas de flujo del prototipo, las vistas 2D acotadas con las medidas de cada componente y el modelo 3D del sistema, incluyendo el archivo editable para su apertura en AutoCAD.

Documento completo (Google Drive):

https://drive.google.com/drive/folders/1YvnT5MQ9aC-0fuTtyud8gmLwE7kAVRwJ?usp=sharing

Planos 2D con medidas, diseño 3D y archivo AutoCAD del prototipo ROCOBOT