UNIVERSIDAD NACIONAL MICAELA BASTIDAS DE APURÍMAC

FACULTAD DE INGENIERÍA

ESCUELA ACADÉMICO PROFESIONAL DE INGENIERÍA INFORMÁTICA Y SISTEMAS

Tesis

Machine Learning para la detección de plagas en las hojas del tomate Abancay 2022

Presentado por:

Bryanne Robert Junior Huamán Cáceres

Para optar el título de Ingeniero Informático y Sistemas

Abancay, Perú

2025

UNIVERSIDAD NACIONAL MICAELA BASTIDAS DE APURÍMAC

FACULTAD DE INGENIERÍA

ESCUELA ACADÉMICO PROFESIONAL DE INGENIERÍA INFORMÁTICA Y SISTEMAS

TESIS

Machine Learning para la detección de plagas en las hojas del tomate Abancay 2022

Presentado por **Bryanne Robert Junior Huamán Cáceres**, para optar el título de Ingeniero Informático y Sistemas

Sustentado y aprobado el 04 de diciembre del 2024 ante el jurado evaluador:

Presidente: *Mtro. Marleny Peralta Ascue*

Primer miembro: *Mag. Mario Aquino Cruz*

Segundo miembro: *Dr. Ronald Alberto Rentería Ayquipa*

Asesor(es): *Dr. Manuel Jesús Ibarra Cabrera*

Año del Fortalecimiento de la Soberanía Nacional

CONSTANCIA DE ORIGINALIDAD N°169-2026

La Universidad Nacional Micaela Bastidas de Apurímac, a través de la Unidad de Investigación de la Facultad de Ingeniería declara que, la tesis titulada: **"Machine Learning para la detección de plagas en las hojas del tomate Abancay 2022"**, presentado por el Bachiller: **Bryanne Robert Junior Huamán Cáceres**, para optar el título de **Ingeniero Informático y Sistemas;** ha sido sometido a un mecanismo de evaluación y verificación de similitud, a través del Software Turnitin, siendo el índice de **similitud ACEPTABLE** de **(9%)** por lo que, cumple con los criterios de originalidad establecidos en el reglamento de investigación, aprobado con Resolución N°168-2024(2)-CU-UNAMBA.

Abancay, 17 de junio del 2026

Atentamente,

UNIVERSIDAD NACIONAL MICAELA BASTIDAS
FACULTAD DE INGENIERÍA

Dr. Walquer Huacani Calsin
DIRECTOR DE LA UNIDAD DE INVESTIGACIÓN
FACULTAD DE INGENIERÍA

C. c:
Archivo/
WHCE/D-UIFI
YOVANA / SEC
REG. N°847

Agradecimiento

Nada de esto habría sido posible sin el amor, el esfuerzo compartido y la fe que tantas personas han depositado en mí.

A Dios, por regalarme cada día la oportunidad de crecer, aprender y abrazar a los que amo.

A mis padres, quienes con su ejemplo, constancia y cariño me enseñaron que los sueños se alcanzan con trabajo y humildad. Gracias por ser mi fuerza en silencio.

A mi esposa, mi compañera en cada desafío. Tu apoyo, tus ideas y tu inmenso corazón han sido pilares esenciales en este camino.

A mis hijos, Ethan y Anne, por ser la luz que me guía y la razón más profunda de este esfuerzo. Todo lo que hago, lo hago pensando en ustedes.

Esta tesis forma parte del II Concurso de Proyectos de Investigación en Centros Experimentales de la UNAMBA, promovido por el Vicerrectorado de Investigación y financiado con fondos del canon y regalías mineras.

Dedicatoria

A mis amados hijos, Ethan Robert y Anne Elaine Amira, quienes llenan mis días de felicidad y son mi mayor inspiración. A mis queridos padres, quienes siempre me han brindado su amor y apoyo incondicional. Finalmente, a mi esposa, con quien comparto todas las alegrías y tristezas de la vida.

Machine Learning para la detección de plagas en las hojas del tomate Abancay 2022

Línea de investigación: Ingeniería Informática, Industria y Sociedad.

ÍNDICE

ÍNDICE DE TABLAS

Pág.

ÍNDICE DE FIGURAS

Pág.

INTRODUCCIÓN

El tomate, científicamente llamado *Lycopersicum esculentum*, es considerado una de las hortalizas de gran importancia en el mundo, ya que aporta una multitud de vitaminas. La producción masiva de tomates en Abancay, Apurímac, juega un papel primordial en la dieta local, con un notable aumento en su producción en un 188,5% registrado en 2021 según el INEI (INEI 2021), sin embargo, estos cultivos están expuestos a diferentes tipos de amenazas, entre ellos, se hace referencia a las plagas, las cuales impiden que el desarrollo de la planta de tomate sea positivo, provocando daños que afecten al cultivo en general, y por ende, grandes pérdidas económicas, convirtiéndose en un problema para los agricultores.

Por otro lado, la tecnología actual representa una herramienta altamente eficiente y accesible para las personas en todo aspecto y ámbito. Por tanto, esta investigación destaca la importancia de los programas de reconocimiento de imágenes, ya sea de personas, animales u objetos. En este contexto, se han desarrollado programas específicos en los últimos años para la detección de plagas en las hojas de tomate. Por ejemplo, tenemos la tesis presentada por ERAZO VALDIVIEZO Jorge Deninson y UEMA HIGA Ken Sebastian (2020), titulado "Redes neuronales para la identificación de enfermedades en el cultivo de tomate (*Solanum lycopersicum L.*) en Honduras". Como también la investigación de GUERRERO ANDRADE Carlos Jonathan y MARTÍNEZ MOSQUERA Silvia Diana (2022), con el título "Reconocimiento de lesiones necróticas para la detección de la plaga thrips en el guisante mediante el uso del modelo de aprendizaje profundo YOLOv4-tiny". Sin embargo, es crucial destacar que estos proyectos se llevaron a cabo en contextos de plantas y plagas distintas.

Cabe indicar que, Machine Learning, o aprendizaje automático, es un área de conocimiento dentro de la Inteligencia Artificial, donde los ordenadores aplican técnicas de aprendizaje con el objetivo de identificar patrones en los datos, y a partir de ellos preparar su propia información y llegar a tomar sus propias decisiones con la mínima intervención humana.

El presente proyecto se enfocó en demostrar su objetivo principal: Comprobar la técnica de Machine Learning más eficiente para detectar las plagas en las hojas de los tomates en los cultivos de Abancay Apurímac, 2022. Para lograrlo, se inicia con la descripción del problema,

se establecen los objetivos. Se detalla la metodología empleada para obtener los resultados, su posterior interpretación y la correspondiente discusión. Finalmente, se presentan las conclusiones y recomendaciones orientadas a la implementación y optimización de estos modelos para apoyar la detección temprana de plagas, permitiendo un manejo más efectivo de los cultivos de tomate.

RESUMEN

El cultivo de tomates es esencial para la alimentación y la economía de los agricultores. Sin embargo, plagas como la "Mosca Blanca" y el "Minador de las Hojas" representan una amenaza significativa para estos cultivos. Por ello, la detección temprana de dichas plagas en las hojas de tomate es crucial para mitigar su impacto económico. La presente investigación tuvo como objetivo "Comprobar la técnica de Machine Learning más eficiente para detectar las plagas en las hojas de los tomates en los cultivos de Abancay, Apurímac, 2022." Para ello, se recopilaron imágenes utilizando una cámara fotográfica, que se usaron para entrenar dos modelos de Machine Learning: YOLOv4 y Faster R-CNN, los cuales se encargaron de identificar el tipo de plaga presente en las hojas de tomate. En total, se capturaron 1160 imágenes de hojas de tomate en Abancay, Apurímac, tanto afectadas por plagas como sanas. Estas imágenes se distribuyeron aleatoriamente en: 90.26% para entrenamiento con etiquetado de plagas, 5.43% para validación y 4.31% para pruebas. El proceso de etiquetado se realizó mediante LabelImg y Roboflow, mientras que Google Colab con Python se utilizó para el entrenamiento y la validación de los modelos, así como para el proceso de pruebas. Al comparar YOLOv4 y Faster R-CNN en la detección de plagas, Faster R-CNN se destaca en la identificación de la Mosca Blanca, con una precisión del 92% y un F1-Score de 0.96, superando ligeramente a YOLOv4, que obtuvo una precisión del 89% y un F1-Score de 0.94. No obstante, en la detección del Minador de Hojas, YOLOv4 muestra mejor desempeño, con una precisión del 83% y un F1-Score de 0.86, frente a la precisión del 81% y un F1-Score de 0.83 de Faster R-CNN. A nivel global, considerando la media armónica del F1-Score para ambas plagas, YOLOv4 alcanza un valor de 0.90, superando el 0.89 de Faster R-CNN, lo que demuestra un rendimiento más equilibrado en la detección de ambas plagas.Por tanto, se concluye que YOLOv4 es el modelo más eficiente para la detección de plagas en hojas de tomate en el contexto general de esta investigación, gracias a su balance óptimo entre precisión y exhaustividad en ambas clases de plagas, lo que lo posiciona como la opción preferida para un desempeño global robusto. No obstante, Faster R-CNN sigue siendo una opción destacada cuando se prioriza la máxima precisión y la minimización de errores.

***Palabras clave**: YOLOv4, Faster R-CNN, machine learning, plagas, tomate.*

ABSTRACT

The cultivation of tomatoes is essential for the food and the economy of farmers. However, pests such as 'Whitefly' and 'Leafminer' pose a significant threat to these crops. Therefore, early detection of these pests on tomato leaves is crucial to mitigate their economic impact. The present research aimed to 'Test the most efficient Machine Learning technique to detect pests on tomato leaves in crops in Abancay, Apurimac, 2022'. To do this, images were collected using a photographic camera, which were used to train two Machine Learning models: YOLOv4 and Faster R-CNN, which were responsible for identifying the type of pest present on tomato leaves.

In total, 1160 images were captured of tomato leaves in Abancay, Apurímac, both affected by pests and healthy. These images were randomly distributed into: 90.26% for training with pest labelling, 5.43% for validation and 4.31% for testing. The labelling process was performed using LabelImg and Roboflow, while Google Colab with Python was used for training and validation of the models, as well as for the testing process. When comparing YOLOv4 and Faster R-CNN in pest detection, Faster R-CNN stands out in the identification of Whitefly, with precision of 92% and an F1-Score of 0.96, slightly outperforming YOLOv4, which obtained a precision of 89% and an F1-Score of 0.94. However, in Leafminer detection, YOLOv4 shows better performance, with a precision of 83% and an F1-Score of 0.86, compared to 81% precision and an F1-Score of 0.83 for Faster R-CNN.

Globally, considering the harmonic mean F1-Score for both pests, YOLOv4 reaches a value of 0.90, exceeding the 0.89 of Faster R-CNN, showing a more balanced performance in the detection of both pests. Therefore, it is concluded that YOLOv4 is the most efficient model for tomato leaf pest detection in the overall context of this research, thanks to its optimal balance between precision and completeness in both pest classes, which positions it as the preferred choice for robust overall performance. However, Faster R-CNN remains an outstanding choice when maximum precision and error minimisation are prioritised.

***Keywords**: YOLOv4, Faster R-CNN, machine learning, pests, tomato.*

CAPÍTULO I

PLANTEAMIENTO DEL PROBLEMA

1.1 Descripción del problema

El tomate es uno de los frutos que provee muchos nutrientes y beneficios al ser humano, siendo uno de los principales vegetales presentes en la dieta de los abanquinos, de esta manera la producción de tomate es realizada a gran escala en todo el Perú, en 2021 la producción de tomate subió un 188,5% en Apurímac según INEI (INEI 2021). No obstante, las plagas del tomate son factores que pueden destruir cultivos completos en Abancay si no se toman acciones tempranas.

Cabe añadir que, las enfermedades y plagas del tomate habitualmente se presentan primero en las hojas y tallos de la planta que posteriormente desencadenan en frutos no consumibles; de tal modo que, la detección temprana de estas plagas ayuda a reducir las pérdidas económicas y de producción del tomate.

Dentro de las plagas más conocidas que afecta al cultivo de tomate están: Araña roja, mosca blanca, pulgón, trips, minadores de hoja, orugas, entre otra (ACOSTA 2022), ver Figura 1.

Fuente: REDAGRICOLA, 2018

Figura 1 — Plagas que afectan al tomate

Por otro lado, las enfermedades más comunes que atacan a las plantaciones de tomate son: oídio, podredumbre, antracnosis, mildiu, alternariosis, entre otros (ACOSTA 2022), ver Figura 2

FUENTE: REDAGRICOLA, 2018

Figura 2 — Enfermedades en el tomate

El productor necesita ser notificado lo más antes posible de la existencia de enfermedades o plagas que atacan al cultivo de tomate; debido a la lejanía de su domicilio, a las visitas esporádicas que realiza a su chacra con el fin de riego de su cultivo y a su desconocimiento de un método accesible y precoz para detectar plagas en el tomate.

Por otra parte, se resalta el uso de la inteligencia artificial que en base a un conjunto de imágenes pueda entrenar y posteriormente permita generar un modelo para detectar plagas, así como por ejemplo, la investigación “Redes neuronales para la identificación de enfermedades en el cultivo de tomate en Honduras” realizada por ERAZO VALDIVIEZO Jorge Deninson y UEMA HIGA Ken Sebastian (2020), quienes obtuvieron resultados satisfactorios al identificar dos de las principales enfermedades que afectan a los tomates en Honduras.

Por lo tanto, la detección de plagas en los cultivos de tomate es relevante para el agricultor, asimismo, es necesario desarrollar herramientas tecnológicas que utilicen técnicas de Machine Learning para detectar tempranamente plagas en los tomates, que pueden ocasionar pérdidas exorbitantes en los cultivos; por consecuencia, se plantea el siguiente problema general:

1.2 Enunciado del problema

1.2.1 Problema general

¿Cuál de las técnicas de Machine Learning es más eficiente para detectar las plagas en las hojas de los tomates en los cultivos de Abancay Apurímac, 2022?

1.2.2 Problemas específicos

- ¿Cuál es la eficiencia de clasificación del modelo generado con Yolo v4 en la detección de las plagas en las hojas de los tomates en los cultivos de Abancay Apurímac, 2022?
- ¿Cuál es la eficiencia de clasificación del modelo generado con Faster R-CNN en la detección de las plagas en las hojas de los tomates en los cultivos de Abancay Apurímac, 2022?

1.3 Justificación de la investigación

El tomate es considerado como una de las hortalizas más consumidas y de un gran valor económico en todo el mundo, por lo tanto, su cultivo se considera muy importante y que demanda mucho cuidado, porque a medida que los cultivos van creciendo, estas plantaciones de tomates son atacados por diferentes factores climáticos o enfermedades y plagas. Todo este problema surge por la falta de capacitación, ausencia de la tecnología y de inversión de los mismos productores. Según el artículo publicado en la página de Redagrícola (REDAGRICOLA, 2018). Al respecto, en los últimos años, la presencia de las enfermedades y plagas han incrementado, y la resistencia de ellos provoca que se usen los agroquímicos, cada vez más dosis y con mayor frecuencia.

Por lo tanto, la presente investigación fue realizada para automatizar el reconocimiento de las plagas que afectan a las hojas de tomate de manera temprana, mediante la aplicación del Machine Learning para posibilitar que el programa pueda identificar plagas por medio de imágenes; de tal manera sean combatidas y lograr reducir pérdidas en los cultivos de tomates.

CAPÍTULO II
OBJETIVOS E HIPÓTESIS

2.1 Objetivos de la investigación

2.1.1 Objetivo general

Comprobar la técnica de Machine Learning más eficiente para detectar las plagas en las hojas de los tomates en los cultivos de Abancay Apurímac, 2022.

2.1.2 Objetivos específicos

- Comprobar la eficiencia de clasificación del modelo generado con Yolo v4 en la detección de plagas en las hojas de los tomates.
- Comprobar la eficiencia de clasificación del modelo generado con Faster R-CNN en la detección de plagas en las hojas de los tomates.

2.2 Hipótesis de la investigación

No aplica.

2.3 Operacionalización de variables

La tabla 1 muestra la operacionalización de las variables

Tabla 1 — Operacionalización de variables

VARIABLE	DIMENSIÓN	INDICADOR	ÍDICE/ESCALA
Variable 1: Machine Learning	Yolo v4	Precisión (Precision)	Escala de 0 a 1
		Recall (Sensibilidad)	Escala de 0 a 1
		F1-Value (F1-Score)	Escala de 0 a 1
		Media armónica de F1-Value	Escala de 0 a 1
	Faster R-CNN	Precisión (Precision)	Escala de 0 a 1
		Recall (Sensibilidad)	Escala de 0 a 1
		F1-Value (F1-Score)	Escala de 0 a 1
		Media armónica de F1-Value	Escala de 0 a 1
Variable 2: Plagas en las hojas del tomate	La mosca blanca (Bemisia Tabaci)	Presencia real total	Número Absoluto
		Detección total	Número Absoluto
	Minador de las hojas (Liriomyza Trifolii)	Presencia real total	Número Absoluto
		Detección total	Número Absoluto

CAPÍTULO III
MARCO TEÓRICO REFERENCIAL

3.1 Antecedentes

3.1.1 Antecedentes internacionales

a) En Honduras, ERAZO VALDIVIEZO Jorge Deninson y UEMA HIGA Ken Sebastian (2020) , presentaron una tesis "Redes neuronales para la identificación de enfermedades en el cultivo de tomate (Solanum lycopersicum L.) en Honduras", los autores señalan que el tomate es una hortaliza muy importante convirtiéndose un alimento indispensable, propone la implementación de mecanismos de inteligencia artificial, la creación de un sistema de redes neuronales que permitan diagnosticar enfermedades en el tomate de forma confiable y precisa. La programación se realizó utilizando lenguaje de programación Python, que es un lenguaje fácil de entender, aunque se tuvo algunas dificultades o limitaciones en el número de enfermedades a detectar, por lo que se priorizó las enfermedades mejor definidas y de fácil interpretación para la red neuronal. La conclusión es que se codifico e implementó el programa y funciona adecuadamente, y logrando diagnosticar correctamente las enfermedades de Septoria sp y tizón temprano.

b) En Colombia, OLIVEROS SABOGAL David Santiago (2019), presentó una tesis "Algoritmo para la medición del grado de severidad de tizón temprano causado por *alternaria solani*, en hojas de tomate, a partir del análisis digital de imágenes", en el que especifica al *Alternaria solani* como un hongo fitopatógeno que afecta principalmente al tomate, propone como desarrollar una herramienta informática a partir del análisis digital de imágenes adquiridas. El objetivo del trabajo fue desarrollar una herramienta para estimar el porcentaje y grado de severidad de tizón temprano en hojas de tomate, basada en la herramienta Leaf Doctor. Los resultados muestran que de acuerdo a las imágenes analizadas se pudo entrenar el algoritmo y se logró proporcionar

como herramienta de apoyo para la detección de enfermades en tomate y que sirven al agricultor para tomar decisiones.

c) En el Estado de México, GARCÍA AMARO Ernesto (2022) presenta una tesis titulado "Uso de Técnicas de Visión Artificial para la Identificación de Daños Foliares Causados por Plagas y Enfermedades en Plantas de Jitomate". En este documento, destaca una de las principales preocupaciones de los agricultores: el descenso económico ocasionado por la presencia no deseada de enfermedades y plagas en los cultivos de jitomate. Con el objetivo de abordar esta problemática, el autor ha desarrollado una metodología con una estructura modular, compuesta por las siguientes etapas: preprocesamiento, segmentación, extracción de características, equilibrio de clases y clasificación. Para llevar a cabo su investigación, el autor utilizó el conjunto de datos Plantvillage, que incluye diez clases distintas, representadas por ocho enfermedades, una plaga y una clase completamente sana. A lo largo de las fases experimentales desarrolladas en este estudio, el sistema implementado ha sido sometido a pruebas y análisis exhaustivos, el cual alcanza un rendimiento del 94,65% de exactitud.

d) En el artículo titulado "Design of Efficient Methods for the Detection of Tomato Leaf Disease Utilizing Proposed Ensemble CNN Model," presentado por HASAN Ulutaş y VEYSEL Aslantaş (2023), ambos autores resaltan el impacto negativo de las enfermedades en las plantas. En su estudio, clasificaron hasta nueve enfermedades en las hojas del tomate, así como hojas sanas, utilizando el aprendizaje profundo con nuevas arquitecturas de conjunto. Además de incluir dos nuevos modelos propuestos de redes neuronales convolucionales (CNN), los autores emplearon otros cuatro modelos CNN bien conocidos (MobileNetV3Small, EfficientNetV2L, InceptionV3 y MobileNetV2).

   Los resultados experimentales obtenidos destacan el rendimiento superior de los modelos de conjunto propuestos, evidenciando un rápido tiempo de entrenamiento y prueba, junto con una precisión del 99,60% en la clasificación. Este estudio contribuye significativamente a la detección eficiente de enfermedades en hojas de tomate, subrayando la importancia de las

arquitecturas de conjunto propuestas para mejorar la precisión y la rapidez en el proceso de clasificación.

e) En artículo de investigación de OPPENHEIM Dor, ROYALTY Robert N., NEMEH Michelle K. y SMART Christine D. (2019), titulado “Using deep learning for image-based potato tuber disease detection”, aparte de señalar que indica una gran variedad de enfermedades, indica que también el clima y la tierra afectan la calidad y el rendimiento de los tubérculos. Los autores señala que es necesario examinar los tubérculos de manera minuciosa, sin embargo, esas estructuras no siempre están presentes, indicando que la detección y clasificación manual de tubérculos enfermos es difícil, costosa y lleva mucho tiempo, mientras que la inspección desde una computadora será más eficiente y rentable. Por lo tanto, propone aprovechar los avances de visión por la computadora y el reconocimiento de objetos, una red neuronal convolucional profunda, utilizando el modelo CNN, los resultados que se obtuvieron al estar completamente entrenados aborda entre el 83% y 96% de los datos, los resultados sugiere la combinación de un algoritmo de ventana deslizante o una red de detección de objetos, como R-CNN más rápida, de tal manera puedan facilitar la detección de las enfermedades de los tubérculos de manera más completa.

f) En Ecuador, GUERRERO ANDRADE Carlos Jonathan y MARTÍNEZ MOSQUERA Silvia Diana (2022) llevaron a cabo una investigación titulada "Reconocimiento de lesiones necróticas para la detección de la plaga thrips en el guisante mediante el uso del modelo de aprendizaje profundo YOLOv4-tiny". En este estudio, los investigadores abordaron los desafíos climáticos que enfrentan los productores en Ecuador, como fuertes heladas y sequías prolongadas. La detección temprana de plagas se ha vuelto esencial para evitar pérdidas en los cultivos.

   Los autores propusieron un sistema de reconocimiento rápido y eficaz de lesiones necróticas causadas por la plaga thrips en el guisante, utilizando el modelo de aprendizaje profundo YOLOv4-tiny. Los resultados obtenidos mostraron que la Intersección sobre la Unión (IoU) alcanzó el 59,23%, con una Precisión Media (mAP) de 81,60% sobre un conjunto de datos de alta densidad.

Esto determina que el sistema logra una efectividad del 86% en la detección del objeto de estudio.

Este estudio demuestra la viabilidad y eficacia del modelo YOLOv4-tiny en la detección rápida y precisa de lesiones necróticas causadas por la plaga thrips en los cultivos de guisantes, ofreciendo así una herramienta valiosa para los productores en la gestión temprana de plagas y la protección de sus cultivos.

g) En Bogotá D.C., SIERRA GUZMÁN, Viviana Yineth (2021) presentó su investigación titulada "Algoritmo de Detección de Mosca Blanca mediante Inteligencia Artificial en Hojas de Plátano". En este estudio, el autor, con el objetivo de realizar un monitoreo e identificación oportuna que garantice una producción acorde con las demandas del mercado, propone un algoritmo de detección de la mosca blanca en plantas de plátano utilizando inteligencia artificial. La solución propuesta por el autor se basa en el uso del software Matlab, donde implementó técnicas avanzadas de procesamiento de imágenes. El algoritmo desarrollado fue entrenado específicamente para la detección y clasificación de hojas sanas y hojas afectadas por la presencia de mosca blanca. Los resultados obtenidos en el análisis de la matriz de confusión revelaron una notable efectividad del 94,4%, junto con una precisión del modelo del 88,8%. Estos resultados destacan la eficacia del algoritmo propuesto por Viviana S. en la identificación precisa de la mosca blanca en hojas de plátano.

3.1.2 Antecedentes nacionales

a) En el Perú, se realizó la tesis “Clasificación de hojas de tomate con plagas o enfermedades usando una máquina de soporte vectorial (SVM)”, presentado por PRIETO MORANTE Juan Sebastián y TRELLES PRIETO Rita Luciana (2021), los autores indican la importancia del cultivo del tomate y que son afectados por distintas plagas o enfermedades, por lo que propone un algoritmo clasificador basado en Máquinas de Soporte Vectorial (SVM) que clasifique las imágenes de hojas de tomate usando algoritmos binarios secuenciales. En la metodología, se utilizó el desarrollo de múltiples capas, con un conjunto de imágenes o dataset de hojas de tomate, luego fueron entrenadas por la computadora con el 80% de los datos, mientras que el 20% fue usado para las pruebas. Los resultados muestran que el programa funciona adecuadamente y

se obtiene la clasificación de hojas en sanas o enfermas con una precisión de 99,3%.; asimismo se clasificó a las hojas enfermas en dos tipos de enfermedades: Tizón tardío y Marchitamiento bacteriano con una precisión 97,2% en promedio.

b) CÓRDOVA PÉREZ, Claudia Sofía (2021) presentó una tesis de investigación titulada "Aplicación de aprendizaje profundo para la detección y clasificación automática de insectos agrícolas en trampas pegantes". En su estudio, destaca los desafíos que enfrenta la horticultura en diversas regiones de Perú debido a la alta incidencia de plagas de insectos. Para abordar este problema, se utilizan trampas pegantes para atraer y capturar insectos. Sin embargo, el proceso manual de identificación de estos insectos puede verse afectado por factores como la habilidad individual y la fatiga, afectando la precisión de la información recopilada. La autora propone el uso de modelos de detección preentrenados, específicamente Faster R-CNN y YOLOv4, aplicándoles aprendizaje por transferencia para adaptarlos a la detección de tres tipos de plagas de insectos: mosca blanca, mosca minadora y pulgón verde. Los resultados obtenidos muestran que el modelo Faster R-CNN alcanzó un 94,06%, mientras que el modelo YOLOv4 obtuvo un 95,82%. La conclusión es que el rendimiento de ambos detectores es aceptable para la detección automática de plagas en trampas pegantes.

c) LÓPEZ PORTOCARRERO, César Augusto (2019), realizó su investigación de tesis “Aplicación de imágenes hiperespectrales para la detección temprana de roya amarilla (hemileya vastatrix) en café (coffea arábica), en el distrito de Limbamba, Provincia Rodríguez de Mendoza Región Amazonas”, su investigación se centra en las enfermedades del café, donde propone desarrollar una aplicación basado en imágenes hiperespectrales. Para lograr el objetivo, se establecieron 6 índices para el entrenamiento, la metodología usada para la clasificación fue el árbol de decisión. Se utilizó el software matemático Matlab versión 2010. Los resultados muestran que las imágenes hiperespectrales fueron comparadas con las clasificadas visualmente, y se llegó a la conclusión que existe diferencia significativa entre ambos métodos, y se detectó que una de las enfermedades tempranas que ataca es la Hemileya vastatrix.

d) Según la tesis de posgrado que lleva por título “Determinación en tiempo real de presencia de cadmio en cultivo de cacao aplicando Machine Learning”, presentado por NEYRA HAU YON Jorge Luis (2021), para poder reducir el riesgo de contaminación con Cadmio la producción del Cacao. Los resultados muestran que se logró hacer funcionar el algoritmo de predicción basados Machine Learning y se puede hacer un monitoreo en tiempo real con la tecnología de la Visión Hiperespectral para su detección temprana, y para luego tomar medidas correctivas y así lograr mitigar la contaminación por metales pesados.

3.1.3 Antecedentes locales

a) MORENO MAYHUIRE Joel Saul (2020), en su tesis "Eficiencia de un Sistema de Control de Calidad mediante Procesamiento Digital de Imágenes en la Clasificación de la Tunta en la Planta de Producción de Kishuara – Andahuaylas" destaca la importancia de la tunta en la gastronomía peruana, especialmente en la planta de producción de Kishuara – Andahuaylas. La investigación utilizó algoritmos de Machine Learning supervisado para clasificar 800 unidades de tuntas, extrayendo características mediante procesamiento digital de imágenes. El modelo basado en Support Vector Machines mostró el mejor rendimiento, logrando una precisión del 93,13%, una sensibilidad del 93,48% y un F1-Value del 93,18%. Este modelo eficiente se implementó en un sistema clasificador de tuntas, alcanzando una precisión del 97,5%, una sensibilidad del 97,5% y un F1-Value del 97,51%.

3.2 Marco teórico

3.2.1 Machine Learning

El Machine Learning, o aprendizaje automático, es una rama de la inteligencia artificial que se enfoca en el diseño y estudio de algoritmos que permiten a las máquinas aprender de datos pasados y mejorar sus decisiones o predicciones para el futuro. A través de la exposición repetida a ejemplos, estos algoritmos buscan identificar patrones y relaciones en los datos, lo que permite construir modelos capaces de realizar tareas de clasificación, regresión y reconocimiento de patrones.

Su objetivo principal es generalizar el conocimiento adquirido a partir de datos de entrenamiento, permitiendo que el sistema aprenda reglas o estructuras que puedan aplicarse a nuevas situaciones. Este proceso de aprendizaje incluye métodos supervisados, no supervisados y de refuerzo, cada uno de los cuales se adapta a diferentes tipos de problemas y estructuras de datos. En aplicaciones avanzadas, como el aprendizaje profundo, se emplean redes neuronales profundas para abordar problemas complejos, tales como el reconocimiento de imágenes y el procesamiento de lenguaje natural (LÓPEZ, 2018).

3.2.1.1 Tipos de Machine Learning

Esta basando en una amplia gama de aplicaciones, es necesario analizar los diferentes casos con lo que Machine Learning se puede encontrar.

a) Aprendizaje supervisado

En este tipo de aprendizaje el algoritmo es entrenado a partir de unos datos que ya vienen etiquetados con la respuesta correcta. Por lo cual, el algoritmo ya puede predecir un resultado basado en el aprendizaje o entrenamiento que ya obtuvo. Los algoritmos supervisados más importantes son:

- K-nearest neighbors
- Red Neuronal
- Máquina de soporte de vectores
- Regresión logística
- Árboles de decisiones y bosques aleatorios (RUSSELL 2018)

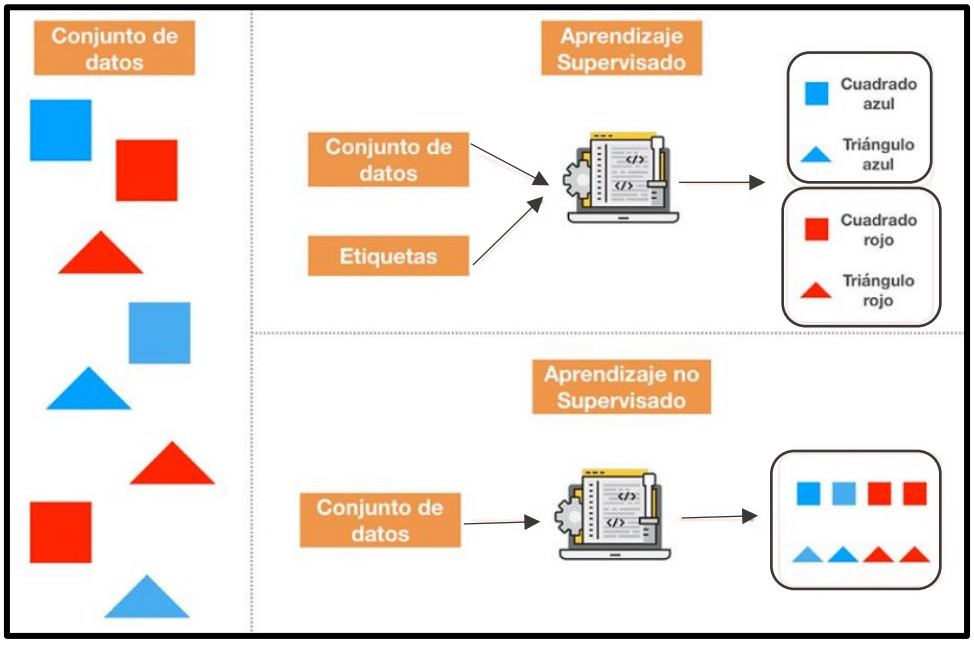
b) Aprendizaje no supervisado

El algoritmo es entrenado, en este caso los datos no se encuentran etiquetados, es decir, no se indica al algoritmo lo que representan los datos. El enfoque es que el algoritmo pueda identificar los patrones para que pueda entender los datos. Un ejemplo muy claro, es el método de los bebés cuando aprender a hablar, lo aprenderán a partir de las personas que lo rodean. Los algoritmos no supervisados son:

- Agrupamiento (Clustering): Medios k, análisis de agrupamientojerárquico).
- Machine Learning de asociación de regla: Eclat y A priori

- Visualización y reducción de dimensionalidad: Núcleo PCA, distribuido de t PCA (RUSSELL, 2018).

En la Figura 3 podemos ver la comparación de ambos aprendizajes.



FUENTE: GONZÁLEZ, 2018

Figura 3 — Comparación de aprendizaje supervisado y aprendizaje xno supervisado

c) Aprendizaje por refuerzo

Estos algoritmos aprenden observando y analizando el mundo que los rodea, su sistema de Inteligencia Artificial observará el ambiente ejecutando diferentes acciones, de los cuales recibirá respuestas, todo esto será considerado como un feedback o retroalimentación, por lo tanto, el sistema aprende de un ensayo-error (RUSSELL, 2018).

3.2.1.2 Aprendizaje Profundo (Deep Learning)

El aprendizaje profundo (Deep Learning) es una subárea del aprendizaje automático que utiliza arquitecturas de redes neuronales profundas para aprender y representar características jerárquicas y complejas de los datos. Este enfoque ha revolucionado múltiples campos, como el procesamiento de imágenes, el reconocimiento de voz y el procesamiento del lenguaje natural, debido a su capacidad de aprendizaje automático escalable y su capacidad

para modelar relaciones complejas en grandes volúmenes de datos (GOODFELLOW, BENGIO Y COURVILLE, 2016).

El término "profundo" se refiere a la presencia de múltiples capas de procesamiento que permiten una transformación progresiva y no lineal de los datos. Las capas iniciales suelen aprender características básicas, como bordes o patrones simples, mientras que las capas posteriores capturan representaciones de mayor nivel de abstracción. La capacidad de capturar esta jerarquía de características hace que el aprendizaje profundo sea particularmente efectivo en tareas de detección y clasificación.

a) Arquitectura de Redes Neuronales Profundas

Las redes neuronales profundas están formadas por neuronas artificiales organizadas en capas. Se componen típicamente de tres tipos principales de capas:

- **Capa de entrada (Input Layer)**: Esta capa recibe los datos brutos que alimentarán la red. En aplicaciones de visión por computadora, cada píxel de una imagen puede representarse como una entrada en esta capa (LECUN, BENGIO Y HINTON, 2015).
- **Capas ocultas (Hidden Layers)**: Pueden ser varias y realizan el procesamiento no lineal de las entradas mediante funciones de activación. Estas capas transforman y representan la información aprendida, lo que les permite identificar características complejas.
- **Capa de salida (Output Layer):** Esta capa proporciona el resultado final de la predicción o clasificación.

El proceso de aprendizaje en las redes profundas se basa en un algoritmo llamado retropropagación, que ajusta iterativamente los pesos de las conexiones entre las neuronas para minimizar la diferencia entre la salida generada y la esperada (error). Este proceso se realiza mediante un mecanismo de optimización, como el gradiente descendente (RUDER, 2016).

b) Redes Neuronales Convolucionales (CNN)

Las redes neuronales convolucionales (Convolutional Neural Networks - CNN) son una categoría específica de redes neuronales utilizadas

principalmente para el procesamiento de datos con una estructura de grilla, como las imágenes (KRIZHEVSKY, SUTSKEVER Y HINTON, 2012). Las CNN se caracterizan por tener capas convolucionales que realizan operaciones de convolución sobre las entradas, detectando características locales, como bordes, texturas y formas. Estas características luego se combinan en las capas superiores para identificar patrones más complejos.

Los componentes clave de las CNN incluyen:

- **Capas convolucionales**: Aplican filtros para extraer características locales.
- **Capas de agrupamiento (Pooling):** Reducen la dimensionalidad de las características extraídas, preservando las más importantes.
- **Capas completamente conectadas (Fully Connected Layers):** Producen la salida final, combinando todas las características detectadas.

c) Aplicaciones en la Detección de Objetos

Modelos como YOLO (You Only Look Once) y Faster R-CNN son redes neuronales convolucionales especializadas para la detección de objetos. Estas arquitecturas permiten no solo identificar la presencia de un objeto en una imagen, sino también localizarlo mediante cuadros delimitadores.

- **YOLO (You Only Look Once):** YOLO es un modelo de detección de objetos que divide la imagen en una cuadrícula y predice múltiples cuadros delimitadores y probabilidades para cada celda en una sola pasada a través de la red. Esto lo hace extremadamente rápido y eficiente en tareas de detección en tiempo real (REDMON y otros, 2016).
- **Faster R-CNN**: Este modelo combina una red de propuestas de regiones (Region Proposal Network, RPN) con una red convolucional para generar regiones de interés en una imagen y

clasificarlas, proporcionando una alta precisión en la detección de objetos, aunque a un costo computacional mayor (REN y otros, 2015).

El aprendizaje profundo en aplicaciones de visión por computadora ha demostrado ser altamente efectivo, y las arquitecturas como YOLO y Faster R-CNN continúan liderando en tareas de detección de objetos y clasificación en tiempo real, con un rendimiento de alta precisión.

3.2.1.3 Modelos de Machine Learning

a) Modelos lineales

Estos modelos procuran encontrar una línea que encaje a la nube de puntos que se disponen. Destacan modelos conocidos y usados como es la regresión lineal. Estos modelos procuran encajar demasiado, puede ocasionar problemas para los nuevos datos que puedan ingresar. No pueden ofrecer buenos resultados para comportamientos más complejos (SANDOVAL 2018).

b) Modelos de árbol

Son modelos más precisos, estables y sencillos, ya que pueden construir reglas de decisión que se pueden representar como un árbol. Llegan a ser más precisos y son muy elaborados, por lo que también pierden rendimiento (SANDOVAL 2018).

c) Redes neuronales

Las redes neuronales artificiales (RNA) son modelos computacionales inspirados en la estructura y funcionamiento del cerebro humano. Estos modelos están compuestos por neuronas artificiales organizadas en capas, lo que permite simular procesos de aprendizaje y reconocimiento de patrones complejos. En una red neuronal típica, cada neurona recibe entradas, realiza una operación matemática sobre ellas y transmite una salida a las neuronas de la siguiente capa. Este proceso de transmisión de información permite que la red ajuste sus parámetros y aprenda a realizar tareas específicas (SANDOVAL, 2018).

La Figura 4 ilustra una red neuronal artificial con una capa de entrada que recibe múltiples características (como luz, temperatura, y

nutrientes), una o más capas ocultas que procesan esta información y una capa de salida que produce resultados específicos, como la predicción de transpiración, asimilación y distribución en plantas. Este tipo de arquitectura es común en redes neuronales y permite analizar y predecir resultados complejos en función de varias variables de entrada.

Las RNA se estructuran en tres capas principales:

- **Capa de entrada**: recibe los datos iniciales y los distribuye a las neuronas de las capas intermedias.
- **Capas ocultas**: procesan la información y detectan patrones, ajustando los pesos de las conexiones entre neuronas mediante algoritmos de optimización.
- **Capa de salida**: genera la predicción o clasificación final en función del análisis realizado en las capas anteriores.

El aprendizaje en redes neuronales ocurre mediante el ajuste iterativo de los pesos que conectan las neuronas, usando técnicas como la retropropagación del error. Este algoritmo compara la salida generada con el valor esperado, ajustando los pesos para minimizar el error y mejorar la precisión del modelo. Gracias a su capacidad para aprender de grandes volúmenes de datos, las redes neuronales son ampliamente utilizadas en inteligencia artificial y visión computacional.

i. Tipos de Redes Neuronales

Existen varios tipos de redes neuronales según su arquitectura y propósito, entre los más comunes se encuentran:

- **Redes Neuronales Feedforward**: La información fluye en una sola dirección, de la capa de entrada a la capa de salida, sin conexiones de retroalimentación. Son utilizadas principalmente en tareas de clasificación y regresión.

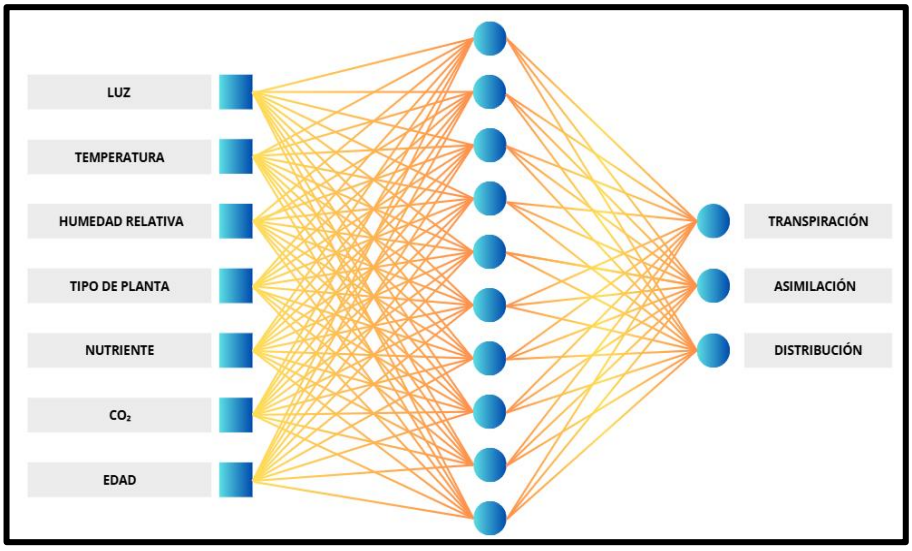
- **Redes Neuronales Convolucionales (CNN)**: Estas redes son ideales para el procesamiento de imágenes y videos debido a sus

capas de convolución, que permiten detectar características espaciales como bordes y texturas. Dentro de las CNN, modelos como YOLO (You Only Look Once) y Faster R-CNN son ampliamente utilizados en detección de objetos.

- **Redes Neuronales Recurrentes (RNN):** Poseen conexiones de retroalimentación y son ideales para tareas secuenciales, como el procesamiento de lenguaje natural y la predicción de series temporales.

ii. Aplicaciones y Avances

Las redes neuronales se han convertido en una herramienta fundamental en la inteligencia artificial, con aplicaciones en reconocimiento de voz, detección de objetos en imágenes y traducción automática. En el contexto de la agricultura, estas redes permiten implementar modelos de visión computacional que detectan plagas y enfermedades en cultivos con alta precisión. Los modelos YOLO y Faster R-CNN, por ejemplo, son utilizados en la detección de plagas en hojas de tomate, lo que permite intervenciones rápidas y precisas, contribuyendo a mejorar la productividad y calidad de los cultivos (SANDOVAL, 2018).



FUENTE: SANDOVAL, 2018

Figura 4 — Redes neuronales

3.2.1.4 Medidas de desempeño del modelo Machine Learning

La Tabla 2 presenta la matriz de confusión, que permite evaluar el desempeño del modelo en términos de precision, recall y F1-value.(ALPAYDIN, 2014)

Tabla 2 — Matriz de confusión

		Valores reales	
Valores predichos		**Positivo**	**Negativo**
	Positivo	True Positive (Verdadero Positivo)	False Positive (Falso Positivo)
	Negativo	False Negative (Falso Negativo)	True Negative (Verdadero Negativo)
FUENTE: ALPAYDIN, 2014			

- **FP: Falso Positivo (False Positive):** El modelo identifica incorrectamente un caso negativo como positivo. Ejemplo: El modelo predice que una persona está enferma cuando en realidad está sana.
- **FN: Falso Negativo (False Negative):** El modelo identifica incorrectamente un caso positivo como negativo. Ejemplo: El modelo predice que una persona está sana cuando en realidad está enferma.
- **TP: Verdadero Positivo (True Positive):** El modelo identifica correctamente un caso positivo. Ejemplo: El modelo predice que una persona está enferma y efectivamente lo está.
- **TN: Verdadero Negativo (True Negative):** El modelo identifica correctamente un caso negativo. Ejemplo: El modelo predice que una persona está sana y efectivamente lo está. (ALPAYDIN, 2014)

Precisión (Precisión):

La precisión considera a la relación entre observaciones señaladas como correctas y el total de observaciones positivas (GIL, CRUZ Y PERDOMO, 2022).

$$\boldsymbol{Precision} = \frac{True\ positives}{True\ positives\ +\ False\ positives}$$

Recall (Sensibilidad)

Recall, considera a la proporción de observaciones señaladas como correctas a todas las observaciones en la clase real (GIL, CRUZ Y PERDOMO, 2022).

$$\boldsymbol{Recall} = \frac{True\ positives}{True\ positives\ +\ False\ negatives}$$

F1-Value (F1-Score)

Es el promedio ponderado de precisión y exhaustividad. Por lo tanto, se considera los falsos positivos como los falsos negativos. F1 suele ser más útil que la precisión, especialmente si tiene una distribución de clases desigual. La precisión funciona mejor si los falsos positivos y los falsos negativos tienen un costo similar. Si el costo de los falsos positivos y los falsos negativos es muy diferente, es mejor mirar tanto *Precisión* como *Recall* (GIL, CRUZ Y PERDOMO, 2022).

$$\boldsymbol{F-Value} = 2 * \frac{Precision * Recall}{Precision\ +\ Recall}$$

Uso de la Media Armónica para la Comparación de Modelos de Detección

Para determinar el modelo más eficiente en la detección de plagas en hojas de tomate (Mosca Blanca y Minador de Hojas), se empleará la media armónica de las métricas de F1-value entre ambas clases. Este enfoque permite obtener una evaluación justa y representativa del desempeño general de cada modelo al considerar sus capacidades para manejar distintas clases de manera equilibrada. Según SASAKI (2007), la media armónica es especialmente útil en escenarios de detección multiclase o desbalanceada, donde la evaluación debe ponderar correctamente la precisión y el recall.

El cálculo de esta métrica permite establecer un criterio de comparación sólido, asegurando que se evalúe el desempeño de cada modelo de manera integral y

equilibrada. Esta metodología ha sido recomendada y utilizada en estudios previos para evaluar modelos en contextos de clasificación compleja y detección de múltiples clases (SOKOLOVA Y LAPALME, 2009)

3.2.1.5 Ciclo de Vida del Machine Learning (ML Lifecycle)

Es una metodología estructurada que organiza las fases principales necesarias para desarrollar, entrenar y evaluar modelos de Machine Learning. Esta metodología cubre desde la recolección de datos hasta el análisis de los resultados, asegurando que cada paso se realice de forma coherente para obtener resultados precisos y útiles. (AMERSHI y otros, 2019)

A continuación, se describen los cinco pasos fundamentales de este ciclo de vida, junto con las referencias específicas para cada punto:

1. **Recolección de Datos:** Es la fase inicial de cualquier proyecto de Machine Learning. En esta etapa, se obtienen los datos necesarios para entrenar el modelo. La calidad y cantidad de los datos recopilados juegan un papel crucial en la precisión de los resultados que ofrecerá el modelo. En proyectos de visión por computadora, como la detección de plagas en hojas de tomate, es importante obtener imágenes que representen adecuadamente las condiciones reales de los objetos de estudio. (GÉRON, 2022)
2. **Preparación y Etiquetado de Datos:** Después de recolectar los datos, es necesario prepararlos y, en muchos casos, etiquetarlos. Para proyectos de machine learning supervisados, como la clasificación de imágenes, el etiquetado manual o automático es fundamental para que el modelo aprenda a distinguir correctamente los elementos que necesita identificar. El etiquetado implica asignar una categoría o etiqueta a cada dato, mientras que la preparación incluye la normalización, eliminación de ruido, y la partición del dataset en subconjuntos de entrenamiento, validación y testing. (GÉRON, 2022)
3. **Entrenamiento de Modelos:** Es una de las fases más críticas en el ciclo de vida del machine learning. Aquí es donde el modelo de aprendizaje automático se entrena usando el conjunto de datos etiquetados previamente. Durante este proceso, el modelo ajusta sus parámetros internos para minimizar el error entre sus predicciones y las etiquetas reales. En el caso de los modelos de detección de objetos, como YOLOv4 y Faster R-CNN, se

ajustan múltiples capas de redes neuronales convolucionales para mejorar la capacidad del modelo de identificar correctamente los objetos de interés en las imágenes. (REN y otros, 2017)

4. **Evaluación del Modelo:** Una vez entrenado el modelo, se procede a evaluarlo utilizando un conjunto de datos separado que no fue visto por el modelo durante el entrenamiento. El conjunto de datos de testing se utiliza para medir el rendimiento del modelo en términos de precisión, recall y F1-Score. Estas métricas proporcionan una visión detallada del equilibrio entre las predicciones correctas y los errores del modelo, y permiten evaluar su capacidad para generalizar y detectar correctamente las plagas en datos nuevos y no conocidos.
5. **Análisis de Resultados:** Consiste en interpretar las métricas obtenidas y comparar diferentes modelos para identificar cuál tiene un mejor rendimiento para el problema en cuestión. En este caso, el análisis se enfocó en comparar YOLOv4 y Faster R-CNN en términos de precisión y velocidad de detección, para seleccionar el modelo que mejor se ajuste a las necesidades del proyecto. (CHOLLET, 2017)

3.2.1.6 Áreas de impacto del Machine Learning

- **Agricultura:** El Machine Learning está transformando la agricultura al mejorar la precisión y eficiencia de las prácticas agrícolas. En la detección de plagas y enfermedades en cultivos, modelos de visión artificial permiten identificar plagas específicas en plantas, analizando patrones y características visuales para alertar a los agricultores sobre la presencia de estas amenazas. Esto posibilita intervenciones tempranas que reducen pérdidas en la producción y mejoran la calidad de los cultivos. La agricultura de precisión, impulsada por el Machine Learning, también optimiza la aplicación de insumos, como fertilizantes y pesticidas, ayudando a minimizar el impacto ambiental (PEREYRA, 2020).

  Además, el Machine Learning contribuye al desarrollo de sistemas de monitoreo de cultivos a gran escala, que integran datos climáticos, condiciones del suelo y patrones de crecimiento. Esta información se utiliza para predecir necesidades de riego y fertilización, así como para anticipar riesgos climáticos, facilitando una gestión más sostenible de los recursos. El uso de estas tecnologías en la agricultura no solo aumenta la rentabilidad, sino

que también contribuye a satisfacer la creciente demanda de alimentos en un mundo cada vez más poblado (CORRALES, 2015).

- **Medicina:** En el ámbito de la medicina, la integración de la inteligencia artificial mediante Machine Learning busca mejorar la eficiencia de los equipos de diagnóstico, reduciendo errores humanos en el análisis de datos y disminuyendo los costos de investigación. Ya se observan avances, como el uso de programas como IBM Watson y chatbots, que interactúan con los pacientes para formular hipótesis sobre su estado de salud o realizar análisis de seguimiento.

  En los últimos años, el Machine learning ha tenido un impacto significativo en la salud y la medicina, como en la detección temprana de células cancerígenas, lo que permite tratamientos más específicos y prolonga la vida de los pacientes. Además, el Machine Learning ha sido crucial en la predicción de brotes epidémicos, mediante herramientas que monitorean brotes a nivel global utilizando datos de satélites, información histórica, máquinas de vectores de soporte y redes neuronales. Estos avances son especialmente vitales en países en desarrollo, donde la infraestructura médica es limitada. Programas como ProMED-mail, que monitorean enfermedades emergentes y brindan informes en tiempo real, han sido desarrollados para abordar esta necesidad. Finalmente, el crecimiento del mercado de aplicaciones y dispositivos médicos ha ampliado considerablemente las expectativas de vida. (HINESTROZA, 2018)
- **Educación:** La aplicación del Machine Learning en la educación podría ser transformadora, permitiendo sistemas que evalúen a cada estudiante y tracen planes de trabajo personalizados para abordar sus necesidades específicas. Esto liberaría a los maestros para que se concentren en suplir las verdaderas deficiencias de cada alumno. Además, la inteligencia artificial podría, eventualmente, asistir a los maestros, eliminando barreras y llevando educación de calidad a todos los rincones del mundo. (HINESTROZA, 2018)
- **Construcción:** En el sector de la construcción, la combinación de Machine Learning y robótica ha revolucionado la producción. Antes, la mayoría de los sistemas eran operados por humanos, lo que limitaba los horarios de trabajo y reducía la eficiencia, especialmente en tareas peligrosas. La automatización

ha permitido aumentar la rentabilidad y la eficiencia en las grandes fábricas. (HINESTROZA, 2018)

- **Finanzas:** En las finanzas, el Machine Learning hizo una de sus primeras apariciones, ya que muchas industrias necesitaban plataformas intuitivas para revisar y verificar trámites y transacciones. Programas como Mint o TurboTax recopilan información de clientes de diversas entidades financieras. Incluso IBM Watson es crucial en mercados como Wall Street, donde realiza la mayoría de las operaciones financieras. (HINESTROZA, 2018)
- **Robótica:** En la robótica, el impacto del Machine Learning ha sido notable, tanto en el presente como en sus proyecciones futuras. Ejemplos como el robot Asimo de Honda muestran cómo los robots pueden realizar tareas que antes se consideraban demasiado complejas para las máquinas. Además, países como China, Japón, Rusia, Corea del Sur y Estados Unidos han logrado grandes avances en esta industria**.** (HINESTROZA, 2018)
- **Turismo:** El aprendizaje automático en turismo optimiza servicios al predecir necesidades, personalizar estancias y mejorar campañas online para aerolíneas, hoteles y agencias. (CÁMARA DE COMERCIO DE MADRID, 2019)
- **E-commerce:** El Machine Learning mejora la atención al cliente con chatbots, optimiza la experiencia de usuario y ajusta los KPIs, o Indicadores Clave de Desempeño. (CÁMARA DE COMERCIO DE MADRID, 2019)
- **Industria:** El Machine Learning y Big Data optimizan sistemas robóticos, reducen costes y tiempos de producción, y mejoran la toma de decisiones. (CÁMARA DE COMERCIO DE MADRID, 2019)

3.2.1.7 Yolo

Yolo es un modelo de detección de objetos de una sola etapa, diseñado para realizar detecciones rápidas y precisas en tiempo real. A diferencia de los detectores de dos etapas, que requieren una fase inicial de propuesta de regiones candidatas antes de realizar la detección final, YOLO procesa la imagen completa de una sola vez, dividiéndola en cuadrículas y detectando objetos en cada cuadrícula de manera simultánea. Esta arquitectura permite una velocidad de procesamiento significativamente mayor, ideal para aplicaciones en las que el tiempo de respuesta es crítico, como los sistemas de visión en automóviles autónomos (GIL, 2020).

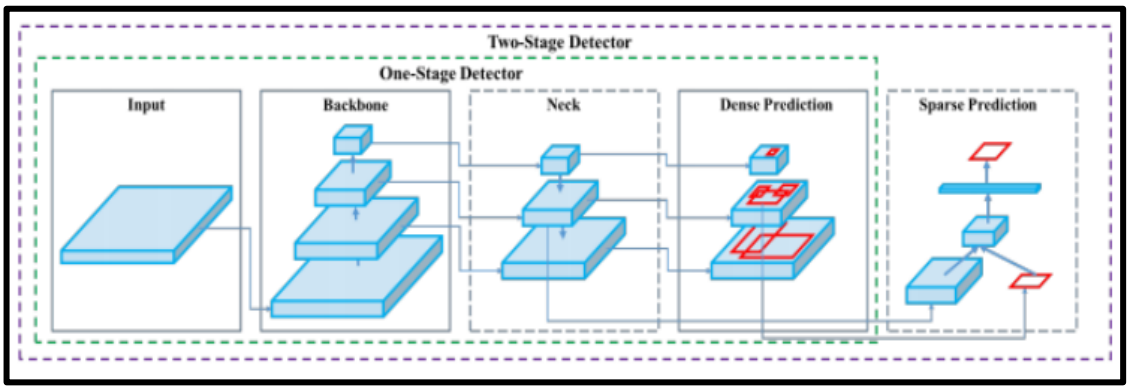
Las características que han hecho a YOLO tan popular incluyen su alta velocidad, la precisión en la detección, su capacidad de generalización a diferentes contextos y su disponibilidad como código abierto. Los principales componentes de YOLO son:

- **Backbone**: Es la columna vertebral de la red neuronal, encargada de la extracción de características. El backbone analiza la imagen y extrae información clave de bajo y alto nivel, necesaria para identificar los objetos presentes.

- **Cuello**: El cuello actúa como un conector entre el backbone y la cabeza de detección. Su función es refinar las características extraídas en diferentes fases del backbone, mejorando la capacidad del modelo para reconocer objetos en diversas escalas.

- **Cabeza**: Este componente realiza la detección final de los objetos. A partir de las características procesadas, la cabeza identifica las clases de objetos y determina su ubicación en la imagen (GIL, 2020).

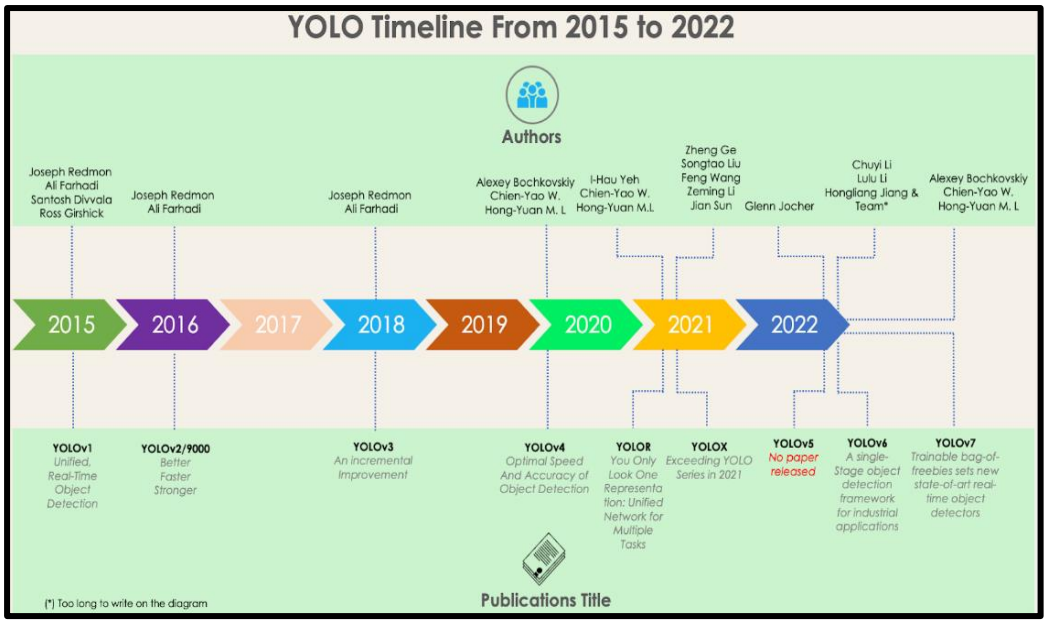
Esta estructura modular de YOLO permite su aplicación en una amplia gama de tareas de visión, combinando eficiencia y precisión, ver Figura 5.



FUENTE: BOCHKOVSKIY, WANG Y LIAO, 2020

Figura 5 — Detector de objetos Yolo detector de una etapa.

Desde su lanzamiento en 2015, Yolo ha evolucionado a través de diferentes versiones. La Figura 6 muestra en detalle cómo ha progresado su desarrollo a lo largo de los años.



FUENTE: DATACAMP, 2024

Figura 6 — Cronología de Yolo de 2015 a 2022

Yolo v4

YoloV4 es una mejora significativa en la familia YOLO, diseñada para optimizar tanto la velocidad como la precisión en la detección de objetos, lo cual lo convierte en una opción ideal para sistemas de producción en entornos de tiempo real. Esta versión está específicamente optimizada para aprovechar el cálculo paralelo en hardware moderno, logrando así una eficiencia sobresaliente en aplicaciones prácticas (DATACAMP, 2024).

En términos de arquitectura, YOLOv4 utiliza CSPDarknet53 como su columna vertebral (backbone), la cual incluye 29 capas de convolución y un total de 27,6 millones de parámetros. Esta estructura proporciona una base robusta y eficiente para la extracción de características. A diferencia de su predecesor YOLOv3, YOLOv4 incorpora varias innovaciones, como el bloque Spatial Pyramid Pooling (SPP), que mejora el campo receptivo de la red y permite capturar mejor las variaciones en escala de los objetos. Además, YOLOv4 emplea PANet en lugar de FPN para la

agregación de características, lo que mejora la precisión en la detección de objetos pequeños y medianos al optimizar el flujo de características a través de la red.

Otras mejoras incluyen la técnica de mosaico para el aumento de datos, que permite entrenar el modelo con diversas combinaciones de imágenes, mejorando su capacidad de generalización. Además, YOLOv4 selecciona sus hiperparámetros de manera óptima mediante el uso de algoritmos genéticos, asegurando un equilibrio adecuado entre precisión y velocidad (DATACAMP, 2024).

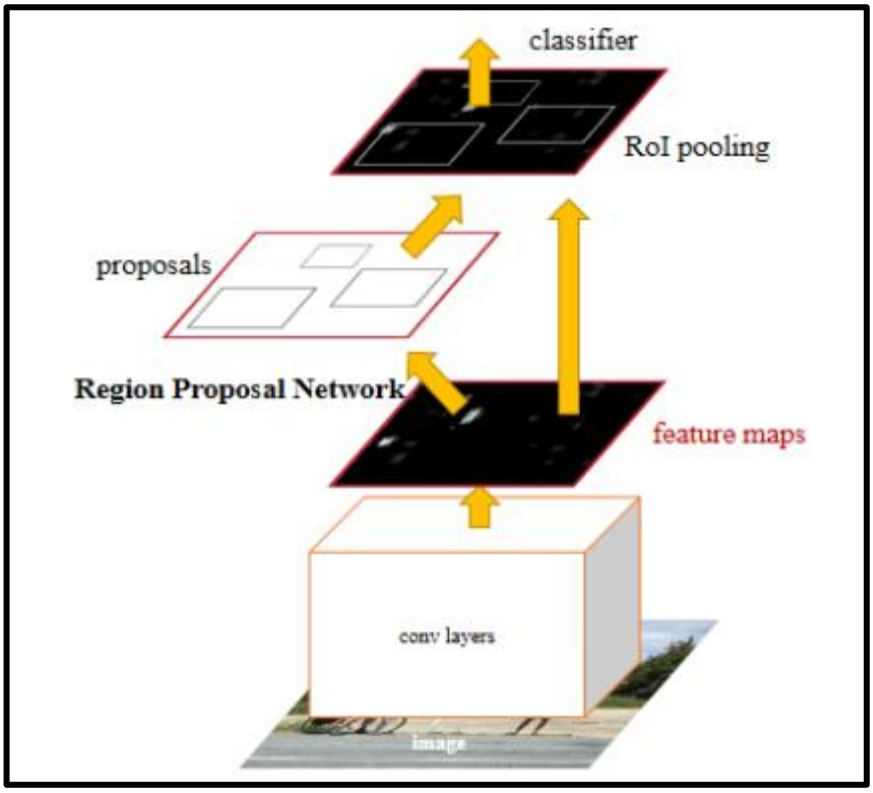
3.2.1.8 Faster R-CNN

Faster R-CNN es un marco de detección de objetos de dos etapas, diseñado para mejorar tanto la precisión como la eficiencia en comparación con sus predecesores. Este modelo se basa en una red convolucional (CNN) para la extracción de características y se estructura en dos subredes principales: una Red de Propuestas de Regiones (Region Proposal Network, RPN) y una red para la clasificación y ajuste de los cuadros delimitadores. La RPN genera propuestas de regiones que podrían contener objetos, y la segunda red clasifica estas regiones y ajusta los cuadros delimitadores para una detección más precisa (REN y otros, 2017).

Faster R-CNN optimiza la carga computacional en comparación con métodos anteriores como R-CNN y Fast R-CNN, ya que unifica la generación de propuestas de regiones y la detección de objetos en un solo flujo de trabajo, reduciendo significativamente el tiempo de procesamiento y permitiendo su uso en aplicaciones de casi tiempo real. Aunque Faster R-CNN es generalmente más lento en inferencia que modelos de una sola etapa, como YOLOv3 o MobileNet, sigue siendo altamente valorado por su precisión y robustez en entornos donde la precisión es crítica. Su implementación está disponible en la API de Detección de Objetos de TensorFlow, con pesos preentrenados y soporte para herramientas como Tensorboard (ROBOFLOW, 2020).

La arquitectura de Faster R-CNN comienza con una CNN preentrenada para la extracción de características y utiliza la RPN para identificar regiones de interés. Estas regiones se pasan a través de la red para clasificar los objetos y ajustar sus cuadros delimitadores, utilizando herramientas de optimización como las máquinas

de vectores de soporte (SVM) para la clasificación final. El resultado es una detección de objetos precisa y bien localizada, ideal para aplicaciones de alta precisión en diversos campos. La Figura 7 ilustra la arquitectura general de Faster R-CNN (CLOUDFACTORY, 2024).



FUENTE: CLOUDFACTORY, 2024

Figura 7 — Arquitectura Faster R-CNN

3.2.2 Cultivo de tomates

El tomate es considerado un alimento básico de gran importancia, por su valor alimenticio y por la gran variedad en la que puede ser consumido, siendo un alimento ligeramente ácido estimula el apetito. Por lo tanto, el cultivo de tomates, es considerado un proceso muy minucioso y delicado. Primeramente, tendrá un proceso de labores de preparatorios, como considerarse el clima, el tipo se tierra; seguidamente un proceso de fertilización de la tierra, con fertilizantes naturales, con abonos minerales, etc.; luego llega el tiempo de la siembra, dependiendo el lugar tendrán su proceso de la siembra de las semillas de tomate, previamente seleccionadas (HERNANSÁEZ Y PASTOR, 1956).

3.2.1.1. Plagas del tomate

La planta de tomate es afectada por diferentes enfermedades, ya sean generados por hongos, bacterias, virus, plagas o agentes climáticos. Las plagas son los que ocasionan daño por otros seres vivos, conocidos como los parásitos o insectos.

La mosca blanca (Bemisia Tabaci)

Es un insecto que provoca diversos daños, por lo menos son 600 tipos de plantas silvestres o de cultivo que son afectados por esta especie alrededor del mundo, mayormente en las zonas tropicales o subtropicales. Este insecto ataca a la planta succionando la sabia en las hojas, extrayendo sus nutrientes, lo que provoca que la planta se debilite y no crezca como corresponde, sobre todo que, al momento de dar su fruto éste no se desarrolle bien ni llegue a una madurez adecuada. (CUÉLLAR Y MORALES, 2006)

Esta plaga se reproduce en cantidad a través de sus larvas por lo que su población se hace grande en poco tiempo, ver Figura 8.

FUENTE: SENASICA, 2020

Figura 8 — Mosca blanca (Bemisia tabaco)

Minador de las hojas (Liriomyza Trifolii)

Es una especie de larva de mosca que se desarrolla alimentándose de las hojas del tomate lo que provoca perforaciones al interior a manera de túnel, por lo que se observan líneas quebradas de color blanco. Esta plaga provoca mucho daño, impidiendo que la planta pueda realizar una fotosíntesis correcta para su desarrollo, y tienden a quebrase con facilidad, debido a sus heridas provoca que la planta pueda

ser susceptible a cualquier otros patógenos como hongos o bacterias, ver Figura 9. (GARCÍA- y otros, 2014).

FUENTE: INTAGRI, 2017

Figura 9 — Minador de las hojas (Liriomyza trifolii)

3.3 Marco conceptual

a) **Análisis Predictivo:** Es una metodología que emplea datos históricos, algoritmos estadísticos y modelos de Machine Learning para reconocer patrones y predecir eventos o comportamientos futuros. Se fundamenta en la idea de que los datos del pasado pueden servir para anticipar resultados futuros. Este enfoque se aplica en diversas áreas como la salud, finanzas, marketing y agricultura. Por ejemplo, puede predecir la aparición de plagas en los cultivos mediante el análisis de datos climáticos y de imágenes. El análisis predictivo aprovecha el aprendizaje a partir de la experiencia para anticipar comportamientos y facilitar la toma de decisiones fundamentadas. (ESPINO, 2017)

b) **Dataset (Conjunto de Datos):** Es una colección estructurada utilizada para análisis, entrenamiento de modelos de Machine Learning y evaluación de algoritmos. En Machine Learning, los datasets generalmente incluyen ejemplos etiquetados que representan las entradas y salidas deseadas para un modelo. Son fundamentales para el entrenamiento y validación, ya que permiten a los algoritmos aprender patrones y relaciones en los datos. (SZELISKI, 2021)

c) **Deep Learning:** (Aprendizaje Profundo) es una rama del Machine Learning, es un sistema de probabilidad que permite a las computadoras a aprender sobre datos con múltiples niveles de abstracción y que éstas lleguen a deducir respuestas automáticas (CALVO, GUZMÁN Y RAMOS, 2018).

d) **Entrenamiento de modelo:** Es el proceso mediante el cual un modelo de Machine Learning aprende a reconocer y detectar objetos en imágenes utilizando datos de entrada. El sistema analiza y descompone las imágenes para extraer características relevantes y ajusta sus parámetros en múltiples iteraciones para identificar patrones. El objetivo es que el modelo pueda realizar detecciones automáticas y precisas sin supervisión humana, facilitando tareas de reconocimiento autónomas. (ROZADA, 2021)

e) **Google Colab:** Es una herramienta colaborativa que permite programar y ejecutar código en Python directamente en la nube, eliminando la necesidad de configuraciones complejas en el equipo local. Una de sus mayores ventajas es el acceso gratuito a recursos como GPUs (unidades de procesamiento gráfico) y TPUs (unidades de procesamiento tensorial), lo que facilita la ejecución de modelos de aprendizaje profundo y tareas intensivas en cómputo. Google Colab se presenta como una herramienta tipo "Colaboratory" de Google, especialmente diseñada para fomentar la investigación y el desarrollo en campos como el aprendizaje automático (Machine Learning) e inteligencia artificial.

   Al usar la versión Pro de Google Colab, se accede a beneficios adicionales como tiempos de ejecución más largos, mayor capacidad de almacenamiento temporal y acceso prioritario a GPUs y TPUs de alto rendimiento, tales como NVIDIA T4, P100 y V100, que son altamente eficientes para el entrenamiento de modelos de aprendizaje profundo. Estas capacidades mejoradas permiten realizar experimentos de manera más rápida y efectiva, facilitando el ajuste de modelos complejos y el manejo de grandes conjuntos de datos.

   Google Colab también ofrece un entorno colaborativo en el que los usuarios pueden compartir y trabajar en cuadernos de código (notebooks) de manera sencilla, permitiendo la colaboración en tiempo real y la integración con Google Drive para almacenar el progreso. Esta herramienta ha ganado popularidad no solo por su facilidad de uso, sino también por democratizar el acceso a recursos avanzados de hardware que anteriormente eran costosos y difíciles de obtener, haciendo que la experimentación y el desarrollo en Machine Learning e inteligencia artificial sean más accesibles para estudiantes, investigadores y profesionales (GOOGLE COLAB, 2024).

f) **Inteligencia Artificial:** Es un campo de la informática dedicado al desarrollo de sistemas capaces de realizar tareas que normalmente requieren inteligencia humana,

como el aprendizaje, la resolución de problemas y la toma de decisiones. (RUSSELL Y NORVIG 2004)

g) **LabelImg**: Es una herramienta gráfica para la anotación y etiquetado de regiones de interés en imágenes. Este proceso implica asignar etiquetas o categorías a datos sin procesar, como imágenes, con el fin de prepararlos para el entrenamiento de modelos de Machine Learning. (CÓRDOVA, 2021)

h) **LOSS:** Mide la discrepancia entre las predicciones del modelo y los valores reales, evaluando cuánto se desvían las predicciones de las etiquetas correctas. Su objetivo es minimizar este valor para mejorar la precisión del clasificador. Un valor de pérdida cercano a 0 indica una alta precisión, ya que las predicciones son precisas y el modelo penaliza adecuadamente las clasificaciones incorrectas. Minimizar la función de pérdida ayuda a maximizar la precisión del clasificador y a mejorar su capacidad predictiva. (ZAMORANO, 2018)

i) **mAP (Mean Average Precision):** Es una métrica ampliamente utilizada para evaluar el rendimiento de modelos en detección de objetos y recuperación de información. Esta métrica ofrece una evaluación global de la precisión del modelo al integrar tanto la precisión como el recall a diferentes niveles de umbral. (REDMON y otros, 2016)

j) **Modelo de datos**: Un modelo de aprendizaje automático se genera entrenando un algoritmo con datos específicos. Tras el entrenamiento, el modelo puede hacer predicciones o proporcionar salidas basadas en nuevas entradas. Por ejemplo, un modelo predictivo, creado por un algoritmo de aprendizaje automático, ofrecerá predicciones basadas en los datos con los que fue entrenado. El aprendizaje automático es fundamental para desarrollar modelos analíticos eficaces. (MORENO, 2020)

k) **Plaga**: Se refiere a cualquier organismo que provoca daños significativos en cultivos, ganado, productos almacenados o en la salud humana y animal. Las plagas abarcan una amplia variedad de seres vivos, como insectos, ácaros, hongos, bacterias, virus, malezas, roedores, y otros organismos que impactan negativamente los recursos naturales y la producción agrícola. Estos organismos pueden causar pérdidas económicas considerables y representar un desafío constante para la gestión y protección de los recursos en diversos sectores. (GALAN ZAPATA, 2021)

l) **Python:** Es un lenguaje de programación de alto nivel, interpretado y de propósito general, destacado por su simplicidad y legibilidad. Creado por Guido van Rossum y lanzado en 1991, Python está diseñado para ser accesible y fácil de usar, con una sintaxis clara que prioriza la legibilidad del código. Como lenguaje orientado a objetos con

tipado dinámico, Python facilita el modularidad y la reutilización del código gracias a su intérprete y a una extensa biblioteca estándar. Estas características hacen de Python una herramienta versátil y eficaz para una amplia variedad de aplicaciones. (MARZAL y otros, 2014)

m) **Redes Neuronales Convolucionales (CNN):** Es una red neuronal de múltiples capas, inspirada en la estructura de la corteza visual de los animales, y es particularmente efectiva en el procesamiento de imágenes. En una CNN, las capas iniciales se encargan de identificar características básicas, como bordes, mientras que las capas más profundas combinan estas características para formar representaciones más complejas. Su principal ventaja radica en que cada sección de la red se entrena para cumplir una función específica, lo que reduce la cantidad de capas ocultas necesarias y acelera el proceso de entrenamiento. Además, su capacidad de mantener la invarianza ante la traslación de patrones las convierte en herramientas altamente eficaces para la detección de patrones en imágenes, abarcando desde rasgos simples hasta estructuras complejas. (CENTENO, 2019)

n) **Roboflow**: es una plataforma potente y fácil de usar que facilita el entrenamiento de modelos de visión por computadora de manera rápida y eficiente. Su interfaz intuitiva permite a los usuarios cargar, etiquetar y organizar imágenes, resaltando aspectos clave para optimizar los conjuntos de datos. Esta funcionalidad de etiquetado es especialmente útil para proyectos de detección de objetos, donde es fundamental contar con datos correctamente anotados para el entrenamiento de modelos. La facilidad de uso de Roboflow la convierte en una opción ideal tanto para principiantes como para profesionales en el campo de la inteligencia artificial, proporcionando herramientas de aumento de datos, gestión de versiones de datasets y exportación a diversos formatos de modelos (ROBOFLOW, 2023).

o) **TensorFlow**: Es una plataforma de código abierto para el aprendizaje automático que ofrece varios niveles de abstracción. Su ecosistema integral y flexible de herramientas y bibliotecas facilita la implementación de aplicaciones de aprendizaje automático, utilizando la API de alto nivel de Keras para simplificar el proceso. (CÓRDOVA PÉREZ, 2021)

p) **Visión por computadora:** Busca desarrollar métodos matemáticos para interpretar la estructura y apariencia tridimensional de objetos en imágenes, imitando la percepción visual humana. Aunque se han hecho avances en la reconstrucción 3D y segmentación de objetos, la capacidad de los sistemas para interpretar imágenes con la misma

precisión y comprensión que un humano sigue siendo inalcanzable. Esto se debe a la complejidad del problema, que requiere modelos avanzados que integren principios de radiometría, óptica y gráficos por computadora, enfrentando desafíos mayores que otros campos, como la modelización del tracto vocal. (SZELISKI, 2021)

q) **Data augmentation**: Son métodos que se usan para aprovechar al máximo los pocos ejemplos de datos de prueba y aumentar la precisión de modelos algorítmicos. Estas técnicas pueden ser escalar, rotar, reflejar horizontal o verticalmente, desplazar horizontal o verticalmente, hacer zoom, entre otros. Se espera que, con estas técnicas, el aumento de datos ayude a prevenir el sobreajuste (problema común cuando se tiene un conjunto de datos pequeño) y así el modelo mejore su capacidad para generalizar (ROMERO-y otros, 2017).

CAPÍTULO IV

METODOLOGÍA

4.1 Tipo y nivel de investigación

Tipo de investigación

La presente investigación está basada en tipo de ***investigación aplicada***, porque procura aplicar las técnicas del Machine Learning en el reconocimiento de imágenes para resolver un problema (HERNÁNDEZ, FERNÁNDEZ Y BAPTISTA, 2014).

Nivel de investigación

El nivel de la investigación será de ***nivel descriptivo***, ya que la máquina podrá aprender a base de patrones y características de tal forma pueda identificar el tipo de plagas, midiendo su eficiencia (HERNÁNDEZ, FERNÁNDEZ Y BAPTISTA, 2014).

4.2 Diseño de la investigación

En esta investigación se realizará el diseño no experimental de tipo transversal, ya que los datos o variables no serán manipulados de manera deliberada, se basará en la observación del comportamiento de la variable de estudio (HERNÁNDEZ, FERNÁNDEZ Y BAPTISTA, 2014).

4.3 Descripción ética de la investigación (si le corresponde)

No corresponde.

4.4 Población y muestra

Población

Se considerará una población total de 1160 fotografías de plagas en hojas de tomate, tomadas en cultivos ubicados en el anexo de Pachachaca, en la ciudad de Abancay, Apurímac.

Muestra

La muestra se basará en fotografías de alta resolución: el 90.26% de las fotografías se utilizarán para entrenamiento del modelo de aprendizaje, el 5.43% se reservará para la validación del modelo y el restante 4.31% se destinará para el proceso de prueba (testing).

4.5 Procedimiento

a) **Etapa I:** Aprobación del proyecto de tesis.

b) **Etapa II:** Análisis y evaluación de los modelos de Machine Learning.

c) **Etapa III:** Adquisición de imágenes para el entrenamiento del modelo, las imágenes serán conseguidas de la siguiente forma: se tomarán fotos de distintas partes de la planta de tomate, sobre todo a las hojas.

d) **Etapa IV:** Etiquetado de imágenes: se etiquetarán las imágenes para identificar y localizar las plagas.

e) **Etapa V**: Codificación de los modelos de detección: se implementarán los modelos YOLOv4 y Faster R-CNN utilizando Python y Google Colab.

f) **Etapa VI:** Entrenamiento del modelo, se usará el 90.26% de las fotografías para el entrenamiento y el 5.43% de fotografías para el proceso de validación.

g) **Etapa VII:** Proceso de pruebas (testing) de las imágenes, se usará el 4.31% de las fotografías.

h) **Etapa VIII:** Comparación y análisis de los resultados obtenidos con cada una de los modelos de Machine Learning.

i) **Etapa IX:** Desarrollo del Informe Final, detallado por los resultados obtenidos en la investigación.

4.6 Técnica e instrumentos

Técnica

Se utilizó la técnica de “observación” en dos modalidades: observación directa y estructurada. La observación directa permite analizar el comportamiento de las plagas en las hojas de tomate mediante fotografías, sin la intervención de intermediarios. A su vez, la observación es estructurada, ya que sigue un procedimiento sistemático y controlado, utilizando cámaras para capturar datos visuales que posteriormente son procesados por modelos de Machine Learning para el entrenamiento, validación y pruebas. (CAMPOS, COVARRUBIAS Y LULE MARTÍ-NEZ, 2012)

Instrumento

El instrumento utilizado fue la “ficha de observación”, para el aprendizaje supervisado, se basará en el conocimiento de la información de entrenamiento, es decir, se entrenará al sistema con una determinada cantidad de datos que ya estarán etiquetados a detalle de

manera correcta, a base de esta información el sistema tendrá la capacidad de definir sus patrones, y podrá procesar la nueva información sin etiquetas (SIMEONE, 2018).

4.7 Estadístico de investigación

- True Positives (TP)
- False Positives (FP)
- False Negatives (FN)
- True Negatives (TN)

- $\boldsymbol{Precision} = \frac{TP}{TP + FP}$
- $\boldsymbol{Recall} = \frac{TP}{TP + FN}$
- $\boldsymbol{F - Value} = 2 * \frac{Precision * Recall}{Precision + Recall}$

CAPÍTULO V
RESULTADOS Y DISCUSIÓN

5.1 Análisis de resultados

A continuación, se presentan los procesos y resultados obtenidos en esta investigación:

5.1.1 Aplicación del ML Lifecycle

1. **Recolección de Datos:** Para la detección de plagas en hojas de tomate, se realizó una campaña de recolección de imágenes en el anexo de Pachachaca, Abancay. Las imágenes fueron capturadas directamente en campo utilizando un teléfono Google Pixel 6a, seleccionado por la calidad de su cámara, que permite obtener imágenes con alta resolución y precisión en los detalles. Esto es crucial para identificar características específicas de las plagas.

   Las fotografías se centraron en las hojas de las plantas de tomate, que son las principales áreas afectadas por las plagas de interés: Mosca blanca (Bemisia tabaci) y Minador de las hojas (Liriomyza Trifolii). Se tomaron imágenes desde diferentes ángulos y distancias para capturar variaciones en la iluminación y en el contexto ambiental, garantizando así un conjunto de datos diverso y representativo.

   Además, se consideraron condiciones de luz natural en diferentes momentos del día para capturar las hojas en situaciones de iluminación variada, con el fin de mejorar la robustez del modelo en condiciones reales. Las imágenes fueron posteriormente organizadas y etiquetadas en función de la plaga presente, lo cual es esencial para el entrenamiento de los modelos de detección de objetos.

2. **Preparación y Etiquetado de Datos:**

   Después de recolectar 462 imágenes reales de hojas de tomate, se procedió a la etapa de preparación y etiquetado de datos para entrenar los modelos de detección de plagas. Este conjunto de datos original incluía imágenes con dos

tipos de plagas y algunas imágenes sin plagas. La distribución de las imágenes en el conjunto de datos original fue la siguiente:

- Clase 1: Mosca blanca (Bemisia tabaci): 247 imágenes
- Clase 2: Minador de las hojas (Liriomyza Trifolii): 193 imágenes
- Sin clase: 22 imágenes (imágenes sin presencia de plagas)

i. **Elección de herramientas de etiquetado**: Para el etiquetado de estas imágenes, inicialmente se utilizó la herramienta **LabelImg** debido a su simplicidad y la posibilidad de trabajar offline. Esto fue particularmente útil en las primeras etapas, ya que permitía etiquetar las imágenes localmente sin necesidad de conexión a internet. LabelImg permite cargar cada imagen, dibujar recuadros delimitadores alrededor de los objetos de interés (en este caso, las plagas) y asignar una etiqueta a cada recuadro. Ver figura 10.

Figura 10 — Etiquetado de imagen usando labelImg

ii. **Migración a Roboflow**: A medida que el proyecto avanzó, se decidió migrar el proceso de etiquetado a Roboflow para aprovechar sus funciones avanzadas y gestionar los datos de manera centralizada. Roboflow permite organizar y exportar los datos en formatos específicos para modelos como YOLOv4 y Faster R-CNN.

En Roboflow, cada una de las 462 imágenes originales fue revisada y etiquetada delimitadores alrededor de las áreas donde se observaban las plagas. A cada recuadro se le asignó una etiqueta específica que

identifica la clase de plaga presente en la imagen: Mosca blanca (Bemisia tabaci) o Minador de las hojas (Liriomyza Trifolii). Este etiquetado preciso es esencial para que los modelos puedan diferenciar entre ambas plagas y ubicarlas correctamente. Ver figura 11.

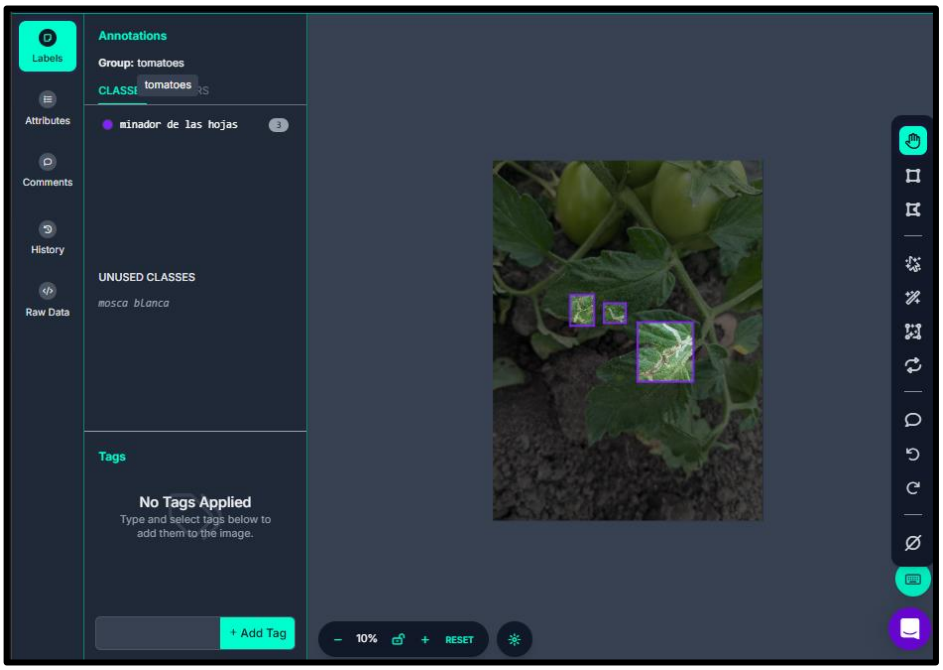


Figura 11 — Etiquetado de imagen usando roboflow

iii. **Modificaciones automáticas aplicadas por Roboflow**: Además del etiquetado, Roboflow aplicó algunas modificaciones automáticas a las imágenes para optimizar su preparación. Ver figura 12:

- **Auto-orientación**: ajuste automático de la orientación de las imágenes para asegurar una alineación uniforme.
- **Redimensionamiento**: todas las imágenes fueron escaladas a 640x640 píxeles para asegurar consistencia en las dimensiones.
- **Remapeo de clases**: verificación y ajuste de las etiquetas de plagas para asegurar que solo las clases relevantes estuvieran presentes en el conjunto de datos.

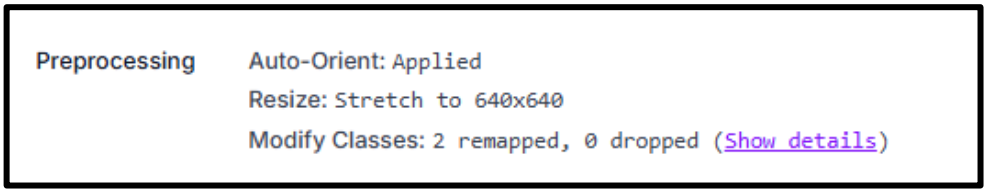


Figura 12 — Etiquetado de imagen usando roboflow

iv. **Aumento de Datos (Data Augmentation):** Una vez completado el etiquetado, se aplicó el aumento de datos (data augmentation) en Roboflow para ampliar el conjunto de datos y mejorar la capacidad de generalización del modelo. El aumento de datos es una técnica ampliamente utilizada en el entrenamiento de modelos de detección de objetos, que genera variaciones artificiales de las imágenes originales, aplicando transformaciones como rotación, cambio de escala y adición de ruido. Este proceso permite enriquecer el conjunto de datos, lo cual es especialmente valioso en proyectos donde la obtención de datos es limitada o costosa. Al exponer al modelo a diversas versiones de las mismas imágenes, el aumento de datos ayuda a mejorar su capacidad de generalización, permitiéndole adaptarse mejor a situaciones reales y variaciones en las condiciones de las imágenes (como cambios de iluminación, orientación y color) que pueda encontrar fuera del entorno controlado de entrenamiento.

En este caso, Roboflow aplicó los siguientes tipos de aumento de datos a las 462 imágenes originales, generando un conjunto de 1160 imágenes en total (Ver figura 13):

- Flip: inversión horizontal de las imágenes.
- Rotación: entre -17° y +17°, generando diferentes orientaciones de las plagas.
- Escala de grises: aplicado al 25% de las imágenes, para introducir variaciones de color.
- Ajuste de tono (Hue): variación de entre -25° y +25° en el tono de la imagen.
- Saturación: variación de entre -25% y +25% en la saturación de color.

- Ruido: adición de ruido de hasta un 5% de los píxeles, para simular condiciones de imagen menos ideales.

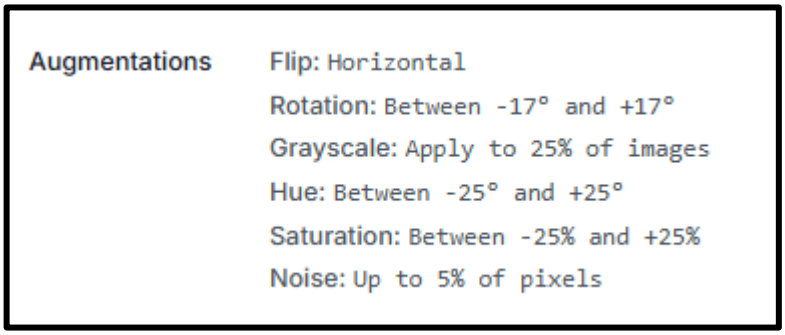


Figura 13 — Aumento de datos realizadas por roboflow

Un ejemplo de las modificaciones realizadas mediante data augmentation se muestra en la Figura 14.

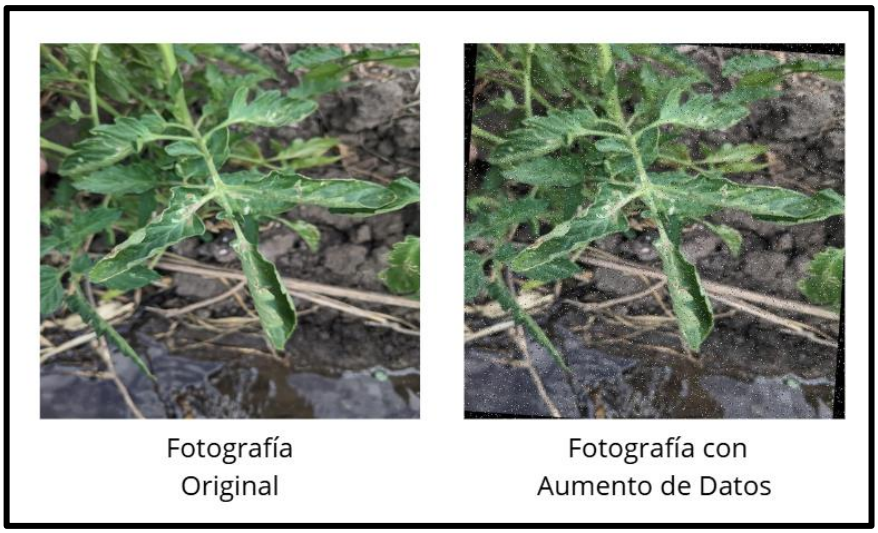


Figura 14 — Comparación entre fotografía original y fotografía con aumento de datos mediante data augmentation

v. **Exportación de datos en formatos específicos para los modelos**: Roboflow permite la exportación del conjunto de datos en formatos específicos para cada modelo. Para este proyecto, se utilizaron YOLOv4 y Faster R-CNN, cada uno con requisitos específicos:

- **Formato para YOLOv4**: exportación en archivos TXT con las coordenadas de los recuadros y la clase de cada objeto.
- **Formato para Faster R-CNN**: exportación en XML, con una estructura más detallada. Ver figura 15.

Figura 15 — Formato de anotaciones de Yolo V4 y Faster R-CNN

vi. **Distribución del conjunto de datos**: Después del aumento de datos, el conjunto de 1160 imágenes se distribuyó en Roboflow de la siguiente manera. Ver figura 16:

- Entrenamiento (90%): 1047 imágenes
- Validación (5%): 63 imágenes
- Testing (4%): 50 imágenes

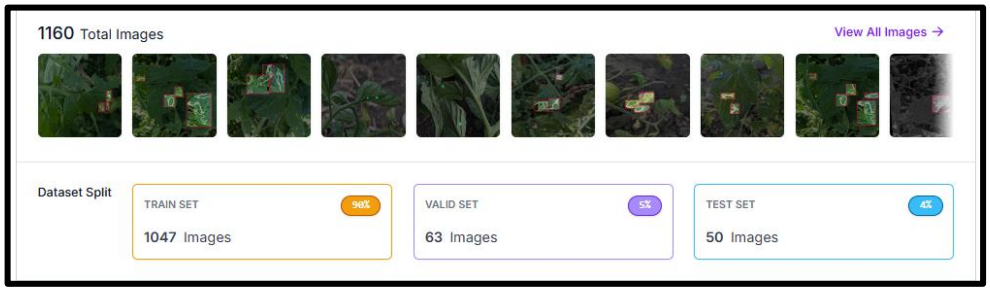


Figura 16 — Distribución del conjunto de datos en entrenamiento, validación y prueba

Aunque en la interfaz de Roboflow se observa una distribución aproximada de 90% para entrenamiento, 5% para validación y 4% para testing, estos valores suman un total de 99% en lugar del 100%. Esta discrepancia se debe a que Roboflow utiliza redondeo hacia números enteros para simplificar la visualización de los datos, presentando porcentajes aproximados que son más fáciles de interpretar de un vistazo. Este tipo de redondeo es común en plataformas de visualización de datos, ya que busca reducir la complejidad

en la interfaz gráfica, pero puede llevar a que la suma de los porcentajes mostrados no coincida exactamente con el total.

A nivel de cálculo, sin embargo, los valores exactos utilizados en el proyecto son:

- 90.26% de las imágenes para el entrenamiento de los modelos (1047 imágenes)
- 5.43% para la validación durante el entrenamiento (63 imágenes)
- 4.31% reservado para el testing final, para evaluar el rendimiento de los modelos ya entrenados (50 imágenes)

Estos porcentajes suman exactamente el 100% del total de 1160 imágenes, lo que asegura que cada imagen está correctamente distribuida entre los conjuntos de entrenamiento, validación y testing, sin dejar imágenes sin asignar. La precisión en esta distribución es importante para documentar de manera exacta el uso de los datos y garantizar la reproducibilidad de los experimentos. Por lo tanto, aunque Roboflow muestre una aproximación en su interfaz, los cálculos internos reflejan la distribución completa de las imágenes en cada conjunto de datos.

Esta diferencia entre la visualización aproximada y los cálculos exactos no afecta la calidad ni el equilibrio del conjunto de datos, pero es importante mencionarla para aclarar el proceso y evitar posibles confusiones sobre el uso y distribución real de las imágenes en el proyecto.
La preparación y etiquetado detallado de este conjunto de datos, incluyendo el aumento de datos, es fundamental para el entrenamiento en Python y Google Colab utilizando los modelos YOLOv4 y Faster R-CNN. Esto garantiza que el modelo aprenda a detectar plagas en hojas de tomate con precisión y eficacia.

3. Entrenamiento de Modelos

Para llevar a cabo el entrenamiento de los modelos de detección de plagas, se desarrollaron notebooks en Google Colab Pro, donde se implementó y configuró todo el código necesario para entrenar los modelos YOLOv4 y

Faster R-CNN en Python. El uso de Google Colab Pro proporcionó acceso a GPUs de alto rendimiento, como las NVIDIA T4 y P100, lo que permitió sesiones prolongadas y una mayor capacidad de procesamiento, esencial para el entrenamiento de modelos de machine learning. Ver figura 17.

Figura 17 — Características de Google Colab Pro

La capacidad de ejecutar sesiones prolongadas en Google Colab Pro permitió llevar a cabo múltiples iteraciones de entrenamiento de manera eficiente. El acceso a GPUs de alto rendimiento y la facilidad de monitoreo en tiempo real proporcionaron un entorno ideal para la optimización y ajuste de modelos complejos.

i. **Configuración y preparación de datos**: El conjunto de datos utilizado para el entrenamiento constaba de 1160 imágenes. Estas imágenes estaban etiquetadas y clasificadas en dos clases:
   - Clase 1: Mosca blanca (Bemisia tabaci)
   - Clase 2: Minador de las hojas (Liriomyza Trifolii)

Los datos se dividieron en 90.26% para el entrenamiento, 5.43% para la validación y 4.31% para el testing. Esta distribución asegura que el modelo pueda aprender de un conjunto amplio mientras se valida continuamente su desempeño en datos no vistos.

ii. **Implementación y configuración en los notebooks:** La configuración y el entrenamiento de los modelos se realizaron directamente en notebooks de Google Colab:

- **YOLOv4**: La implementación se realizó en Darknet, utilizando un archivo de configuración (.cfg) para definir la arquitectura y los parámetros del modelo.
    - **yolov4-obj.cfg**: Este archivo define la arquitectura de la red neuronal, los parámetros del modelo, como el tamaño del lote, las capas, la tasa de aprendizaje y otras configuraciones específicas del modelo YOLOv4. Este archivo es fundamental para ajustar el modelo a las necesidades específicas del proyecto, permitiendo personalizar el entrenamiento para mejorar la detección de plagas en hojas de tomate. Ver figura 18,

```
[yolo]
mask = 6,7,8
classes=2
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1
scale_x_y = 1.05
iou_thresh=0.213
cls_normalizer=1.0
iou_normalizer=0.07
iou_loss=ciou
nms_kind=greedynms
beta_nms=0.6
max_delta=5
```

Figura 18 — Archivo de configuración de YoloV4

- **obj.data**: Este archivo especifica la información necesaria sobre el conjunto de datos, como la cantidad de clases que se deben detectar (en este caso, Mosca blanca y Minador

de las hojas), las ubicaciones de los archivos de entrenamiento y validación, y el archivo de nombres de las clases. Ver figura 19.

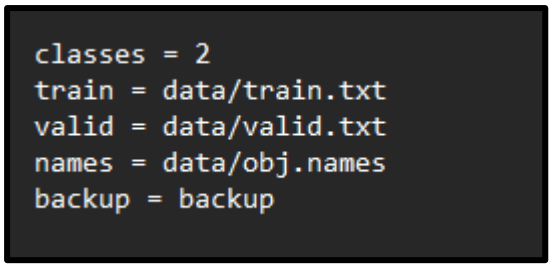


Figura 19 — Configuración de archivos para entrenamiento en YoloV4

- **obj.names**: Contiene una lista con los nombres de las clases que el modelo debe detectar. En este proyecto, incluye dos etiquetas: Mosca blanca y Minador de las hojas. Este archivo permite al modelo interpretar los resultados de la detección y asignar nombres a las predicciones de las clases detectadas. Ver figura 20.

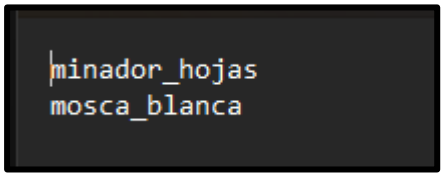


Figura 20 — Clases utilizadas para la detección de plagas en YoloV4

- **Faster R-CNN**: Para Faster R-CNN, se utilizó un entorno basado en TensorFlow. Utilizando un archivo de configuración (.pipeline) para definir los parámetros del modelo.
  - **pipeline.config**: Este archivo define la arquitectura del modelo, los hiperparámetros de entrenamiento, las configuraciones de optimización, el tamaño del lote, la tasa de aprendizaje, y las métricas. El pipeline.config también especifica la ubicación de los registros TFRecord

y los puntos de control (checkpoints) preentrenados que se utilizarán como base para el ajuste fino del modelo. Ver figura 21.

```
model {
  faster_rcnn {
    num_classes: 2
    image_resizer {
      keep_aspect_ratio_resizer {
        min_dimension: 640
        max_dimension: 640
        pad_to_max_dimension: true
      }
    }
    feature_extractor {
```

Figura 21 — Configuración del archivo pipeline.config para el modelo Faster R-CNN

- **label_map.pbtxt**: Este archivo mapea las clases de etiquetas (en este caso, Mosca blanca y Minador de las hojas) a identificadores numéricos. Este mapeo es necesario para que el modelo entienda cómo etiquetar y clasificar correctamente las detecciones durante el entrenamiento y la inferencia. Ver figura 22.

```
item {
  id: 1
  name: 'minador_hojas'
}

item {
  id: 2
  name: 'mosca_blanca'
}
```

Figura 22 — Clases utilizadas para la detección de plagas en Faster R-CNN

iii. **Monitoreo del entrenamiento**: Durante el entrenamiento, se realizó un monitoreo constante de las métricas clave para evaluar su desempeño:

- **Loss** (Pérdida): Se monitoreó el valor de la función de pérdida para garantizar que disminuyera de manera consistente, indicando que el modelo estaba aprendiendo de los datos de entrenamiento.
- **Precisión y Recall**: Estas métricas permitieron evaluar qué tan bien el modelo estaba clasificando las plagas detectadas y cuántas estaba identificando correctamente.
- **Intersección sobre Unión (IoU)**: Se utilizó para medir la precisión espacial de las detecciones, comparando qué tan bien los recuadros predichos se alineaban con las etiquetas reales.

vii. Exportación de los modelos entrenados:

- **YOLOv4**: Una vez finalizado el entrenamiento, el modelo guarda los pesos en un archivo .weights (por ejemplo, yolov4-obj_best.weights). Este archivo contiene los pesos optimizados del modelo, que se pueden utilizar para realizar inferencias sobre nuevas imágenes en el proceso de testing. El archivo de configuración (.cfg) y el archivo de datos (.data) deben ser utilizados en combinación con los pesos para cargar el modelo y realizar detecciones.
- **Faster R-CNN**: Al finalizar el entrenamiento, se realizó la exportación del modelo en formato TensorFlow SavedModel. Esto generó un archivo saved_model.pb y otros archivos asociados necesarios para realizar inferencias. Este formato permite cargar el modelo de manera eficiente para realizar pruebas y detecciones sobre nuevos datos.

En conclusión, el entrenamiento de YOLOv4 y Faster R-CNN en Google Colab resultó en modelos preparados para detectar y clasificar plagas en hojas de tomate con niveles de precisión adecuados, listos para ser evaluados y validados mediante el conjunto de testing.

4. **Evaluación con Datos de Testing**

Tras el entrenamiento de los modelos, se procedió a la evaluación utilizando las imágenes del conjunto de testing, que constaban del 4.31% del total de imágenes (50 imágenes). El proceso de testing se llevó a cabo en notebooks de Google Colab para cada uno de los modelos, empleando configuraciones específicas para la inferencia.

- **YOLOv4**: Se cargaron los pesos entrenados (yolov4-obj_best.weights) y se ejecutó la inferencia sobre las imágenes del conjunto de testing. Utilizando Darknet, el modelo predijo las ubicaciones de las plagas y sus etiquetas de clase. Los resultados predichos fueron visualizados, mostrando las detecciones realizadas por el modelo sobre cada imagen. Ver figura 23.

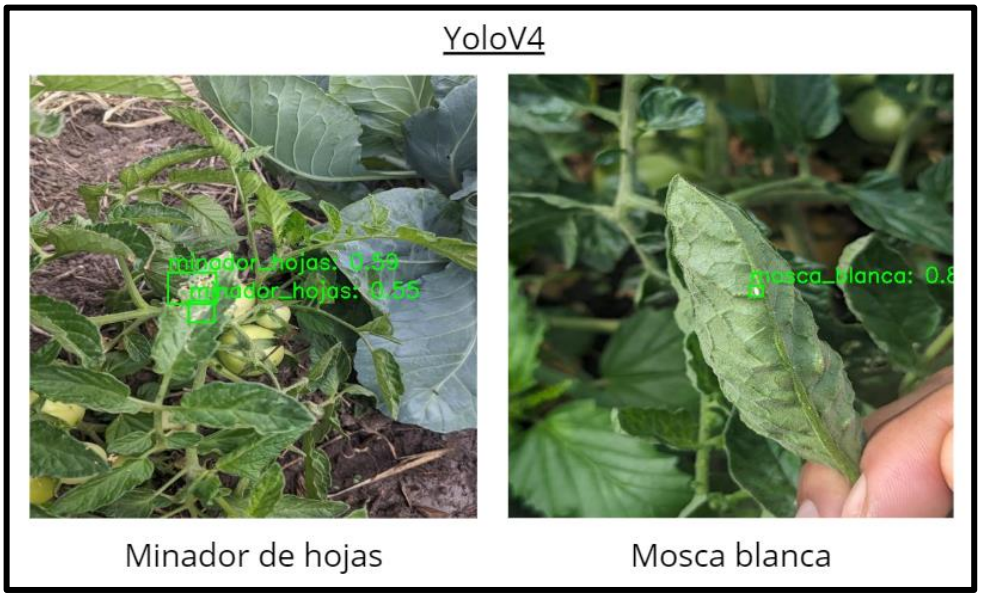


Figura 23 — Predicción de plagas realizada por Yolo V4

- **Faster R-CNN**: El proceso de evaluación para este modelo se llevó a cabo utilizando la API de TensorFlow Object Detection. El modelo, previamente exportado como saved_model.pb, se cargó y realizó inferencias sobre el conjunto de testing. Las predicciones incluyeron la detección de las plagas y sus correspondientes puntuaciones de confianza. Ver figura 24.

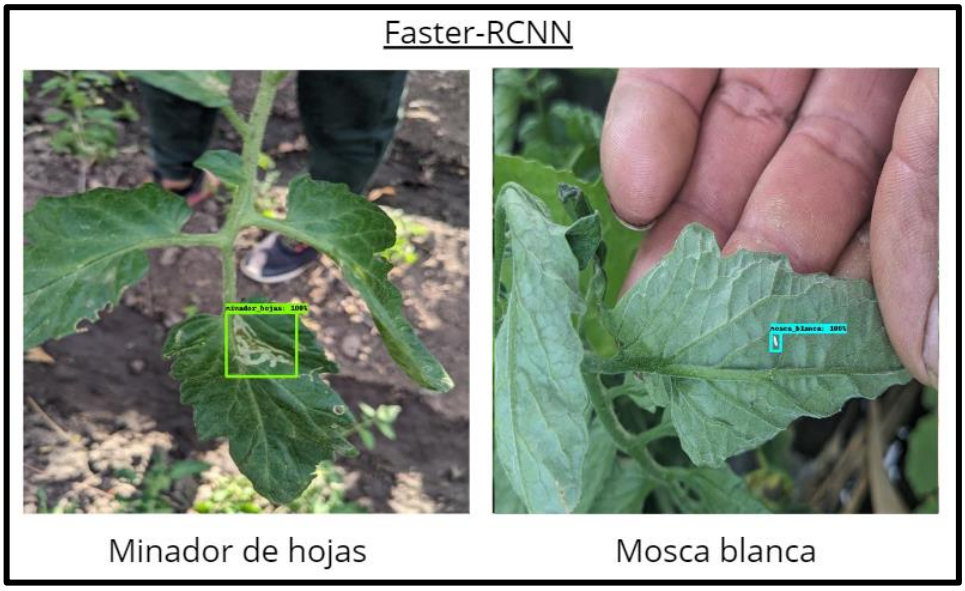


Figura 24 — Predicción de plagas realizada por Faster R-CNN

Ambos procesos de evaluación se centraron en analizar la capacidad de los modelos para identificar de manera precisa y rápida las plagas Mosca blanca y Minador de las hojas en las imágenes.

5. Análisis de Resultados

El análisis de la eficiencia de los modelos YOLOv4 y Faster R-CNN se llevó a cabo utilizando las métricas de precisión, recall y F1-Value como indicadores clave, así como la media armónica del F1-Value a nivel general para determinar qué modelo es más eficiente en la detección de plagas. Este enfoque equilibrado permitió evaluar la capacidad de cada modelo para identificar correctamente plagas en las hojas de tomate, considerando tanto su precisión como su capacidad de recuperación de manera integrada:

- **Preparación de los resultados de testing**: Se utilizaron las predicciones generadas durante el proceso de evaluación de ambos modelos en el conjunto de testing. Estas predicciones incluyeron las detecciones realizadas sobre cada imagen de prueba, con información detallada sobre la clase de plaga identificada (Mosca blanca o Minador de las hojas).

- **Conteo de Detecciones y Comparación de Resultados**: Se realizó un conteo de detecciones para evaluar el desempeño de cada modelo y determinar el número de:
    - **Detecciones completas**: Cuando el modelo detecta todas las instancias reales de la clase de plaga.
    - **Detecciones adicionales**: Cuando el modelo identifica más instancias de las que realmente existen.
    - **Detecciones incompletas**: Cuando el modelo detecta menos instancias de las que realmente existen.
    - **Sin detección**: Cuando el modelo no detecta ninguna instancia de la clase presente.

  Este conteo es necesario para calcular métricas de evaluación clave que ofrecen una visión detallada de la eficiencia de los modelos, tales como:
    - **Precisión (Precision**): Proporción de detecciones correctas sobre el total de detecciones realizadas por el modelo.
    - **Exhaustividad (Recall)**: Proporción de detecciones correctas sobre el total de instancias reales presentes en el conjunto de testing.
    - **Valor F (F1-Value):** Calculado como la media armónica entre la precisión y el recall, proporciona un balance entre ambas métricas y es especialmente útil cuando existe un desbalance entre falsos positivos y falsos negativos.

  El objetivo de este análisis es identificar cómo los modelos manejan las detecciones y la relación entre la precisión y el recall, permitiendo una evaluación detallada de su eficiencia en la detección de plagas en hojas de tomate.

- **Tabla de Etiquetas Reales del Conjunto de Testing**

  La siguiente tabla N° 3 muestra las etiquetas reales para las dos clases de plagas presentes en el conjunto de testing. Esta información sirve como referencia para comparar las predicciones realizadas por los modelos y calcular las métricas de evaluación:

Tabla 3 — Etiquetas reales del conjunto de testing para las clases mosca blanca y minador de las hojas

Código	Etiquetas Reales - Mosca Blanca	Etiquetas Reales - Minador de Hojas
IMG01	1	0
IMG02	1	0
IMG03	0	4
IMG04	1	0
IMG05	0	0
IMG06	0	3
IMG07	1	0
IMG08	0	1
IMG09	0	2
IMG10	0	3
IMG11	0	2
IMG12	0	1
IMG13	0	2
IMG14	0	4
IMG15	0	1
IMG16	1	0
IMG17	1	0
IMG18	1	0
IMG19	0	2
IMG20	0	3
IMG21	0	1
IMG22	0	3
IMG23	1	0
IMG24	1	0
IMG25	0	2
IMG26	0	2
IMG27	0	3
IMG28	1	0
IMG29	0	1
IMG30	0	2
IMG31	2	0
IMG32	0	4
IMG33	0	1
IMG34	0	2
IMG35	0	1
IMG36	0	0
IMG37	1	0
IMG38	1	0

IMG39	0	3
IMG40	0	2
IMG41	0	2
IMG42	1	0
IMG43	2	0
IMG44	2	0
IMG45	2	0
IMG46	2	0
IMG47	0	1
IMG48	1	0
IMG49	0	0
IMG50	0	2
Total	**24**	**60**

5.1.2 Resultado del procesamiento de datos

Para obtener los resultados, se llevó a cabo el entrenamiento de dos modelos: YOLOv4 y Faster R-CNN, utilizando fotografías de alta resolución que muestran dos plagas específicas: Mosca Blanca y Minador de las Hojas. El proceso de entrenamiento se realizó siguiendo la metodología ML Lifecycle, abarcando desde la recolección de datos hasta el análisis de los resultados, conforme a los puntos descritos previamente. La herramienta principal empleada fue Google Colab, que facilitó la ejecución del código, así como el manejo y procesamiento de las imágenes almacenadas en Google Drive.

En el Anexo 01 y 02 se muestran las fotografías de las hojas de tomate con ambas plagas, estas imágenes se encuentran almacenadas en Google Drive con la cuenta de usuario Robert Huamán Cáceres.

El acceso a estas imágenes y al contenido del proyecto puede ser revisado por cualquier interesado a través del siguiente enlace: https://drive.google.com/drive/folders/1BfEkffJsLiZ-qgEDGkVxrdkqEFxv2nC4?usp=sharing, podemos encontrar 2 carpetas: Yolo y Faster-RCNN.

Yolo v4:

a) Entre los archivos encontraremos unos archivos con la extensión ". ipynb":

- **tomatoe_model_train_yolo.ipynb** : Tenemos el archivo donde se encuentra el código utilizado para entrenar las imágenes con el modelo Yolo v4. En el Anexo 06, Figura 93 , se muestra el código fuente o podemos acceder directamente a Google Colab a través del

link: https://colab.research.google.com/drive/1rdf4CxvV66ynONhmELuUFVKCGh13TPPt

- **tomatoe_model_prediction_yolo.ipynb** : Se muestra el código utilizado para realizar predicciones con imágenes utilizando el modelo YOLO V4 previamente entrenado. En el Anexo 06, Figura 94, se muestra el código fuente o podemos acceder directamente a Google Colab a través del link: https://colab.research.google.com/drive/1rdjjKtfHGW8n3F4cuH-ccUIdn7HTViZ4

b) También encontraremos carpetas con las fotos del cultivo de tomate:

- **images_train**: Se encuentran las fotos con las que se realizó el entrenamiento.
- **images_evaluated**: Se encuentran las fotos evaluadas por el modelo Yolo v4.

TensorFlow:

a) Entre los archivos encontraremos unos archivos con la extensión ". ipynb":

- **tomatoe_model_train_rcnn.ipynb :** Se encuentra el código con el que se entrenó con las imágenes con el modelo Faster R-CNN. En el Anexo 07, Figura 95 se muestra el código fuente o podemos acceder directamente a Google Colab a través del link: https://colab.research.google.com/drive/1v6bzQYZnNe366rHep94SUqLYP4QTTRYs
- **tomatoe_model_prediction_rcnn.ipynb** : Se muestra el código donde se realizó las predicciones con las imágenes con el modelo Faster R-CNN ya entrenado. En el Anexo 07, Figura 96 se muestra el código fuente o podemos acceder directamente a Google Colab a través del link: https://colab.research.google.com/drive/1v6wisnVKv_hPqkCUK072wkSg-FF_GJDm
-

b) También encontraremos carpetas con las fotografías:

- **images_train**: Se encuentran las fotografías con las que se realizó el entrenamiento con el modelo Faster R-CNN.

- **images_prediction_evaluated**: Se encuentran las fotografías evaluadas por el modelo Faster R-CNN.

5.2 Contrastación de información

5.2.1 Recopilación de datos

Para el procesamiento de datos, se utilizó un único conjunto compuesto por un total de 1160 fotografías. Del total, el 90.26% (1047 imágenes) se asignó específicamente para el entrenamiento del modelo. Estas imágenes muestran hojas de tomate en diversas condiciones, incluyendo aquellas afectadas por plagas como la Mosca blanca (Bemisia tabaci) y el Minador de las hojas (Liriomyza Trifolii), así como hojas saludables sin rastro de plagas. El 5.43% (63 imágenes) del conjunto se reservó para la validación del desempeño del modelo, mientras que el 4.31% (50 imágenes) restante se destinó a las pruebas finales.

El modelo se entrenó utilizando un único conjunto de datos que incluía todas las clases (Mosca blanca, Minador de las hojas). Esto permitió que los modelos YOLOv4 y Faster R-CNN aprendieran a detectar ambas plagas de manera simultánea, sin segmentar el conjunto de datos en procesos independientes. La distribución específica de las imágenes por clase se detalla a continuación en la tabla 4:

Tabla 4 — Distribución de imágenes por conjunto y clase

Conjunto	Clase	Cantidad de Imágenes
Entrenamiento (1047 imágenes)	Mosca blanca (Bemisia tabaci)	561
	Minador de las hojas (Liriomyza Trifolii)	435
	Sin clase (imágenes sin plagas)	53
Validación (63 imágenes)	Mosca blanca (Bemisia tabaci)	41
	Minador de las hojas (Liriomyza Trifolii)	20
	Sin clase (imágenes sin plagas)	2
Testing (50 imágenes)	Mosca blanca (Bemisia tabaci)	19
	Minador de las hojas (Liriomyza Trifolii)	28
	Sin clase (imágenes sin plagas)	3

Cabe destacar que esta distribución sigue la recomendación de Yoshua Bengio en su investigación "Practical Recommendations for Gradient-Based Training of Deep Architectures" (2012), que sugiere utilizar la mayor cantidad de datos posible para el entrenamiento, especialmente cuando los datos son limitados. Al destinar el 90.26% (1047 imágenes) para el entrenamiento, se maximizó la capacidad del modelo para aprender patrones complejos. Bengio también recomienda utilizar el 5.43% (63 imágenes) para la validación, lo cual facilita el ajuste de hiperparámetros y ayuda a prevenir el sobreajuste. Además, Ron Kohavi, en su estudio "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection" (1995), destaca la importancia de un conjunto de prueba independiente para evaluar objetivamente el modelo. Aunque el 4.31% (50 imágenes) destinado al testing puede parecer pequeño, proporciona una indicación valiosa de la efectividad del modelo.

Para obtener una comprensión más detallada de los datos utilizados en el entrenamiento, se recomienda consultar el Anexo 03. Para examinar las pruebas de

los datos y las detecciones realizadas por cada modelo, se sugiere revisar el Anexo 04.

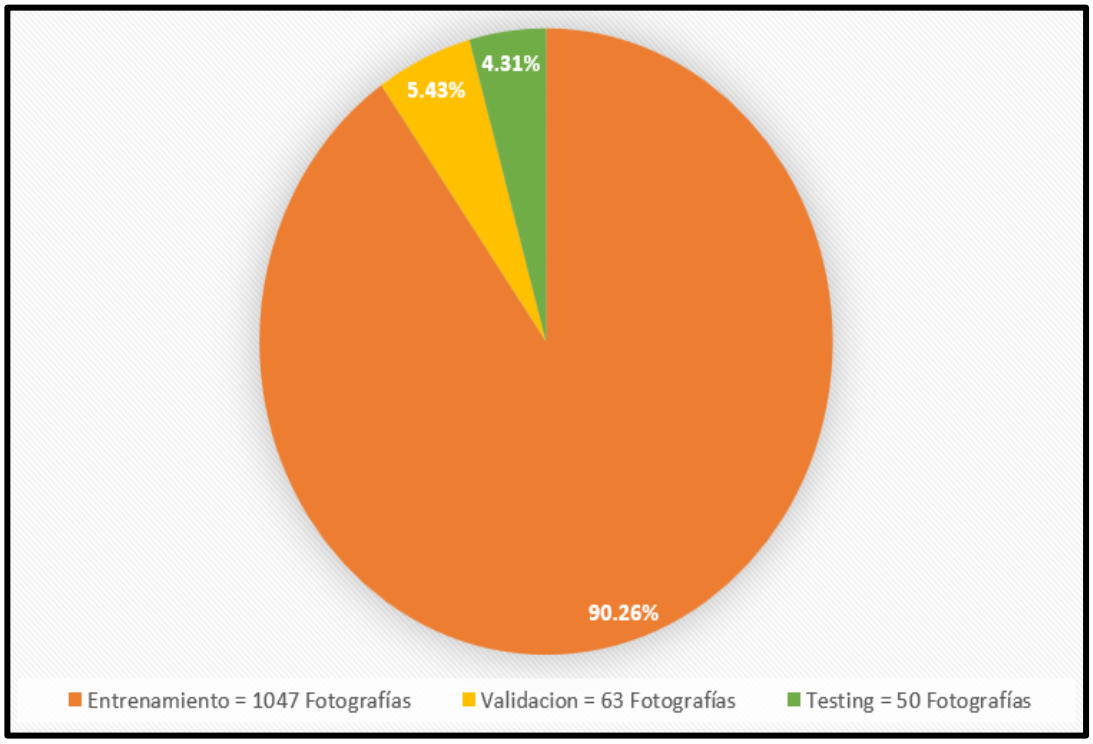


Figura 25 — Detalle de porcentaje de distribución de fotos

Es importante aclarar que, al calcular los valores de la matriz de confusión, es posible que los resultados no coincidan completamente con los totales de detección reportados por cada modelo debido a un margen de error inherente en las tecnologías de inteligencia artificial y aprendizaje automático (Machine Learning).

a) La mosca blanca

Dentro del conjunto único de 1160 imágenes, 561 imágenes contenían la presencia de Mosca blanca (Bemisia tabaci). Estas imágenes, distribuidas en los conjuntos de entrenamiento, validación y testing, se combinaron con imágenes que contenían Minador de las hojas y sin presencia de plagas. Esto permitió que ambos modelos aprendieran a detectar múltiples plagas de manera simultánea. En las figuras 26 y 27 se muestran ejemplos de detección de Mosca blanca con los modelos YOLOv4 y Faster R-CNN, respectivamente.

Figura 26 — Hojas de tomate con presencia de la mosca blanca evaluada por el modelo Yolo v4

Figura 27 — Hojas de tomate con presencia de la mosca blanca evaluada por el modelo Faster R-CNN

b) Minador de las hojas

De manera similar, el mismo conjunto de 1160 imágenes incluyó 435 imágenes etiquetadas para Minador de las hojas (Liriomyza Trifolii), distribuidas junto con imágenes de Mosca blanca y sin plagas para el entrenamiento, validación y testing. Este enfoque integrado permitió que los modelos aprendieran de manera conjunta.

En las figuras 28 y 29 se presentan ejemplos de detección de Minador de las hojas con los modelos YOLOv4 y Faster R-CNN.

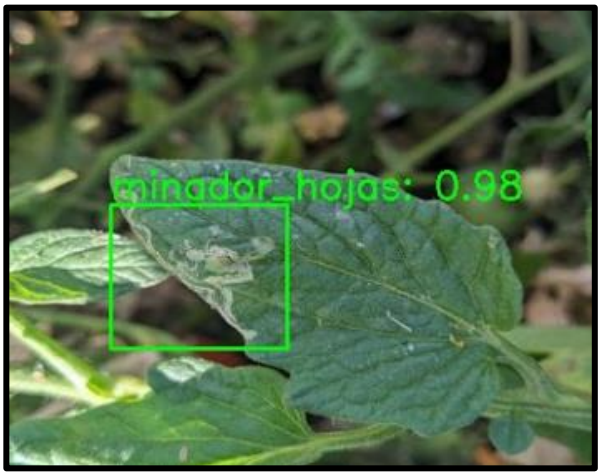


Figura 28 — Hojas de tomate con presencia de minador de las hojas evaluada por el modelo Yolo v4

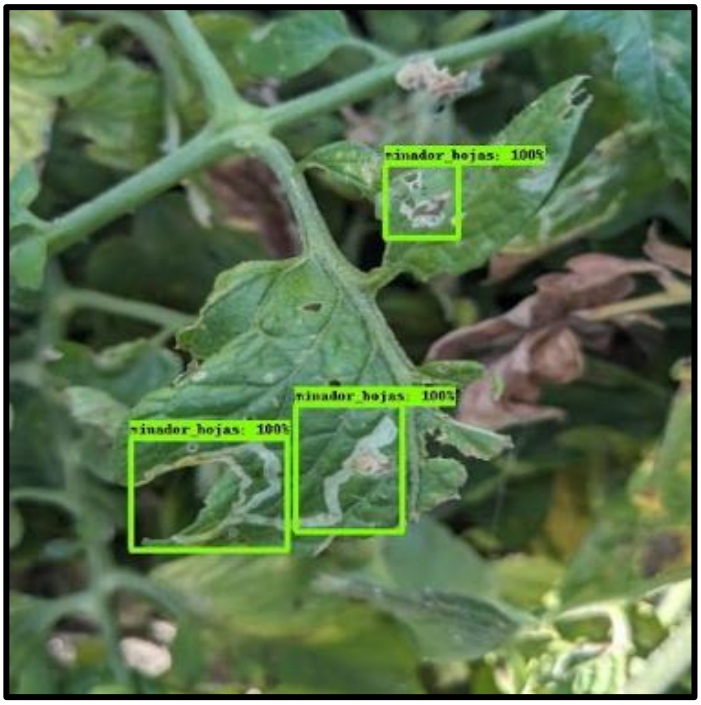


Figura 29 — Hojas de tomate con presencia de minador de las hojas evaluada por el modelo Faster R-CNN

5.2.2 Estadística de entrenamiento

5.2.2.1. Tendencia del LOSS (Pérdidas) durante el entrenamiento del modelo Yolo v4

La siguiente figura (Figura 30) ilustra la variación del valor del LOSS durante el proceso de entrenamiento del modelo YOLOv4. En el contexto de aprendizaje automático, el LOSS representa una medida de la diferencia entre las predicciones del modelo y las etiquetas reales de los objetos en las imágenes. Durante el entrenamiento, el objetivo es reducir este valor, ya que una disminución constante en el LOSS indica que el modelo está aprendiendo y mejorando su capacidad para realizar predicciones precisas.

El entrenamiento de YOLOv4 se realizó utilizando 1047 imágenes para el conjunto de entrenamiento, distribuidas en lotes efectivos de 4 imágenes (configurado a través de un batch size de 64 con subdivisiones de 16). Adicionalmente, se emplearon 63 imágenes para el proceso de validación, lo que permitió realizar ajustes continuos en los hiperparámetros del modelo y evaluar el rendimiento de manera intermedia durante el proceso de aprendizaje. El modelo fue entrenado durante un total de 2000 iteraciones, lo que resultó en aproximadamente 7.64 épocas completas sobre todo el conjunto de entrenamiento. Esto significa que cada imagen del conjunto de entrenamiento fue utilizada aproximadamente 7.64 veces para ajustar los pesos del modelo, permitiendo un aprendizaje más profundo y específico.

El LOSS disminuyó gradualmente a medida que avanzaba el proceso de entrenamiento, lo que indica que el modelo fue mejorando su capacidad de detección y localización de objetos. La figura también muestra el progreso del mAP (Mean Average Precision), una métrica clave que evalúa la precisión promedio del modelo para detectar las clases objetivo (en este caso, Mosca blanca y Minador de las hojas). Un mAP más alto sugiere que el modelo es más preciso y confiable, mientras que una reducción constante en el LOSS confirma que el modelo está optimizando su rendimiento.

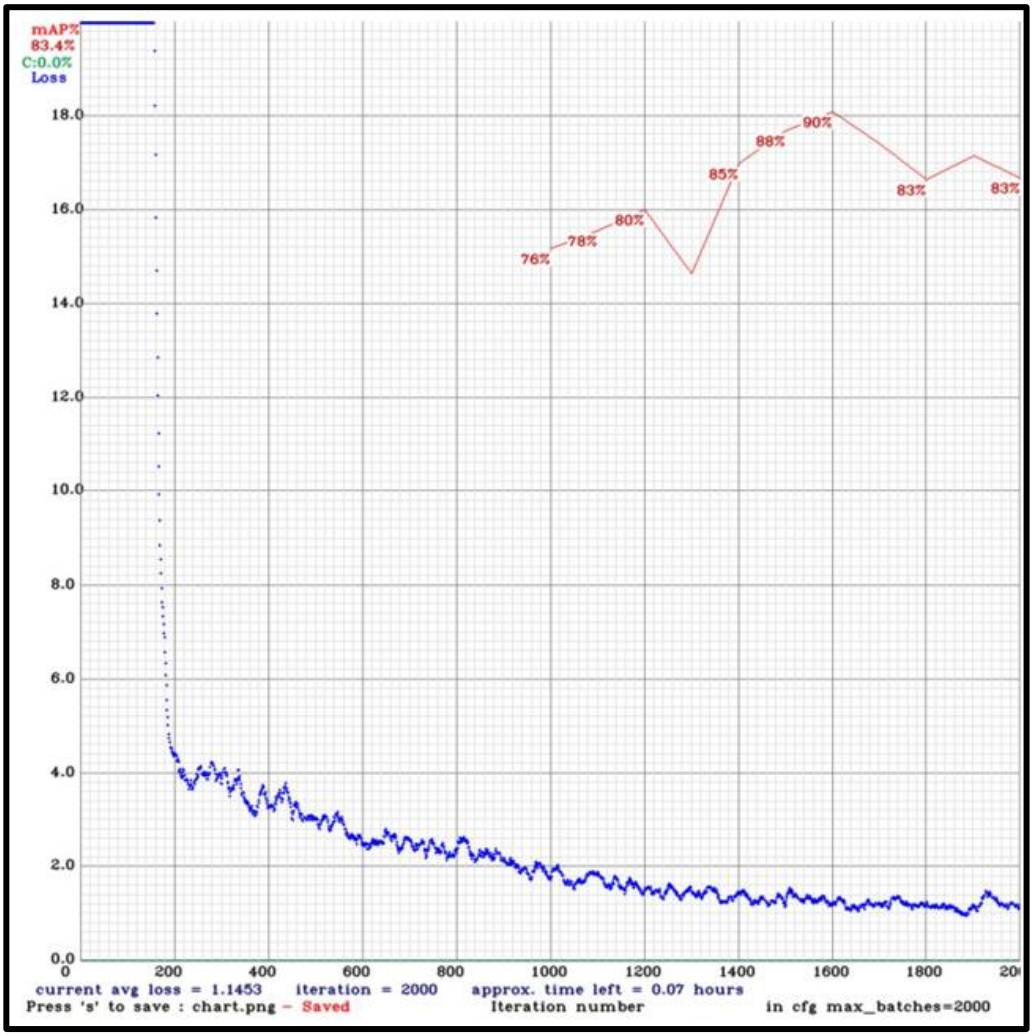


Figura 30 — Evolución del LOSS durante el entrenamiento del modelo Yolov4

La figura 30 ilustra la evolución del LOSS y del mAP durante el entrenamiento del modelo. Los elementos clave que se destacan son:

- **Curva azul (LOSS)**: Representa la reducción del valor de las pérdidas durante el entrenamiento.
- **Curva roja (mAP)**: Indica la evolución del mAP a lo largo de las iteraciones.
- **Iteraciones**: Número de actualizaciones de los pesos del modelo.
- **Eje Y (LOSS y mAP):** Muestra los valores del LOSS y el mAP alcanzados en cada punto del proceso.
- **Eje X (Iteraciones)**: Representa el número de iteraciones realizadas.

5.2.2.1. Tendencia del LOSS (Pérdidas) durante el entrenamiento del modelo Faster R-CNN

La figura 31 muestra la evolución del valor del LOSS durante el proceso de entrenamiento del modelo Faster R-CNN. En este contexto, el LOSS refleja la diferencia entre las predicciones generadas por el modelo y las etiquetas reales asociadas a los objetos en las imágenes. La disminución progresiva de este valor indica que el modelo está aprendiendo a detectar y clasificar las plagas en las hojas de tomate con mayor precisión, lo cual es el objetivo primordial durante el entrenamiento.

El modelo Faster R-CNN fue entrenado utilizando 1047 imágenes del conjunto de entrenamiento, junto con un conjunto de 63 imágenes para validación, durante un total de 25,000 pasos. Con una configuración de batch size de 4 imágenes por iteración, esto equivale aproximadamente a 96 épocas de entrenamiento (donde una época representa pasar por el conjunto de entrenamiento completo una vez).

El LOSS total reportado en el proceso de entrenamiento se descompone en varias métricas, incluidas las pérdidas de clasificación y localización tanto para la red de propuesta de regiones (RPN) como para el clasificador de cuadros delimitadores (Box Classifier). La disminución constante del total_loss durante el entrenamiento indica que el modelo fue capaz de ajustar sus predicciones y mejorar su precisión a medida que avanzaba.

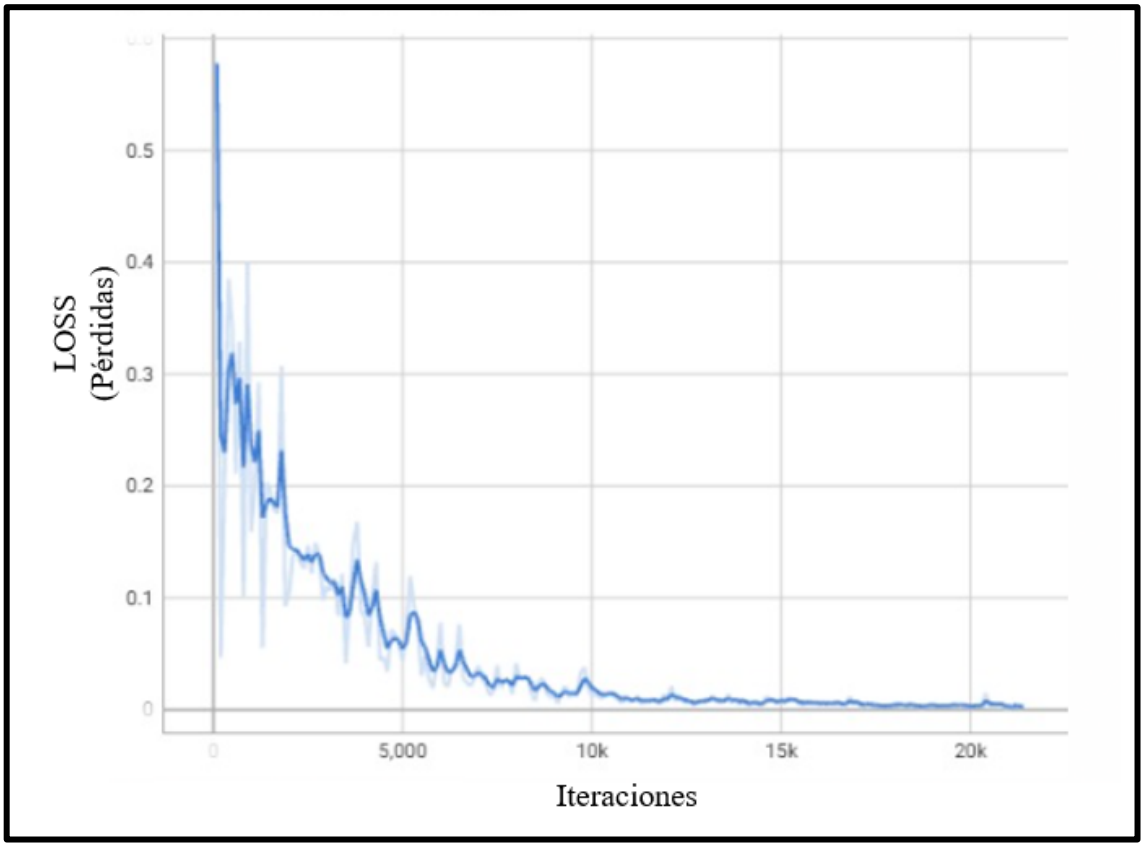


Figura 31 — Evolución del LOSS durante el entrenamiento del modelo Faster R-CNN

El Figura 31 ilustra la evolución del LOSS durante el entrenamiento del modelo. Los elementos clave que se destacan son:

- **Curva del LOSS**: Representa la disminución de las pérdidas a lo largo del entrenamiento.
- **Iteraciones**: Muestra el progreso del entrenamiento en términos de pasos o iteraciones.
- **Eje Y (LOSS):** Indica el valor del total_loss durante cada iteración del entrenamiento.
- **Eje X (Iteraciones):** Representa el número de pasos realizados durante el entrenamiento.

5.2.3 Contrastación de objetivos

5.2.3.1 Contrastación del objetivo 1:

Comprobar la eficiencia de clasificación del modelo generado con Yolo v4 en la detección de plagas en las hojas de los tomates.

A) La mosca blanca (Bemisia Tabaci)

a) Especificación de datos

La verificación de los resultados obtenidos en la detección de la plaga "Mosca blanca" se realizó utilizando un conjunto de pruebas que representa el 4.31% de las imágenes recolectadas, lo que equivale a un total de 50 fotografías.

En la Tabla N° 5, se presenta un desglose detallado de la evaluación del modelo YOLOv4 para esta plaga, con una estructura que permite interpretar claramente los resultados obtenidos. A continuación, se describe cada columna de la tabla:

- **Columna "Código (COD)":** Esta columna contiene un identificador único asignado a cada imagen del conjunto de pruebas, utilizando un formato de referencia que comienza con "IMG" seguido de un número secuencial (por ejemplo, IMG01, IMG02, IMG03, etc.).
- **Columna "Etiquetas Reales":** En esta columna se muestra el conteo exacto de instancias de la plaga "Mosca blanca" presentes en cada imagen, determinado mediante un proceso manual de etiquetado.
- **Columna "Detecciones"**: Esta columna indica la cantidad de instancias de la plaga detectadas por el modelo YOLOv4 en cada imagen evaluada.
- **Columna "Comentarios"**: Proporciona una descripción cualitativa del resultado de la detección, clasificando las predicciones del modelo en una de las siguientes categorías:
    - **Detección completa**: Cuando el número de instancias detectadas por el modelo coincide exactamente con el conteo real.
    - **Detección adicional (n de más):** Cuando el modelo detecta más instancias de las que existen en realidad (falsos positivos), siendo "n" la cantidad de instancias adicionales.

- **Detección incompleta (faltan n)**: Cuando el modelo detecta menos instancias de las que existen en realidad (falsos negativos), siendo "n" la cantidad de instancias faltantes.
- **Sin detección (correcto):** Cuando tanto la etiqueta real como la predicción del modelo son "0", es decir, no se detectó ninguna instancia y esto es correcto.

Tabla 5 — Detalle de etiquetas reales y detecciones de la mosca blanca con Yolo v4

Yolo v4			
Mosca Blanca			
Código	**Etiquetas Reales**	**Detecciones**	**Comentarios**
IMG1	1	1	Detección completa
IMG2	1	2	Detección adicional (1 de más)
IMG3	0	0	Sin detección (correcto)
IMG4	1	1	Detección completa
IMG5	0	0	Sin detección (correcto)
IMG6	0	0	Sin detección (correcto)
IMG7	1	1	Detección completa
IMG8	0	0	Sin detección (correcto)
IMG9	0	0	Sin detección (correcto)
IMG10	0	0	Sin detección (correcto)
IMG11	0	0	Sin detección (correcto)
IMG12	0	0	Sin detección (correcto)
IMG13	0	0	Sin detección (correcto)
IMG14	0	0	Sin detección (correcto)
IMG15	0	0	Sin detección (correcto)
IMG16	1	1	Detección completa
IMG17	1	1	Detección completa
IMG18	1	1	Detección completa
IMG19	0	0	Sin detección (correcto)
IMG20	0	0	Sin detección (correcto)
IMG21	0	0	Sin detección (correcto)
IMG22	0	0	Sin detección (correcto)
IMG23	1	1	Detección completa
IMG24	1	1	Detección completa
IMG25	0	0	Sin detección (correcto)
IMG26	0	0	Sin detección (correcto)
IMG27	0	0	Sin detección (correcto)
IMG28	1	1	Detección completa
IMG29	0	0	Sin detección (correcto)
IMG30	0	0	Sin detección (correcto)

IMG31	2	2	Detección completa
IMG32	0	0	Sin detección (correcto)
IMG33	0	0	Sin detección (correcto)
IMG34	0	0	Sin detección (correcto)
IMG35	0	0	Sin detección (correcto)
IMG36	0	0	Sin detección (correcto)
IMG37	1	1	Detección completa
IMG38	1	1	Detección completa
IMG39	0	0	Sin detección (correcto)
IMG40	0	0	Sin detección (correcto)
IMG41	0	0	Sin detección (correcto)
IMG42	1	1	Detección completa
IMG43	2	2	Detección completa
IMG44	2	2	Detección completa
IMG45	2	2	Detección completa
IMG46	2	2	Detección completa
IMG47	0	0	Sin detección (correcto)
IMG48	1	2	Detección adicional (1 de más)
IMG49	0	1	Detección adicional (1 de más)
IMG50	0	0	Sin detección (correcto)
Total	24	26	

- **Ejemplo Explicativo**

  Consideremos la fila correspondiente a "IMG48" como ejemplo:

  - **Código (COD):** IMG48
  - **Etiquetas Reales**: El valor es "1", indicando que hay una instancia de "Mosca blanca" presente en la imagen, según el etiquetado manual.
  - **Detecciones**: El modelo detectó "3" instancias, lo que significa que identificó una instancia adicional que no estaba presente en la realidad.
  - **Comentarios**: "Detección adicional (1 de más)" indica que el modelo YOLOv4 identificó incorrectamente una instancia adicional, lo que resulta en un falso positivo.

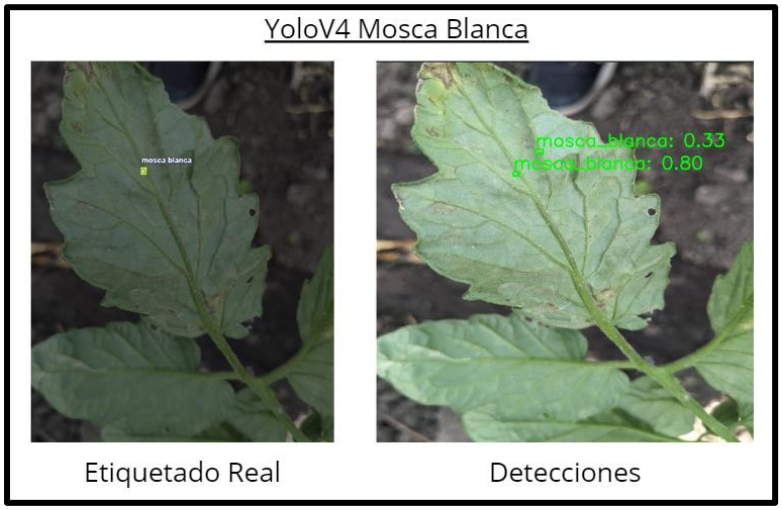


Figura 32 — Detección de mosca blanca usando YoloV4

El Anexo 05 contiene todas las imágenes evaluadas por el modelo para su consulta.

b) Estadístico

Matriz de confusión

La evaluación del modelo YOLOv4 para la detección de la plaga "Mosca Blanca" se realizó utilizando un conjunto de pruebas compuesto por 50 imágenes. La matriz de confusión detalla cómo el modelo clasificó las instancias reales y predichas:

- **True Positives (TP):** Representan las instancias en las que el modelo identificó correctamente la presencia de Mosca Blanca. En este caso, hubo 24 instancias correctamente detectadas.
- **False Positives (FP):** Son los casos en los que el modelo predijo erróneamente la presencia de Mosca Blanca cuando en realidad no había ninguna. En total, se identificaron 2 instancias de este tipo.
- **False Negatives (FN):** Indican las instancias en las que el modelo no logró detectar la Mosca Blanca cuando está realmente estaba presente. En este análisis, no hubo casos de este tipo (FN = 0).
- **True Negatives (TN):** En el caso de este análisis, los True Negatives hacen referencia a las imágenes en las que el modelo predijo correctamente la ausencia de Mosca Blanca. Se registraron 31 imágenes donde no se detectó Mosca Blanca y, efectivamente, no había ninguna instancia real presente.

Tabla 6 — Matriz de confusión para la mosca blanca con Yolo v4

		Valores reales	
Valores predichos		**Positivo**	**Negativo**
	Positivo	True Positive: 24	False Positive: 3
	Negativo	False Negative: 0	True Negative: 31

Estadística de predicciones

Nos basamos en las fórmulas:

- $\boldsymbol{Precision} = \frac{TP}{TP + FP}$
- $\boldsymbol{Recall} = \frac{TP}{TP + FN}$
- $\boldsymbol{F - Value} = 2 * \frac{Precision * Recall}{Precision + Recall}$

Los valores obtenidos para cada métrica se presentan en la Tabla 7.

Tabla 7 — Cuadro estadístico para la mosca blanca con Yolo v4

Estadística para la "Mosca Blanca"		
Precision	**Recall**	**F1-value**
0,89	1,0	0,94

c) Análisis de estadística:

- **Precision (Precisión):** De cada 100 instancias que el modelo identificó como "Mosca Blanca", 89 eran realmente correctas. Es decir, cuando el modelo dice que ha encontrado una Mosca Blanca, en un 89% de los casos, es verdad.
- **Exhaustividad (Recall):** El modelo identificó correctamente todas las Moscas Blancas que había en los datos. En otras palabras, no dejó

pasar ninguna. Su capacidad para encontrar todas las instancias relevantes es perfecta.

- **Valor F (F1-value):** Esta métrica combina la precisión y la exhaustividad en un solo número. Un valor de 0,94 indica que el modelo tiene un equilibrio muy bueno entre la precisión y la exhaustividad. Es decir, es muy eficaz tanto en encontrar todas las instancias correctas como en minimizar los errores.

En resumen, el modelo YOLOv4 para la detección de la Mosca Blanca demuestra una alta precisión y recall, siendo eficiente para identificar todas las instancias relevantes con pocos errores.

B) Minador de las hojas (Liriomyza Trifolii)

a) Especificación de datos

La verificación de los resultados obtenidos en la detección de la plaga "Minador de las hojas" se realizó utilizando un conjunto de pruebas que representa el 4.31% de las imágenes recolectadas, lo que equivale a un total de 50 fotografías.

En la Tabla N° 8, se presenta un desglose detallado de la evaluación del modelo YOLOv4 para esta plaga, con una estructura que permite interpretar claramente los resultados obtenidos. A continuación, se describe cada columna de la tabla:

- **Columna "Código (COD)":** Esta columna contiene un identificador único asignado a cada imagen del conjunto de pruebas, utilizando un formato de referencia que comienza con "IMG" seguido de un número secuencial (por ejemplo, IMG01, IMG02, IMG03, etc.).
- **Columna "Etiquetas Reales":** En esta columna se muestra el conteo exacto de instancias de la plaga "Minador de las hojas" presentes en cada imagen, determinado mediante un proceso manual de etiquetado.
- **Columna "Detecciones"**: Esta columna indica la cantidad de instancias de la plaga detectadas por el modelo YOLOv4 en cada imagen evaluada.
- **Columna "Comentarios"**: Proporciona una descripción

cualitativa del resultado de la detección, clasificando las predicciones del modelo en una de las siguientes categorías:

- **Detección completa**: Cuando el número de instancias detectadas por el modelo coincide exactamente con el conteo real.
- **Detección adicional (n de más):** Cuando el modelo detecta más instancias de las que existen en realidad (falsos positivos), siendo "n" la cantidad de instancias adicionales.
- **Detección incompleta (faltan n)**: Cuando el modelo detecta menos instancias de las que existen en realidad (falsos negativos), siendo "n" la cantidad de instancias faltantes.
- **Sin detección (correcto):** Cuando tanto la etiqueta real como la predicción del modelo son "0", es decir, no se detectó ninguna instancia y esto es correcto.

Tabla 8 — Detalle de etiquetas reales y predicciones de minador de las hojas con Yolo v4

Yolo v4			
Minador de hojas			
Código	**Etiquetas Reales**	**Detecciones**	**Comentarios**
IMG01	0	0	Sin detección (correcto)
IMG02	0	0	Sin detección (correcto)
IMG03	4	4	Detección completa
IMG04	0	0	Sin detección (correcto)
IMG05	0	1	Detección adicional (1 de más)
IMG06	3	3	Detección completa
IMG07	0	0	Sin detección (correcto)
IMG08	1	1	Detección completa
IMG09	2	3	Detección adicional (1 de más)
IMG10	3	3	Detección completa
IMG11	2	3	Detección adicional (1 de más)
IMG12	1	1	Detección completa
IMG13	2	2	Detección completa
IMG14	4	4	Detección completa
IMG15	1	1	Detección completa
IMG16	0	0	Sin detección (correcto)

IMG17	0	0	Sin detección (correcto)
IMG18	0	0	Sin detección (correcto)
IMG19	2	3	Detección adicional (1 de más)
IMG20	3	1	Detección incompleta (faltan 2)
IMG21	1	1	Detección completa
IMG22	3	1	Detección incompleta (faltan 2)
IMG23	0	0	Sin detección (correcto)
IMG24	0	0	Sin detección (correcto)
IMG25	2	1	Detección incompleta (falta 1)
IMG26	2	2	Detección completa
IMG27	3	4	Detección adicional (1 de más)
IMG28	0	0	Sin detección (correcto)
IMG29	1	1	Detección completa
IMG30	2	2	Detección completa
IMG31	0	0	Sin detección (correcto)
IMG32	4	5	Detección adicional (1 de más)
IMG33	1	3	Detección adicional (2 de más)
IMG34	2	4	Detección adicional (2 de más)
IMG35	1	1	Detección completa
IMG36	0	1	Detección adicional (1 de más)
IMG37	0	0	Sin detección (correcto)
IMG38	0	0	Sin detección (correcto)
IMG39	3	3	Detección completa
IMG40	2	3	Detección adicional (1 de más)
IMG41	2	1	Detección incompleta (falta 1)
IMG42	0	0	Sin detección (correcto)
IMG43	0	0	Sin detección (correcto)
IMG44	0	0	Sin detección (correcto)
IMG45	0	0	Sin detección (correcto)
IMG46	0	0	Sin detección (correcto)
IMG47	1	1	Detección completa
IMG48	0	0	Sin detección (correcto)
IMG49	0	0	Sin detección (correcto)
IMG50	2	1	Detección incompleta (falta 1)
Total	60	63	

- **Ejemplo Explicativo**

  Consideremos la fila correspondiente a "IMG06" como ejemplo:

  - **Código (COD):** IMG06
  - **Etiquetas Reales**: El valor es "3", indicando que hay tres instancias de "Minador de hojas" presente en la imagen, según el etiquetado manual.
  - **Detecciones**: El modelo detectó "3" instancias, lo que

significa que identificó correctamente todas las instancias.

- **Comentarios**: "Detección completa" indica que el modelo YOLOv4 identificó correctamente todas las instancias presentes en la imagen.

Figura 33 — Detección de minador de las hojas utilizando YoloV4

El Anexo 05 contiene todas las imágenes evaluadas por el modelo para su consulta.

b) Estadístico

Matriz de confusión

La evaluación del modelo YOLOv4 para la detección de la plaga "Minador de las hojas " se realizó utilizando un conjunto de pruebas compuesto por 50 imágenes. La matriz de confusión detalla cómo el modelo clasificó las instancias reales y predichas:

- **True Positives (TP):** El modelo identificó correctamente 58 casos de Minador de las Hojas. Esto representa las instancias en las que el modelo predijo correctamente la presencia de la plaga.
- **False Positives (FP):** El modelo detectó incorrectamente 12 casos como Minador de las Hojas cuando en realidad no lo eran. Estos son errores donde el modelo predijo la presencia de la plaga cuando no estaba presente.
- **False Negatives (FN):** Hubo 7 casos de Minador de las Hojas que

el modelo no logró identificar, lo que indica que algunas instancias reales de la plaga no fueron detectadas.

- **True Negatives (TN):** El modelo identificó correctamente 22 casos como no afectados por Minador de las Hojas, lo que significa que predijo la ausencia de la plaga cuando realmente no estaba presente.

Tabla 9 — Matriz de confusión para minador de las hojas con Yolo v4

		Valores reales	
Valores predichos		**Positivo**	**Negativo**
	Positivo	True Positive: 58	False Positive: 12
	Negativo	False Negative: 7	True Negative: 22

Estadística de predicciones

Nos basamos en las fórmulas:

- $\boldsymbol{Precision} = \frac{TP}{TP + FP}$
- $\boldsymbol{Recall} = \frac{TP}{TP + FN}$
- $\boldsymbol{F - Value} = 2 * \frac{Precision * Recall}{Precision + Recall}$

Los valores obtenidos para cada métrica se presentan en la Tabla 10.

Tabla 10 — Cuadro estadístico para minador de las hojas con Yolo v4

Estadística para "Minador de las hojas"		
Precision	**Recall**	**F1-value**
0,83	0,89	0,86

c) **Análisis de estadística:**

- **Precisión (Precision):** De cada 100 veces que el modelo identifica una hoja como afectada por el Minador, 83 veces es correcto. Es decir, cuando el modelo dice que hay un Minador en una hoja, hay una alta probabilidad de que esté en lo cierto.
- **Exhaustividad (Recall):** El modelo ha encontrado el 89% de todas las hojas realmente afectadas por el Minador. Esto significa que la mayoría de las hojas que deberían haber sido identificadas fueron detectadas por el modelo.
- **Valor F (F1-value):** Este valor combina precisión y exhaustividad en un solo número. Un 0,86 indica que el modelo tiene un excelente equilibrio entre encontrar todas las hojas afectadas y evitar errores al identificarlas.

En resumen, El modelo para detectar el Minador de las Hojas es efectivo, con una alta precisión para identificar correctamente las hojas afectadas y un buen rendimiento general en su detección.

5.2.3.2 Contrastación del objetivo 2:

Comprobar la eficiencia de clasificación del modelo generado con Faster R-CNN en la detección de plagas en las hojas de los tomates.

A) La mosca blanca (Bemisia Tabaci)

a) Especificación de datos

La verificación de los resultados obtenidos en la detección de la plaga "Mosca blanca" se realizó utilizando un conjunto de pruebas que representa el 4.31% de las imágenes recolectadas, lo que equivale a un total de 50 fotografías.

En la Tabla N° 11, se presenta un desglose detallado de la evaluación del modelo Faster R-CNN para esta plaga, con una estructura que permite interpretar claramente los resultados obtenidos. A continuación, se describe cada columna de la tabla:

- **Columna "Código (COD)":** Esta columna contiene un identificador único asignado a cada imagen del conjunto de

pruebas, utilizando un formato de referencia que comienza con "IMG" seguido de un número secuencial (por ejemplo, IMG01, IMG02, IMG03, etc.).

- **Columna "Etiquetas Reales":** En esta columna se muestra el conteo exacto de instancias de la plaga "Mosca blanca" presentes en cada imagen, determinado mediante un proceso manual de etiquetado.
- **Columna "Detecciones"**: Esta columna indica la cantidad de instancias de la plaga detectadas por el modelo Faster R-CNN en cada imagen evaluada.
- **Columna "Comentarios"**: Proporciona una descripción cualitativa del resultado de la detección, clasificando las predicciones del modelo en una de las siguientes categorías:
    - **Detección completa**: Cuando el número de instancias detectadas por el modelo coincide exactamente con el conteo real.
    - **Detección adicional (n de más):** Cuando el modelo detecta más instancias de las que existen en realidad (falsos positivos), siendo "n" la cantidad de instancias adicionales.
    - **Detección incompleta (faltan n)**: Cuando el modelo detecta menos instancias de las que existen en realidad (falsos negativos), siendo "n" la cantidad de instancias faltantes.
    - **Sin detección (correcto):** Cuando tanto la etiqueta real como la predicción del modelo son "0", es decir, no se detectó ninguna instancia y esto es correcto.

Tabla 11 — Detalle de etiquetas reales y predicciones de la mosca blanca con Faster R-CNN

Faster R-CNN			
Mosca Blanca			
Código	**Etiquetas Reales**	**Detecciones**	**Comentarios**
IMG01	1	1	Detección completa
IMG02	1	2	Detección adicional (1 de más)

IMG03	0	0	Sin detección (correcto)
IMG04	1	1	Detección completa
IMG05	0	0	Sin detección (correcto)
IMG06	0	0	Sin detección (correcto)
IMG07	1	1	Detección completa
IMG08	0	0	Sin detección (correcto)
IMG09	0	0	Sin detección (correcto)
IMG10	0	0	Sin detección (correcto)
IMG11	0	0	Sin detección (correcto)
IMG12	0	0	Sin detección (correcto)
IMG13	0	0	Sin detección (correcto)
IMG14	0	0	Sin detección (correcto)
IMG15	0	0	Sin detección (correcto)
IMG16	1	1	Detección completa
IMG17	1	1	Detección completa
IMG18	1	1	Detección completa
IMG19	0	0	Sin detección (correcto)
IMG20	0	0	Sin detección (correcto)
IMG21	0	0	Sin detección (correcto)
IMG22	0	0	Sin detección (correcto)
IMG23	1	1	Detección completa
IMG24	1	1	Detección completa
IMG25	0	0	Sin detección (correcto)
IMG26	0	0	Sin detección (correcto)
IMG27	0	0	Sin detección (correcto)
IMG28	1	1	Detección completa
IMG29	0	0	Sin detección (correcto)
IMG30	0	0	Sin detección (correcto)
IMG31	2	2	Detección completa
IMG32	0	0	Sin detección (correcto)
IMG33	0	0	Sin detección (correcto)
IMG34	0	0	Sin detección (correcto)
IMG35	0	0	Sin detección (correcto)
IMG36	0	0	Sin detección (correcto)
IMG37	1	1	Detección completa
IMG38	1	1	Detección completa
IMG39	0	0	Sin detección (correcto)
IMG40	0	0	Sin detección (correcto)
IMG41	0	0	Sin detección (correcto)
IMG42	1	1	Detección completa
IMG43	2	2	Detección completa
IMG44	2	2	Detección completa
IMG45	2	2	Detección completa
IMG46	2	2	Detección completa
IMG47	0	0	Sin detección (correcto)

IMG48	1	1	Detección completa
IMG49	0	1	Detección adicional (1 de más)
IMG50	0	0	Sin detección (correcto)
Total	24	25	

- **Ejemplo Explicativo**

  Consideremos la fila correspondiente a "IMG48" como ejemplo:

  - **Código (COD):** IMG48
  - **Etiquetas Reales**: El valor es "1", indicando que hay una instancia de "Mosca blanca" presente en la imagen, según el etiquetado manual.
  - **Detecciones**: El modelo detectó "1" instancias, lo que significa que identificó correctamente la instancia presente.
  - **Comentarios**: "Detección completa" indica que el modelo Faster R-CNN identificó correctamente las instancias presentes en la imagen.

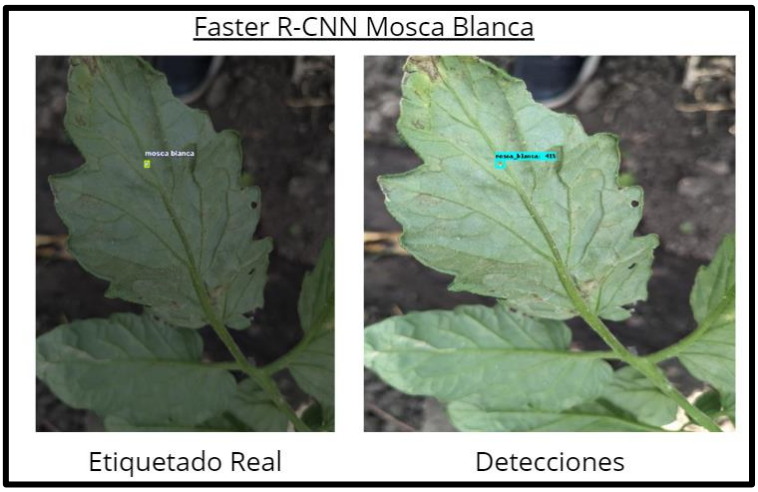


Figura 34 — Detección de mosca blanca utilizando Faster R-CNN

El Anexo 05 contiene todas las imágenes evaluadas por el modelo para su consulta.

b) Estadístico

Matriz de confusión

La evaluación del modelo Faster R-CNN para la detección de la plaga "Mosca Blanca" se realizó utilizando un conjunto de pruebas compuesto por 50 imágenes. La matriz de confusión detalla cómo el modelo clasificó las instancias reales y predichas:

- **True Positives (TP):** El modelo identificó correctamente 24 casos de Mosca Blanca.
- **False Positives (FP):** El modelo identificó incorrectamente 2 caso como Mosca Blanca cuando en realidad no lo era.
- **False Negatives (FN):** No hubo casos de Mosca Blanca que el modelo no lograra identificar, indicando que todos los casos reales fueron detectados.
- **True Negatives (TN):** En el caso de este análisis, los True Negatives hacen referencia a las imágenes en las que el modelo predijo correctamente la ausencia de Minador de las hojas. Se registraron 31 imágenes donde no se detectó Mosca Blanca y, efectivamente, no había ninguna instancia real presente.

Tabla 12 — Matriz de confusión para mosca blanca con Faster R-CNN

		Valores reales	
Valores predichos		**Positivo**	**Negativo**
	Positivo	True Positive: 24	False Positive: 2
	Negativo	False Negative: 0	True Negative: 31

c) Estadística de predicciones

Nos basamos en las fórmulas:

- $\boldsymbol{Precision} = \frac{TP}{TP + FP}$
- $\boldsymbol{Recall} = \frac{TP}{TP + FN}$
- $\boldsymbol{F - Value} = 2 * \frac{Precision * Recall}{Precision + Recall}$

Los valores obtenidos para cada métrica se presentan en la tabla 13.

Tabla 13 — Cuadro estadístico para mosca blanca con Faster R-CNN

Estadística para "Mosca blanca"		
Precision	**Recall**	**F1-value**
0.92	1.0	0.96

d) Análisis de estadística:

- **Precisión (Precision):** De cada 100 veces que el modelo identifica una hoja como afectada por la Mosca Blanca, 92 veces es correcto. Esto significa que cuando el modelo dice que ha encontrado una Mosca Blanca, hay una alta probabilidad de que esté en lo cierto.
- **Exhaustividad (Recall):** El modelo ha encontrado el 100% de todas las hojas realmente afectadas por la Mosca Blanca. No ha pasado por alto ninguna hoja que debería haber identificado. Esto muestra que el modelo es excelente para detectar todas las instancias de Mosca Blanca.
- **Valor F (F1-value):** Este número combina precisión y exhaustividad en un solo valor. Un 0,96 indica que el modelo tiene un equilibrio muy bueno entre encontrar todas las hojas afectadas y evitar errores al identificarlas. Es un valor muy alto, lo que significa que el modelo es muy eficaz en general.

En resumen, el modelo muestra un excelente rendimiento con alta precisión, exhaustividad y exactitud, destacando su capacidad para identificar correctamente las instancias positivas sin cometer muchos errores.

B) Minador de las hojas (Liriomyza Trifolii)

a) Especificación de datos

La verificación de los resultados obtenidos en la detección de la plaga "Minador de las hojas" se realizó utilizando un conjunto de pruebas que representa el 4.31% de las imágenes recolectadas, lo que equivale a un total de 50 fotografías.

En la Tabla N° 14, se presenta un desglose detallado de la evaluación del modelo Faster R-CNN para esta plaga, con una estructura que permite

interpretar claramente los resultados obtenidos. A continuación, se describe cada columna de la tabla:

- **Columna "Código (COD)":** Esta columna contiene un identificador único asignado a cada imagen del conjunto de pruebas, utilizando un formato de referencia que comienza con "IMG" seguido de un número secuencial (por ejemplo, IMG01, IMG02, IMG03, etc.).
- **Columna "Etiquetas Reales":** En esta columna se muestra el conteo exacto de instancias de la plaga "Minador de las hojas" presentes en cada imagen, determinado mediante un proceso manual de etiquetado.
- **Columna "Detecciones"**: Esta columna indica la cantidad de instancias de la plaga detectadas por el modelo Faster R-CNN en cada imagen evaluada.
- **Columna "Comentarios"**: Proporciona una descripción cualitativa del resultado de la detección, clasificando las predicciones del modelo en una de las siguientes categorías:
    - **Detección completa**: Cuando el número de instancias detectadas por el modelo coincide exactamente con el conteo real.
    - **Detección adicional (n de más)**: Cuando el modelo detecta más instancias de las que existen en realidad (falsos positivos), siendo "n" la cantidad de instancias adicionales.
    - **Detección incompleta (faltan n)**: Cuando el modelo detecta menos instancias de las que existen en realidad (falsos negativos), siendo "n" la cantidad de instancias faltantes.
    - **Sin detección (correcto)**: Cuando tanto la etiqueta real como la predicción del modelo son "0", es decir, no se detectó ninguna instancia y esto es correcto.
    - **Sin detección**: Cuando el modelo no detecto ninguna instancia, pero en realidad si existían instancias en la

imagen (falsos negativos).

Tabla 14 — Detalle de etiquetas reales y predicciones de minador de las hojas con Faster R-CNN

Faster R-CNN			
Minador de hojas			
Código	**Etiquetas Reales**	**Detecciones**	**Comentarios**
IMG01	0	0	Sin detección (correcto)
IMG02	0	0	Sin detección (correcto)
IMG03	4	4	Detección completa
IMG04	0	0	Sin detección (correcto)
IMG05	0	1	Detección adicional (1 de más)
IMG06	3	3	Detección completa
IMG07	0	0	Sin detección (correcto)
IMG08	1	1	Detección completa
IMG09	2	3	Detección adicional (1 de más)
IMG10	3	3	Detección completa
IMG11	2	2	Detección completa
IMG12	1	1	Detección completa
IMG13	2	0	Sin detección
IMG14	4	4	Detección completa
IMG15	1	2	Detección adicional (1 de más)
IMG16	0	0	Sin detección (correcto)
IMG17	0	0	Sin detección (correcto)
IMG18	0	0	Sin detección (correcto)
IMG19	2	3	Detección adicional (1 de más)
IMG20	3	1	Detección incompleta (faltan 2)
IMG21	1	1	Detección completa
IMG22	3	4	Detección adicional (1 de más)
IMG23	0	0	Sin detección (correcto)
IMG24	0	0	Sin detección (correcto)
IMG25	2	2	Detección completa
IMG26	2	2	Detección completa
IMG27	3	4	Detección adicional (1 de más)
IMG28	0	0	Sin detección (correcto)
IMG29	1	1	Detección completa
IMG30	2	2	Detección completa
IMG31	0	0	Sin detección (correcto)
IMG32	4	2	Detección incompleta (faltan 2)
IMG33	1	2	Detección adicional (1 de más)
IMG34	2	3	Detección adicional (1 de más)
IMG35	1	1	Detección completa
IMG36	0	0	Sin detección (correcto)

IMG37	0	0	Sin detección (correcto)
IMG38	0	0	Sin detección (correcto)
IMG39	3	3	Detección completa
IMG40	2	5	Detección adicional (3 de más)
IMG41	2	1	Detección incompleta (falta 1)
IMG42	0	0	Sin detección (correcto)
IMG43	0	0	Sin detección (correcto)
IMG44	0	0	Sin detección (correcto)
IMG45	0	0	Sin detección (correcto)
IMG46	0	0	Sin detección (correcto)
IMG47	1	1	Detección completa
IMG48	0	0	Sin detección (correcto)
IMG49	0	0	Sin detección (correcto)
IMG50	2	0	Sin detección
Total	60	61	

- **Ejemplo explicativo**

  Consideremos la fila correspondiente a "IMG06" como ejemplo:

  - **Código (COD):** IMG06
  - **Etiquetas Reales**: El valor es "3", indicando que hay tres instancias de "Minador de hojas" presente en la imagen, según el etiquetado manual.
  - **Detecciones**: El modelo detectó "3" instancias, lo que significa que identificó correctamente todas las instancias.
  - **Comentarios**: "Detección completa" indica que el modelo Faster R-CNN identificó correctamente todas las instancias presentes en la imagen.

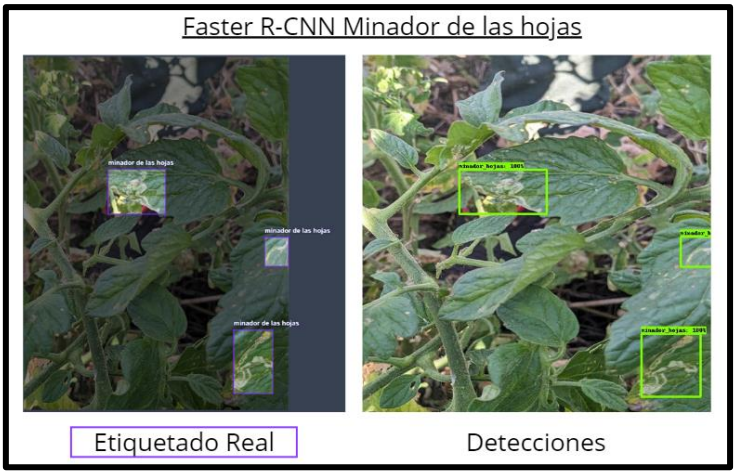


Figura 35 — Detección de minador de las hojas utilizando Faster R-CNN

El Anexo 05 contiene todas las imágenes evaluadas por el modelo para su consulta.

b) Estadístico

Matriz de confusión

La evaluación del modelo Faster R-CNN para la detección de la plaga "Minador de las hojas" se realizó utilizando un conjunto de pruebas compuesto por 50 imágenes. La matriz de confusión detalla cómo el modelo clasificó las instancias reales y predichas:

- **True Positives (TP):** El modelo identificó correctamente 47 casos de Minador de las Hojas. Esto representa las instancias en las que el modelo predijo correctamente la presencia de la plaga.
- **False Positives (FP):** El modelo detectó incorrectamente 11 casos como Minador de las Hojas cuando en realidad no lo eran. Estos son errores donde el modelo predijo la presencia de la plaga cuando no estaba presente.
- **False Negatives (FN):** Hubo 9 casos de Minador de las Hojas que el modelo no logró identificar, lo que indica que algunas instancias reales de la plaga no fueron detectadas.
- **True Negatives (TN):** El modelo identificó correctamente 22 casos como no afectados por Minador de las Hojas, lo que significa que predijo la ausencia de la plaga cuando realmente no estaba presente.

Tabla 15 — Matriz de confusión para minador de las hojas con Faster R-CNN

		Valores reales	
Valores predichos		**Positivo**	**Negativo**
	Positivo	True Positive: 47	False Positive: 11
	Negativo	False Negative: 9	True Negative: 22

Estadística de predicciones

Nos basamos en las fórmulas:

- $\boldsymbol{Precision} = \frac{TP}{TP + FP}$
- $\boldsymbol{Recall} = \frac{TP}{TP + FN}$
- $\boldsymbol{F - Value} = 2 * \frac{Precision * Recall}{Precision + Recall}$

Los valores obtenidos para cada métrica se presentan en la tabla 16.

Tabla 16 — Cuadro estadístico para minador de las hojas con Faster R-CNN

Estadística para "Minador de las hojas"		
Precision	**Recall**	**F1-value**
0,81	0,84	0,83

c) Análisis de estadística:

- **Precisión (Precision):** De cada 100 veces que el modelo identifica una hoja como afectada por el Minador, 81 veces está en lo cierto. Esto significa que cuando el modelo predice la presencia de la plaga, hay una buena probabilidad de que la predicción sea precisa.
- **Exhaustividad (Recall):** El modelo ha detectado el 84% de todas las hojas que realmente estaban afectadas por el Minador. mostrando un buen desempeño en encontrar instancias relevantes, pero aún con margen de mejora.
- **Valor F (F1-value):** Este valor combina la precisión y la exhaustividad en una sola cifra. Un 0,83 indica que el modelo muestra un equilibrio razonable entre precisión y exhaustividad.

En resumen, el modelo para el Minador de las hojas demostró ser eficaz en la detección de la plaga, con un buen balance entre la precisión y la capacidad de identificar las instancias relevantes. Los resultados reflejan un alto nivel de aciertos y un desempeño estable, resaltando su utilidad en la detección automática de la plaga objetivo.

5.2.3.3 Contrastación del objetivo general:

Comprobar la técnica de Machine Learning más eficiente para detectar las plagas en las hojas de los tomates en los cultivos de Abancay Apurímac, 2022.

a) **Estadística especificación de datos**

Con fundamento en los objetivos específicos, en la tabla 17 y tabla 18 procederemos a realizar la comparación del rendimiento de ambos modelos con respecto a cada una de las plagas, ya sea Mosca blanca o Minador de las hojas. Este análisis nos permitirá evaluar la eficiencia de Yolo v4 y Faster R-CNN en la detección y clasificación de las plagas mencionadas, brindando así una visión detallada de cómo cada modelo aborda de manera específica la tarea en cuestión.

b) Estadístico

A) La mosca blanca (Bemisia Tabaci)

Tabla 17 — Estadística de comparación de modelos para la mosca blanca

Algoritmo o Modelo Machine Learning	Precision	Recall	F1-value
Yolo v4	0,89	1,0	0,94
Faster R-CNN	0,92	1,0	0,96

Para una visualización más clara y comprensible de los resultados obtenidos, se recomienda consultar la figura 36, el cual ilustra de manera comparativa los valores de precisión (precision), exhaustividad (recall) y valor F (F1-value) de los modelos evaluados. Este enfoque visual facilita la identificación de las diferencias clave en el rendimiento entre los modelos Yolo v4 y Faster R-CNN en la detección de la plaga "Mosca Blanca".

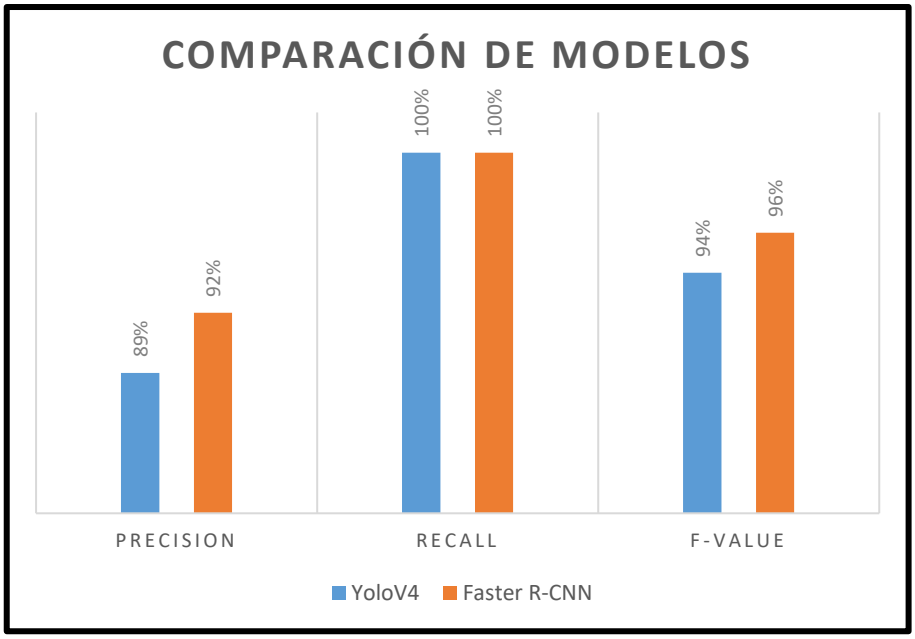


Figura 36 — Comparación de modelos para la mosca blanca

Análisis de estadística comparativa

- **YOLOv4**: El modelo presenta un F1-Value de 0.94, con una precisión del 89% y un recall del 100%. Esto muestra que YOLOv4 es capaz de identificar todas las instancias de Mosca Blanca, aunque tiene una menor precisión en comparación con Faster R-CNN.
- **Faster R-CNN**: Con un F1- Value de 0.96, una precisión del 92% y un recall perfecto del 100%, este modelo muestra un rendimiento superior en la detección de Mosca Blanca.

B) Minador de las hojas (Liriomyza Trifolii)

Tabla 18 — Estadística de comparación de modelos para minador de las hojas

Algoritmo o Modelo Machine Learning	Precision	Recall	F1-value
Yolo v4	0,83	0,89	0,86
Faster R-CNN	0,81	0,84	0,83

En la siguiente figura 37 se presenta una visualización comparativa de los valores de precisión (precision), exhaustividad (recall) y valor F (F1-

value) de los modelos evaluados. Esta figura ofrece una perspectiva clara y detallada, permitiendo identificar de manera sencilla las diferencias clave en el rendimiento de los modelos YOLOv4 y Faster R-CNN en la detección de la plaga "Minador de las Hojas".

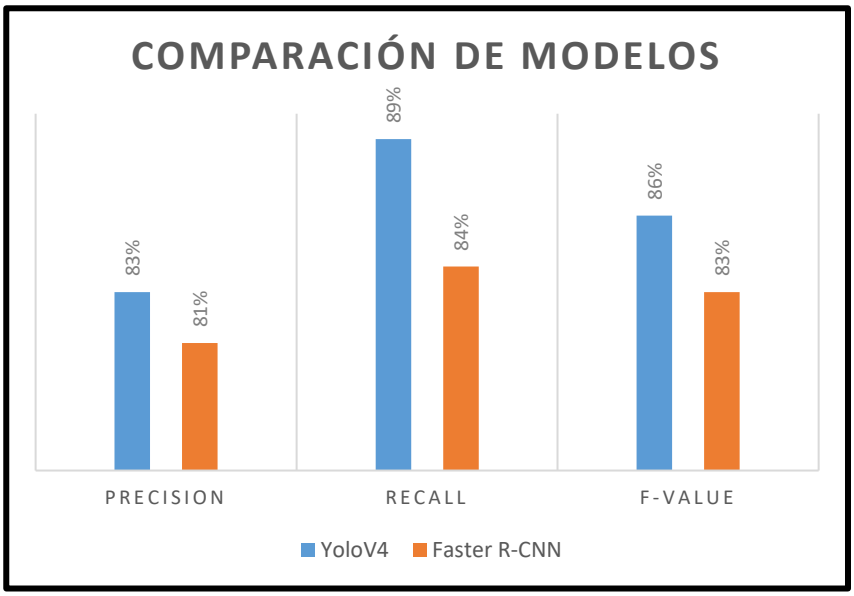


Figura 37 — Comparación de modelos para minador de las hojas

Análisis de estadística comparativa

- **YOLOv4**: Muestra un F1-Value de 0.86, con una precisión del 83% y un recall del 89%, lo que indica un mejor desempeño que Faster R-CNN en la detección de esta plaga.
- **Faster R-CNN**: Con un F1-Value de 0.83, precisión del 81% y recall del 84%, tiene un rendimiento inferior al de YOLOv4 para esta plaga.

c) Cálculo de la media armónica del F1-Value

La media armónica de los F1-Value para cada modelo se calcula utilizando la siguiente fórmula:

$$\boldsymbol{Media\ Armónica} = \frac{n}{\sum_{i=1}^{n} \frac{1}{x^i}}$$

donde x_i son los F1-Value de cada clase.

- **Cálculo para YoloV4**:
  - $F1 - Value_{Mosca\ blanca} = 0.94$
  - $F1 - Value_{Minador\ de\ las\ hojas} = 0.86$
  - $Media\ Armónica = 0.90$

- **Cálculo para Faster R-CNN:**
  - $F1 - Value_{Mosca\ blanca} = 0.96$
  - $F1 - Value_{Minador\ de\ las\ hojas} = 0.83$
  - $Media\ Armónica = 0.89$

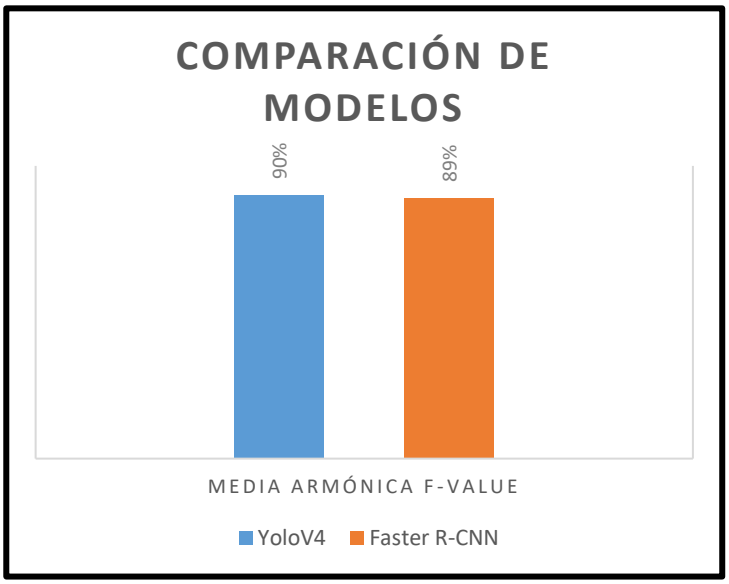


Figura 38 — Comparación de media armónica F-Value entre YoloV4 y Faster R-CNN

El análisis global de los modelos mediante la media armónica del F1-Value ofrece una visión integral de su rendimiento al considerar ambas clases. YOLOv4 obtiene una media armónica de 0.90, mientras que Faster R-CNN alcanza una media armónica de 0.89. Este resultado sugiere que YOLOv4 tiene un mejor desempeño global al detectar las plagas de Mosca Blanca y Minador de las Hojas.

En conclusión, con base en los resultados obtenidos y el análisis global mediante la media armónica del F1-Value, se concluye que YOLOv4 es el modelo más eficiente para la detección de plagas en las hojas de tomate en los cultivos de Abancay, Apurímac, 2022. Si bien Faster R-CNN muestra un rendimiento superior en la detección de la Mosca

Blanca, el desempeño global de YOLOv4, evidenciado por su media armónica de 0.90, lo posiciona como el modelo preferido para la tarea general de detección de plagas. Esto se debe a su mejor equilibrio en la identificación de ambas clases de plagas, lo que lo hace adecuado para aplicaciones que requieren alta eficiencia y precisión en múltiples clases de detección. Aunque Faster R-CNN sigue siendo una opción destacada en contextos donde la precisión absoluta es prioritaria, YOLOv4 ofrece un balance más robusto para un rendimiento generalizado.

5.2 Discusión de resultados

Por medio de los resultados obtenidos con respecto al objetivo general, "Comprobar la técnica de Machine Learning más eficiente para detectar plagas en las hojas de los tomates en los cultivos de Abancay, Apurímac, 2022", se concluye que YOLOv4 ofrece un desempeño global más eficiente al considerar la detección de ambas plagas: Mosca Blanca y Minador de las Hojas. Esto se evidencia a través de la media armónica del F1-Value, con un valor de 0.90 para YOLOv4 frente a 0.89 para Faster R-CNN, reflejando un mejor balance general en la eficacia para la detección precisa de las plagas analizadas.

Para la Mosca Blanca, Faster R-CNN obtiene un F1-Value de 0.96 en comparación con el 0.94 de YOLOv4, lo que demuestra una eficacia ligeramente superior al lograr un mejor balance entre precisión (precision) y exhaustividad (recall), minimizando falsos positivos y negativos. Sin embargo, en la detección del Minador de Hojas, YOLOv4 sobresale con un F1-Value de 0.86, superando al valor de 0.83 alcanzado por Faster R-CNN. Esto sugiere que YOLOv4 es más eficaz para la detección de esta plaga específica, manteniendo un buen equilibrio entre la identificación de verdaderos positivos y la reducción de falsos negativos.

Este resultado se encuentra en consonancia con las conclusiones de CÓRDOVA PÉREZ, Claudia Sofía (2021), quien usó modelos de detección preentrenados, específicamente Faster R-CNN y YOLOv4, para la detección de tres tipos de plagas de insectos. En su investigación, Faster R-CNN logró una precisión del 94.06%, mientras que YOLOv4 obtuvo un 95.82%, lo que muestra que ambos modelos ofrecen un rendimiento aceptable, con YOLOv4 destacándose por su eficacia en aplicaciones donde se necesita un balance entre precisión y velocidad de detección.

De manera similar, la investigación de GUERRERO ANDRADE Carlos Jonathan y MARTÍNEZ MOSQUERA Silvia Diana (2022) respalda la eficacia de los modelos YOLO, mostrando su viabilidad en la detección rápida y precisa de lesiones necróticas causadas por la plaga thrips en guisantes, con una precisión de 81.60% y una efectividad del 86%. Aunque el contexto sea diferente, el desempeño de YOLOv4 evidencia su consistencia en la detección eficiente de plagas.

Además, los hallazgos de HASAN Ulutaş y VEYSEL Aslantaş (2023), destacan el uso de modelos de redes neuronales convolucionales (CNN) para clasificar hasta nueve enfermedades en hojas de tomate, alcanzando una precisión del 99.60% mediante arquitecturas de conjunto. Este respaldo subraya la capacidad de modelos como YOLOv4 para abordar problemas de clasificación complejos de manera eficiente.

En conclusión, aunque Faster R-CNN presenta un rendimiento destacado en la detección de la Mosca Blanca, YOLOv4 logra una mayor eficacia general al considerar ambas plagas, con una media armónica superior del F1-Value. Esto lo posiciona como el modelo preferido para aplicaciones que demandan un equilibrio sólido entre precisión y eficacia general en la detección de múltiples clases de plagas.

CAPÍTULO VI
CONCLUSIONES Y RECOMENDACIONES

6.1. Conclusiones

En este estudio de investigación, se ha determinado que tanto YOLOv4 como Faster R-CNN presentan ventajas específicas en la detección de plagas en hojas de tomate; sin embargo, el análisis global realizado mediante la media armónica del F1-Value arroja a YOLOv4 como el modelo más eficaz en términos generales.

- YOLOv4 obtuvo una media armónica del F1-Value de 0.90, superando a Faster R-CNN, que alcanzó una media armónica de 0.89. Esto refleja un mejor equilibrio global en la detección de las dos clases de plagas estudiadas (Mosca Blanca y Minador de Hojas). Para la detección de la "Mosca Blanca", YOLOv4 mostró una precisión (precision) del 89%, una exhaustividad (recall) del 100% y un valor F (F1-value) del 94%, mientras que para el "Minador de Hojas" alcanzó una precisión del 83%, una exhaustividad del 89% y un F1-value de 86%. Esto evidencia que YOLOv4 es capaz de ofrecer una detección robusta y equilibrada para múltiples clases, haciéndolo adecuado para aplicaciones en las que se requiera una buena combinación de precisión, velocidad y capacidad de detección global.
- Por otro lado, Faster R-CNN se destacó por su rendimiento superior en la detección de la "Mosca Blanca", alcanzando un F1-Value de 0.96, con una precisión del 92% y una exhaustividad del 100%. Sin embargo, su rendimiento disminuyó en la detección del "Minador de Hojas", con un F1-Value de 0.83. Aunque mostró un desempeño sobresaliente en la detección precisa de una clase, su rendimiento general fue inferior a YOLOv4 en términos del balance global de métricas. Esto lo posiciona como una opción adecuada para escenarios en los que la precisión absoluta en una clase específica sea prioritaria, pero no logra igualar el equilibrio general mostrado por YOLOv4.

La conclusión de este análisis, respaldada por el uso de la media armónica del F1-Value como criterio integral, muestra que YOLOv4 es el modelo más eficinete para la detección de plagas en hojas de tomate en los cultivos de Abancay, Apurímac, 2022, debido a su

capacidad para mantener un equilibrio entre precisión y exhaustividad en múltiples clases. Su desempeño general lo posiciona como la opción preferida cuando se busca una solución robusta, eficiente y aplicable a contextos variados. No obstante, Faster R-CNN sigue siendo una alternativa valiosa en casos donde la precisión para una clase específica es esencial y los errores deben minimizarse al máximo.

6.2. Recomendaciones

Se recomienda utilizar YOLOv4 como modelo principal en aplicaciones donde se requiere un equilibrio global de precisión y exhaustividad para la detección de múltiples clases de plagas. Su mejor desempeño global, demostrado a través de la media armónica del F1-Value, lo hace ideal para situaciones en las que es necesario un balance entre velocidad y eficacia general. Es especialmente adecuado para contextos en los que la detección rápida y confiable de múltiples plagas es una prioridad.

Para aplicaciones donde la precisión absoluta y la minimización de errores en una clase específica (como la Mosca Blanca) son esenciales, se recomienda considerar el uso de Faster R-CNN. Este modelo demostró un rendimiento sobresaliente en la detección precisa de una clase, lo que lo convierte en una opción valiosa en escenarios críticos donde los falsos positivos y falsos negativos deben ser minimizados.

Se debe tener en cuenta que Faster R-CNN tiende a requerir más recursos computacionales y un tiempo de inferencia mayor en comparación con YOLOv4. Por lo tanto, es importante planificar adecuadamente la infraestructura computacional necesaria para su implementación.

Se recomienda continuar optimizando ambos modelos mediante el uso de conjuntos de datos más amplios y representativos, así como explorar técnicas de mejora y ajuste fino de parámetros para maximizar la precisión y la exhaustividad. Además, se sugiere evaluar el impacto de nuevos enfoques y arquitecturas en la mejora del rendimiento global, especialmente en la detección de plagas en entornos diversos y con distribuciones desbalanceadas.

Finalmente, es importante evaluar y actualizar continuamente las estrategias de implementación y monitoreo de ambos modelos en función de los cambios en las

condiciones de cultivo, nuevas plagas emergentes o variaciones climáticas. Esto asegurará que la precisión y la eficacia de los modelos de detección se mantengan a largo plazo.

REFERENCIAS BIBLIOGRÁFICAS

ACOSTA, Mª Belén. 2022. Enfermedades del tomate - Guía completa con FOTOS. ecologiaverde.com. 22 de octubre de 2019.

ALPAYDIN, E. 2014. Introduction to Machine Learning. 3.ª ed. Massachusetts, Estados Unidos de América: MIT Press, 2014. ISBN 9780262028189.

AMERSHI, Saleema; BEGEL, Andrew; BIRRELL, Emre K.; FORD, Danielle; GAY, George; KAHNG, Andrea; NAGAPPAN, Nachiappan; NUSHI, Besmira; ZIMMER, Kurt. 2019. Software engineering for machine learning: a case study. En: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP). IEEE, 2019. ISBN 9781728117607.

BENGIO, Yoshua. 2012. Practical recommendations for gradient-based training of deep architectures. En: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 437–478. ISBN 9783642352881.

BOCHKOVSKIY, Alexey; WANG, Chien-Yao; LIAO, Hong-Yuan Mark. 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection. 2020.

CALVO, Javier; GUZMÁN, Manuel A.; RAMOS, Daniel. 2018. Machine Learning, una pieza clave en la transformación de los modelos de negocio. Madrid: Management Solutions, 2018.

CÁMARA DE COMERCIO DE MADRID. 2019. Machine Learning y su impacto en el ámbito empresarial. 30 de diciembre de 2019.

CAMPOS Y COVARRUBIAS, Guillermo; LULE MARTÍNEZ, Nallely Emma. 2013. La observación, un método para el estudio de la realidad. Xihmai. 2013, 7(13). ISSN 1870-6703.

CENTENO FRANCO, Alba. 2019. Deep Learning. Trabajo Fin de Grado. Universidad de Sevilla, Facultad de Matemáticas, 2019.

CHOLLET, Francois. 2017. Deep learning with python. Manning Publications Co. LLC, 2017. ISBN 9781638352044.

CLOUDFACTORY. 2024. Faster R-CNN. 2024.

CÓRDOVA PÉREZ, Claudia Sofía. 2021. Aplicación de aprendizaje profundo para la detección y clasificación automática de insectos agrícolas en trampas pegantes: una revisión de literatura. Tesis de bachiller, Pontificia Universidad Católica del Perú, 2021.

CORRALES, Daniel Camilo. 2015. Hacia la detección de plagas y enfermedades en cultivos a través de aprendizaje supervisado. Ingeniería y Universidad. 2015, 19(1), 207-228.

CUÉLLAR, María Elena; MORALES, Francisco J. 2006. La mosca blanca Bemisia tabaci (Gennadius) como plaga y vectora de virus en fríjol común (Phaseolus vulgaris L.). Revista Colombiana de Entomología. 2006, 32(1), 1–9. ISSN 2665-4385.

DATACAMP. 2024. Explicación de la detección de objetos YOLO: Guía para principiantes. En: Blog de DataCamp. 29 de enero de 2024.

ERAZO VALDIVIEZO, Jorge Deninson; UEMA HIGA, Ken Sebastian. 2020. Redes neuronales para la identificación de enfermedades en el cultivo de tomate (Solanum lycopersicum L.) en Honduras. Proyecto Especial de Graduación. Escuela Agrícola Panamericana, Zamorano, 2020.

ESPINO TIMÓN, Carlos. 2017. Análisis predictivo: Técnicas y modelos utilizados y aplicaciones del mismo - herramientas Open Source que permiten su uso. Trabajo de Fin de Grado. Universitat Oberta de Catalunya, 2017.

GALÁN ZAPATA, Jefferson Luis. 2019. Sistema inteligente de reconocimiento de imágenes para apoyar el diagnóstico de plagas y enfermedades en el cultivo de arroz en el departamento de Lambayeque en el año 2019. Tesis de bachiller, Universidad Católica Santo Toribio de Mogrovejo, 2021.

GARCÍA AMARO, Ernesto. 2022. Uso de técnicas de visión artificial para la identificación de daños foliares causados por plagas y enfermedades en plantas de jitomate. Tesis de doctorado, Universidad Autónoma del Estado de México, 2022.

GARCÍA-PALACIOS, Daniel; BAUTISTA MARTÍNEZ, Néstor; VALDEZ CARRASCO, Jorge Manuel. 2014. Identificación de minadores (diptera: agomyzidae) asociados con hortalizas, y sus parasitoides. ACTA ZOOLÓGICA MEXICANA (N.S.). 2014, 30(1). ISSN 2448-8445.

GÉRON, Aurélien. Hands-On. 2022. Machine Learning with Scikit-Learn, Keras, and TensorFlow. 2.ª ed. O'Reilly Media, Incorporated, 2022. ISBN 9781098122461.

GIL GAVELA, Sergio. 2020. Desarrollo de aplicación basada en CNN para algoritmos de visión en coches autónomos. Trabajo de Fin de Grado, Universidad Complutense de Madrid, Facultad de Informática, 2020.

GIL RUBIO, Ricardo; CRUZ PEREZ, Edwin Andrés; PERDOMO CHARRY, Oscar. 2022. Modelos de machine learning para clasificar la cartera en un fondo de pensiones. Trabajo de grado, Universidad Santo Tomás, 2022.

GONZÁLEZ, Ligdi. 2024. Diferencia entre aprendizaje supervisado y no supervisado. Aprende IA. 15 de junio de 2018.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. 2016. Deep Learning. Cambridge: MIT Press, 2016. ISBN 9780262035613.

GOOGLE COLAB. 2024. Te damos la bienvenida a Colaboratory. 2024

GUERRERO ANDRADE, Carlos Jonathan; MARTÍNEZ MOSQUERA, Silvia Diana. 2022.

Reconocimiento de lesiones necróticas para la detección de la plaga thrips en el guisante mediante el uso del modelo deep learning YOLOv4-tiny. Latin-American Journal of Computing (LAJC). 2022, 9(1), 46–59. ISSN 1390-9134.

HERNÁNDEZ SAMPIERI, Roberto; FERNÁNDEZ COLLADO, Carlos; BAPTISTA LUCIO, Pilar. 2014. Metodología de la investigación. 6.ª ed. México: McGraw-Hill, 2014. ISBN 9781456223960.

HERNANSAEZ MEORO, Pedro; PASTOR MANZANO, José. 1956. El Tomate, su cultivo y sus enfermedades. Anales de la Universidad de Murcia (Ciencias). 1956, 15(3-4), 477–494. ISSN 0210-4721.

HINESTROZA RAMÍREZ, Denniye. 2018. El Machine Learning a través de los tiempos, y los aportes a la humanidad. Trabajo de grado de pregrado, Universidad Libre, 2018.

INSTITUTO NACIONAL DE ESTADÍSTICA E INFORMÁTICA (INEI). 2021. Perú: Panorama Económico Departamental N° 5: Mayo 2021. Lima: INEI, 2021.

INTAGRI S.C. 2024. Estrategias de Control de Minadores en Tomate. 2017.

KOHAVI, Ron. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence. San Francisco: Morgan Kaufmann Publishers Inc., 1995, pp. 1137–1143. ISBN 9781558603639.

KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. 2012. Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems 25 (NIPS 2012). 2012, 1097–1105.

LÓPEZ BRIEGA, R.E. 2018. *Introducción al Machine Learning*. 2018.

LÓPEZ PORTOCARRERO, César Augusto. 2019. Aplicación de imágenes hiperespectrales para la detección temprana de roya amarilla (Hemileya vastatrix) en café (Coffea arábica), en el Distrito de Limbamba, Provincia Rodríguez de Mendoza, Región Amazonas. Tesis de maestría, Universidad Nacional Toribio Rodríguez de Mendoza de Amazonas, 2019.

MARZAL VARÓ, Andrés; GRACIA LUENGO, Isabel; GARCÍA SEVILLA, Pedro. 2014. Introducción a la programación con Python 3. Castelló de la Plana: Universitat Jaume I, 2014. ISBN 9788469711781.

MORENO MAYHUIRE, Joel Saul. 2020. Eficiencia de un sistema de control de calidad mediante procesamiento digital de imágenes en la clasificación de la tunta en la planta de producción de Kishuara - Andahuaylas. Tesis de pregrado, Universidad Nacional José María Arguedas, 2020.

NEYRA HAU YON, Jorge Luis. 2021. Determinación en tiempo real de presencia de cadmio

en cultivo de cacao aplicando Machine Learning. Tesis de maestría, Universidad de Piura, 2021.

OLIVEROS SABOGAL, David Santiago. 2019. Algoritmo para la medición del grado de severidad de tizón temprano causado por *Alternaria solani* en hojas de tomate, a partir del análisis digital de imágenes. Tesis de pregrado, Universidad El Bosque, 2019.

OPPENHEIM, Dor. ROYALTY, Robert N. NEMEH, Michelle K. SMART, Christine D. 2019, Using deep learning for image-based potato tuber disease detection. Phytopathology. 2019, 109(6), 1083–1087. ISSN 1943-7684.

ORACLE CORPORATION. 2020. Lifecycle of machine learning models. 2020. pp. 12.

PEREYRA, María Eugenia. 2020. Detección de enfermedades y plagas en cultivos mediante Machine Learning. Tesis de grado, Universidad Nacional de La Plata, 2020.

PRIETO MORANTE, Juan Sebastián; TRELLES PRIETO, Rita Luciana. 2021. Clasificación de hojas de tomate con plagas o enfermedades usando una máquina de soporte vectorial (SVM). Tesis de bachiller, Universidad de Piura, 2021.

REDAGRICOLA. 2018. Reinventar el cultivo del tomate. 2018.

REDMON, Joseph; DIVVALA, Santosh; GIRSHICK, Ross; FARHADI, Ali. 2016. You only look once: unified, real-time object detection. En: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2016. ISBN 9781467388511.

REN, Shaoqing; HE, Kaiming; GIRSHICK, Ross; SUN, Jian. 2017. Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2017, 39(6), 1137–1149. ISSN 2160-9292.

ROBOFLOW. 2023. Getting Started with Roboflow. Roboflow Blog. 16 de marzo de 2023.

ROMERO-LOPEZ, Adria; GIRO-I-NIETO, Xavier; BURDICK, Jack; MARQUES, Oge. Skin. 2017. Lesion Classification from Dermoscopic Images Using Deep Learning Techniques. En: Biomedical Engineering. Calgary, AB, Canada: ACTAPRESS, 2017. ISBN 9780889869905.

ROZADA RANEROS, Saúl. 2021. Estudio de la arquitectura YOLO para la detección de objetos mediante deep learning. Tesis de maestría, Universidad de Valladolid, 2021.

RUIZ, J.Z., 2018. Comparativa y análisis de algoritmos de aprendizaje automático para la predicción del tipo predominante de cubierta arbórea. . Madrid, España.

RUDER, Sebastian. 2016. An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747. 2016.

RUSSELL, Rudolph. 2018. MACHINE LEARNING: Guía Paso a Paso Para Implementar Algoritmos De Machine Learning Con Python. CreateSpace Independent Publishing Platform, 2018. ISBN 9781720933687.

RUSSELL, Stuart J.; NORVIG, Peter. 2004. Inteligencia Artificial: Un Enfoque Moderno. 2ª

ed. Madrid: Pearson Educación, 2004. ISBN 9788420540030.

SANDOVAL SERRANO, Lilian Judith. 2018. Algoritmos de aprendizaje automático para el análisis y predicción de datos. Revista Tecnológica. 2018, 11(1), 36–40.

SASAKI, Yutaka. 2007. The Truth of the F-Measure. Informe técnico. School of Computer Science, University of Manchester, 2007.

SENASICA. 2024. Mosquita blanca. 2020.

SIERRA GARCÍA, Víctor Yair. 2021. Algoritmo de detección de mosca blanca por medio de inteligencia artificial. Trabajo de grado, Universidad Nacional Abierta y a Distancia (UNAD), 2021.

SIERRA GUZMÁN, Viviana Yineth. 2021. Algoritmo de detección de mosca blanca por medio de inteligencia artificial en las hojas de plátano. Trabajo de grado, Universidad Nacional Abierta y a Distancia (UNAD), 2021.

SIMEONE, Osvaldo. 2018. A very brief introduction to machine learning with applications to communication systems. IEEE Transactions on Cognitive Communications and Networking. 2018, 4(4), 648–664. ISSN 2372-2045.

SOKOLOVA, Marina; LAPALME, Guy. 2009. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*. 2009, 45(4), 427–437. ISSN 0306-4573.

SZELISKI, Richard. 2022. Computer Vision: Algorithms and Applications. 2ª ed. Cham: Springer, 2022. ISBN 9783030343743.

ULUTAŞ, Hasan; ASLANTAŞ, Veysel. 2023. Design of efficient methods for the detection of tomato leaf disease utilizing proposed ensemble CNN model. Electronics. 2023, 12(4), 827. ISSN 2079-9292.

ZAMORANO, Juan. 2018. Comparativa y análisis de algoritmos de aprendizaje automático para la predicción del tipo predominante de cubierta arbórea. Tesis de maestría, Universidad Complutense de Madrid, 2018.

ANEXOS

Anexo 01 — Fotografías de mosca blanca y minador de hojas con Yolo v4

Figura 39 — Fotografías de mosca blanca con Yolo v4

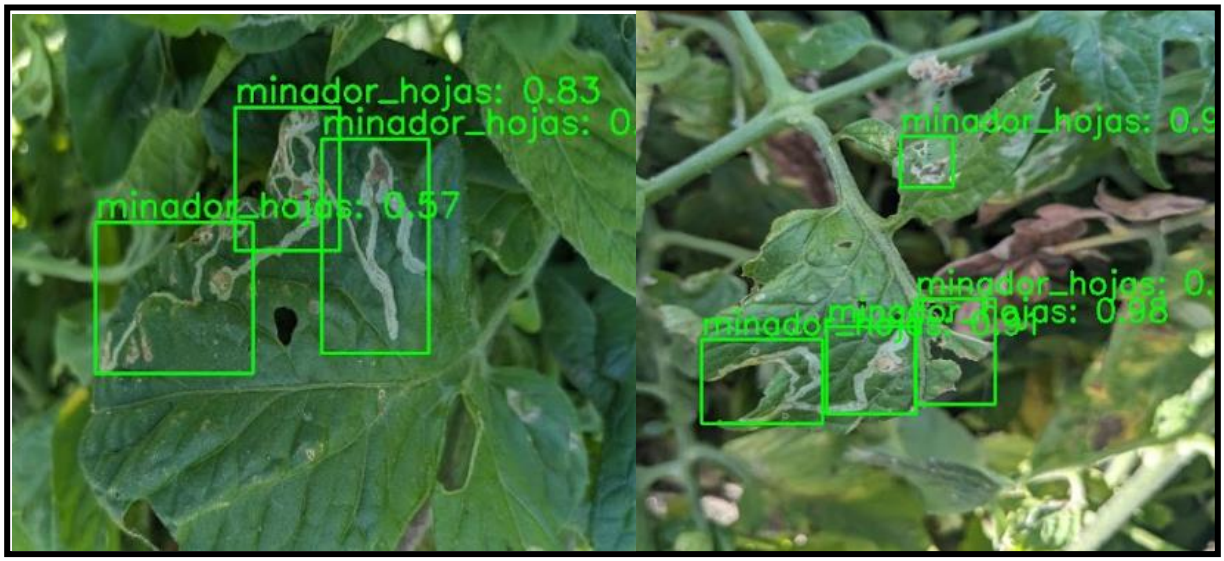


Figura 40 — Fotografías de minador de hojas con Yolo v4

Anexo 02 — Fotografías de mosca blanca y minador de hojas con Faster R-CNN

Figura 41 — Fotografías de mosca blanca con Faster R-CNN

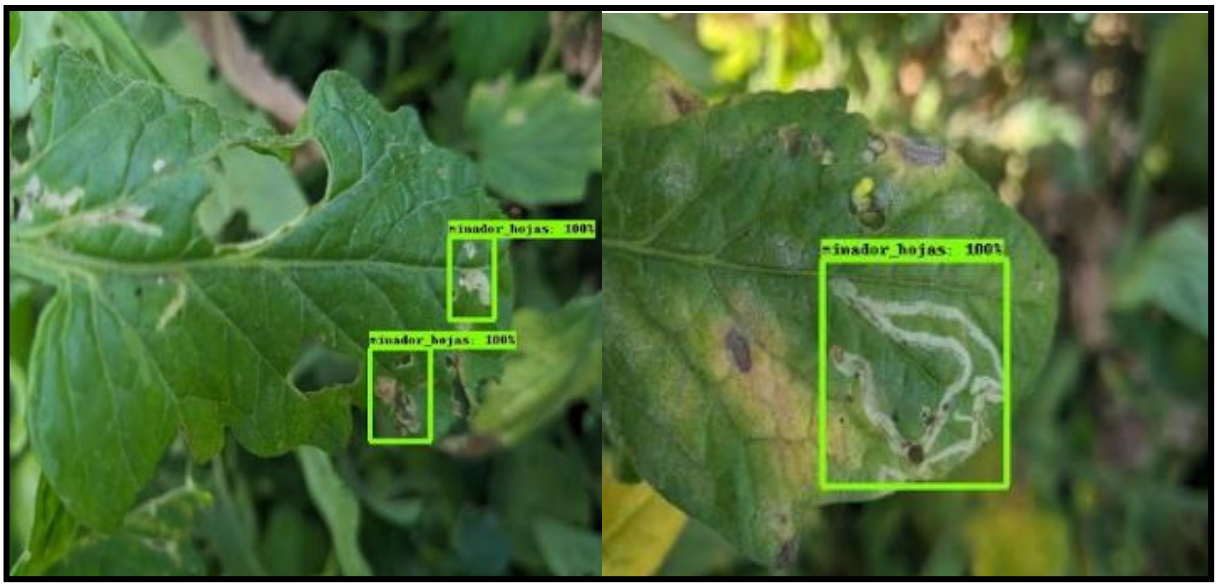


Figura 42 — Fotografías de minador de hojas con Faster R-CNN

Anexo 03 — Ficha de observación para entrenamiento de modelos.

Tabla 19 — Ficha de observación para entrenamiento de modelos.

Ficha de Observación			
Estado:	Entrenamiento		
Modelos:	- Yolo v4 - Faster R-CNN		
Cantidad de imágenes:	1047 fotografías		
Validación de Datos:	Las fotografías fueron previamente etiquetadas con el tipo de plaga correspondiente para facilitar el aprendizaje supervisado de ambos modelos.		
categoría	imagen_jpg	label	cantidad_etiquetas
train	PXL_20230618_125200102_jpg.rf.52424360e602842ff8efaeac2d4cd494.jpg	**minador_hojas**	1
train	PXL_20230730_133545823_jpg.rf.55e701e985af01cc132c6f27e2999896.jpg	**mosca_blanca**	1
train	PXL_20230618_124936663_jpg.rf.c5255d21bbe8ef9bc7c3647ff1153f64.jpg	**minador_hojas**	3
train	PXL_20230730_132724087_jpg.rf.7fc5b9b546f088b941f540e5e977d49e.jpg	**mosca_blanca**	1
train	PXL_20230730_134834879_jpg.rf.3d56d43afaf1d02181259962a93c465e.jpg	**mosca_blanca**	1
train	PXL_20230730_132618672_jpg.rf.600caa60031d7665329546e86cb11c18.jpg	**mosca_blanca**	2
train	PXL_20230730_133403291_jpg.rf.67ee00a4cb276138eef3007fa85306a7.jpg	**mosca_blanca**	1
train	PXL_20230618_125229091_jpg.rf.1f25e04d79a1cf5826c5ba3b549da425.jpg	**minador_hojas**	2
train	PXL_20230730_133541009-MP_jpg.rf.5a01faf9e6df254895c9f36c14ad7460.jpg	**mosca_blanca**	1
train	PXL_20230618_124711663_jpg.rf.c7ddbf683df5efe5f69a32b3e4ed88e7.jpg	**minador_hojas**	2
train	PXL_20230618_130055516_jpg.rf.f7415ffce78aba91ef7e5d3a12c45c2f.jpg	**minador_hojas**	1
train	PXL_20230730_132905998_jpg.rf.572580e6bad74c9b103b559fe3e67534.jpg	**mosca_blanca**	1
train	PXL_20230730_133730817_jpg.rf.9c9065470bb68e8b3f17bb864bf5b36e.jpg	**mosca_blanca**	1
train	PXL_20230618_130333385_jpg.rf.2c0647fbcfb5800b79b8e164dad9b9e2.jpg	**minador_hojas**	4
train	PXL_20230618_125003173_jpg.rf.b755a3812264785b220ca3a33e646f9e.jpg	**minador_hojas**	1

train	PXL_20230618_124741114_jpg.rf.196e17e14d356d4edf5385cea90d7531.jpg	**minador_hojas**	5
train	PXL_20230730_132156514-PORTRAIT_jpg.rf.364664a1729c955167f90febcff27689.jpg	**mosca_blanca**	2
train	PXL_20230730_134011400_jpg.rf.f257977486ef9074ae080e2f63efac02.jpg	**mosca_blanca**	1
train	PXL_20230618_130246733_jpg.rf.fbf52cbb8db65ec5facbd3fae5fc577c.jpg	**minador_hojas**	1
train	PXL_20230730_134039601_jpg.rf.47947cef8845943f46dffc3a97739476.jpg	**mosca_blanca**	3
train	PXL_20230618_124859656_jpg.rf.d7ae57c87d2f8520b2e5d1d253c74185.jpg	**minador_hojas**	3
train	PXL_20230618_125154096_jpg.rf.b8b1b1da15bb1246b58ce52f942ad5be.jpg	**minador_hojas**	1
train	PXL_20230730_134837489_jpg.rf.4cc9812aabb559a0190deed6658c942c.jpg	**mosca_blanca**	1
train	PXL_20230730_132700044_jpg.rf.cc8c6fe4168ae3d1a46b049276179f19.jpg	**mosca_blanca**	1
train	PXL_20230730_133651326_jpg.rf.9e693ec34a83260012639827f267fbad.jpg	**mosca_blanca**	2
train	PXL_20230618_125044439-MP_jpg.rf.a247fb29914f0b8dcb19590ee84bdd7b.jpg	**mosca_blanca**	7
train	PXL_20230730_132411362_jpg.rf.05be0942a7ee180bd2570c60abe90fd2.jpg	**minador_hojas**	3
train	PXL_20230618_130550359_jpg.rf.11623426f4e9fb91611d1440d32c5151.jpg	**minador_hojas**	3
train	PXL_20230730_132627265_jpg.rf.75e7935c0004bede56f396a0522c2f1d.jpg	**mosca_blanca**	1
train	PXL_20230618_125046961-MP_jpg.rf.52528f1ece6c911319c8fdb2e9bf2c26.jpg	**mosca_blanca**	1
...	...		
train	PXL_20230730_132354886_jpg.rf.77d6abc49a8ad616c790d8d42dd185dc.jpg	**mosca_blanca**	1

Anexo 04 — Ficha de observación para la prueba de modelos.

Tabla 20 — Ficha de observación para la prueba de modelos.

Ficha de Observación								
Estado:			Pruebas (Testing)					
Modelos:			* Yolo v4 * Faster R-CNN					
Cantidad de imágenes:			50 fotografías					
Validación de Datos:			Las fotografías fueron previamente etiquetadas con el tipo de plaga correspondiente para facilitar el aprendizaje supervisado de ambos modelos.					
Categoria	**Codigo**	**Nombre imagen**	Mosca Blanca			Minador de las hojas		
			Etiquetas reales	Detecciones YoloV4	Detecciones Faster RCNN	Etiquetas reales	Detecciones YoloV4	Detecciones Faster RCNN
test	IMG1	PXL_20230730_132133906-PORTRAIT_jpg.rf.9a6cbed3d96f2256c8d60d3f69032483.jpg	1	1	1	0	0	0
test	IMG2	PXL_20230730_132617885_jpg.rf.5dd74ef5d3d70e626682ff4e45cef2ed.jpg	1	2	2	0	0	0
test	IMG3	PXL_20230618_130510905_jpg.rf.f0641e9c1cace26983a11feb4ac1d0e9.jpg	0	0	0	4	4	4
test	IMG4	PXL_20230730_132853404_jpg.rf.013b753b6599e3907f9c0eb6c77b1b45.jpg	1	1	1	0	0	0
test	IMG5	PXL_20230618_130134755_jpg.rf.a80cb564efa3c6375ca18e16f5e5bb43.jpg	0	0	0	0	1	1
test	IMG6	PXL_20230618_130221462_jpg.rf.c9d477551c107fb2e9855fe2c3100c2f.jpg	0	0	0	3	3	3
test	IMG7	PXL_20230730_132359596_jpg.rf.3b9b48769771ffa2efb920ca68e1501c.jpg	1	1	1	0	0	0
test	IMG8	PXL_20230618_125109439_jpg.rf.678de7ab1f130a668e871c73c203d608.jpg	0	0	0	1	1	1
test	IMG9	PXL_20230618_125113915-MP_jpg.rf.6cdba63b8eb6888aa4717506fcb8dfc0.jpg	0	0	0	2	3	3
test	IMG10	PXL_20230618_130046931_jpg.rf.6b6b2f7b73e6c682e5ac6cce199d6c3a.jpg	0	0	0	3	3	3

test	IMG11	PXL_20230618_124710817_jpg.rf.bd06e6f74f47725bb3603887f86e6ae4.jpg	0	0	0	2	3	2
test	IMG12	PXL_20230618_124956283-MP_jpg.rf.35a20122406a52e7c46a41b883e9a95f.jpg	0	0	0	1	1	1
test	IMG13	PXL_20230618_124705435_jpg.rf.77cbaf3642ee1bdf2e5be14f3f567cca.jpg	0	0	0	2	2	0
test	IMG14	PXL_20230618_130210232_jpg.rf.b274ac91ae13818b6b706ff0d3775dc2.jpg	0	0	0	4	4	4
test	IMG15	PXL_20230618_124934589-MP_jpg.rf.155dffce90ea79e707b379681a4c3a8c.jpg	0	0	0	1	1	2
test	IMG16	PXL_20230730_133301849-MP_jpg.rf.4b7d95a2fa1693855d778e9bd1e249b0.jpg	1	1	1	0	0	0
test	IMG17	PXL_20230618_125739771_jpg.rf.00b48852bb3420dd059241e1e0933e21.jpg	1	1	1	0	0	0
test	IMG18	PXL_20230730_132658391-MP_jpg.rf.428ff7b0eada2a46370c0b5d3fa0317f.jpg	1	1	1	0	0	0
test	IMG19	PXL_20230618_130046188_jpg.rf.9e65bf8bcb0f7a3d86d1920cb7daedeb.jpg	0	0	0	2	3	3
test	IMG20	PXL_20230618_124935215_jpg.rf.2d91a0d070ba03ffc4c97e013f923169.jpg	0	0	0	3	1	1
test	IMG21	PXL_20230618_130242173_jpg.rf.6dc58f230798b93c9fb805956c5b35a6.jpg	0	0	0	1	1	1
test	IMG22	PXL_20230618_124935962_jpg.rf.0ef01096fa880386e23073a88ffe74e7.jpg	0	0	0	3	1	4
test	IMG23	PXL_20230730_132726654_jpg.rf.a2dfc3fe23a00c5df2417d71a72ab32c.jpg	1	1	1	0	0	0
test	IMG24	PXL_20230730_132628035_jpg.rf.01abd5364a0f8adec6fcf5e1b612364d.jpg	1	1	1	0	0	0
test	IMG25	PXL_20230618_130106323_jpg.rf.62f1931ee8ce3f9391e35d668259e4f9.jpg	0	0	0	2	1	2
test	IMG26	PXL_20230618_130450422_jpg.rf.25b2135a0dd1fa15f3d80ac283d92127.jpg	0	0	0	2	2	2
test	IMG27	PXL_20230618_130209132_jpg.rf.01a950a5b9b654e5df3a75e1366eb71b.jpg	0	0	0	3	4	4
test	IMG28	PXL_20230730_132238547-MP_jpg.rf.a78a91f14a00d0a164fbf9ede44150c2.jpg	1	1	1	0	0	0
test	IMG29	PXL_20230618_130348865_jpg.rf.70cdb1a266e1b5c479bc538a710ad319.jpg	0	0	0	1	1	1
test	IMG30	PXL_20230618_130447106_jpg.rf.e4c13ae8f956996bf6660cad8e0936d2.jpg	0	0	0	2	2	2
test	IMG31	PXL_20230730_132205710-PORTRAIT_jpg.rf.bfc1a995682c350f4937415b9235b189.jpg	2	2	2	0	0	0

test	IMG32	PXL_20230618_130551402_jpg.rf.ee901a9a44ca89c8d521bd0ccdb98523.jpg	0	0	0	4	5	2
test	IMG33	PXL_20230618_130521832_jpg.rf.e609b52e131f2a46543fa05286a1b65b.jpg	0	0	0	1	3	2
test	IMG34	PXL_20230618_130336107-MP_jpg.rf.54c48dba086cf0a8b0ae21d4a6898535.jpg	0	0	0	2	4	3
test	IMG35	PXL_20230618_130309228_jpg.rf.fed69cb61dd5847f45cef6d342ed1a4c.jpg	0	0	0	1	1	1
test	IMG36	PXL_20230618_125814929_jpg.rf.e8258a1117f496095b550033d30631e9.jpg	0	0	0	0	1	0
test	IMG37	PXL_20230730_132128154-PORTRAIT_jpg.rf.a19be18f6afc5ee3fa006289ce076e18.jpg	1	1	1	0	0	0
test	IMG38	PXL_20230618_125056064_jpg.rf.cea6907d79e546acc48c1451894f91fd.jpg	1	1	1	0	0	0
test	IMG39	PXL_20230618_125117551-MP_jpg.rf.84e8bd495dcce96802da1f52f66155f6.jpg	0	0	0	3	3	3
test	IMG40	PXL_20230618_130432141_jpg.rf.09b0df5e39194cb1057e1b024820696b.jpg	0	0	0	2	3	5
test	IMG41	PXL_20230618_125108267-MP_jpg.rf.f2fc45a14e6b7cc2a9bbfaf78618d0b8.jpg	0	0	0	2	1	1
test	IMG42	PXL_20230730_132832502_jpg.rf.aa9a3bc5270201bc6e0fc46516ca2598.jpg	1	1	1	0	0	0
test	IMG43	PXL_20230730_132204879-PORTRAIT_jpg.rf.1492a2585ec041143d7e96ac48baa1ba.jpg	2	2	2	0	0	0
test	IMG44	PXL_20230730_132208080-PORTRAIT_jpg.rf.229de635f0bdf56adf2a518f8ca85d69.jpg	2	2	2	0	0	0
test	IMG45	PXL_20230730_132157288-PORTRAIT_jpg.rf.1449004e1dc5f761cd8830787f4ba351.jpg	2	2	2	0	0	0
test	IMG46	PXL_20230618_125040865_jpg.rf.c2dccab9304a4a9329d4adf73c111348.jpg	2	2	2	0	0	0
test	IMG47	PXL_20230618_124700906_jpg.rf.57e807bcf820860450640e76a1f39215.jpg	0	0	0	1	1	1
test	IMG48	PXL_20230618_125053126_jpg.rf.eb6d392b4ef77142cf007be4613670f8.jpg	1	2	1	0	0	0
test	IMG49	PXL_20230730_132246863_jpg.rf.186fb103270afc3ddefd3b6a4a9064e3.jpg	0	1	1	0	0	0
test	IMG50	PXL_20230618_130413850_jpg.rf.d8acc659ecaff188b1f404d2a0c7e4d6.jpg	0	0	0	2	1	0

Anexo 05 — Resultados de las detecciones realizadas por YOLOv4 y Faster R-CNN en las imágenes de prueba

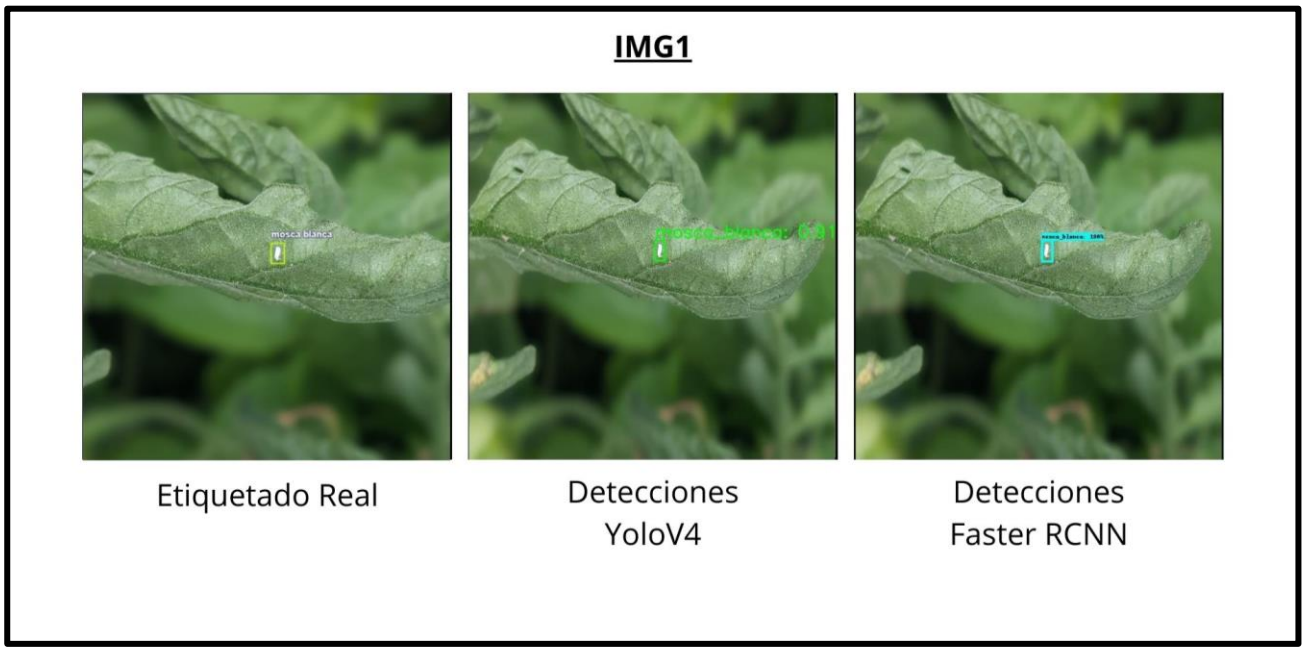


Figura 43 — Resultados de las detecciones IMG1

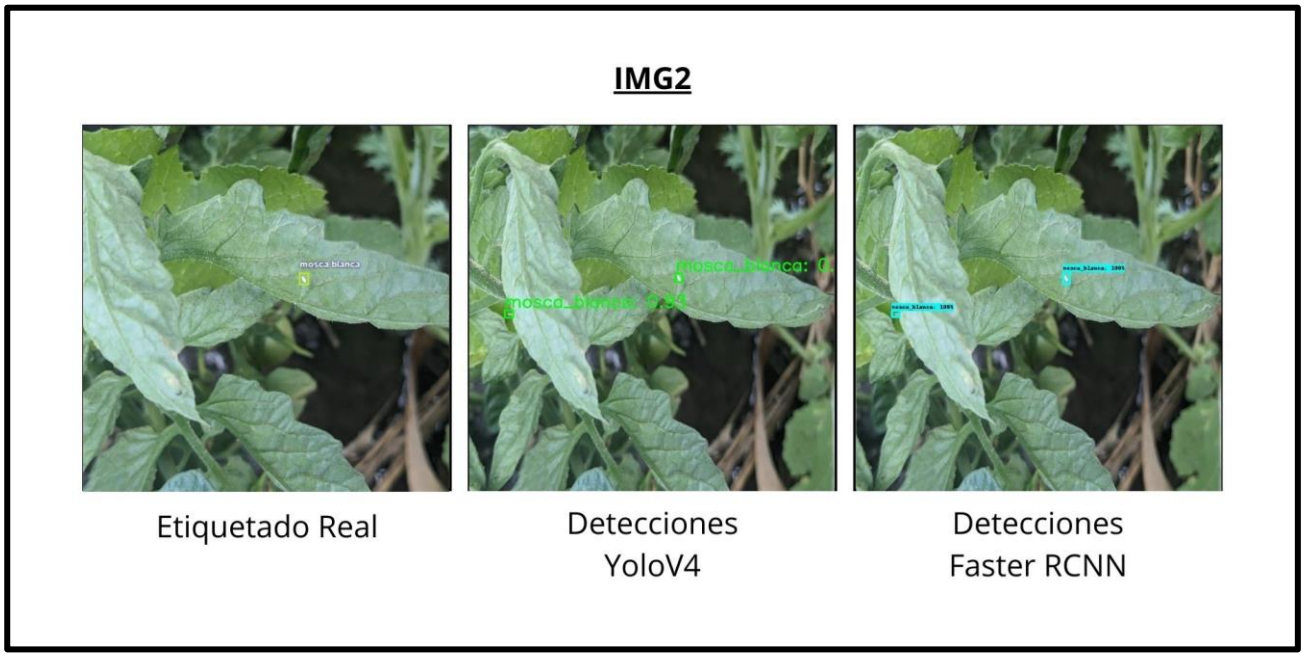


Figura 44 — Resultados de las detecciones IMG2

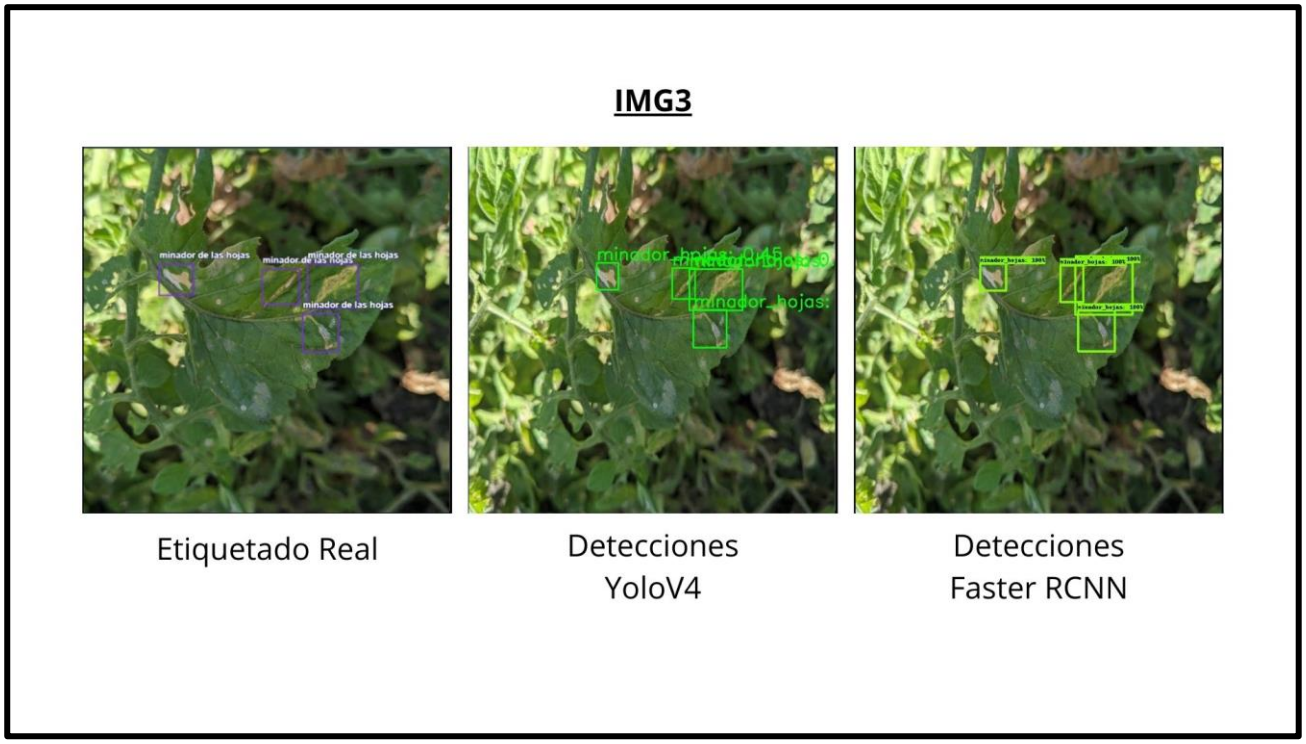


Figura 45 — Resultados de las detecciones IMG3

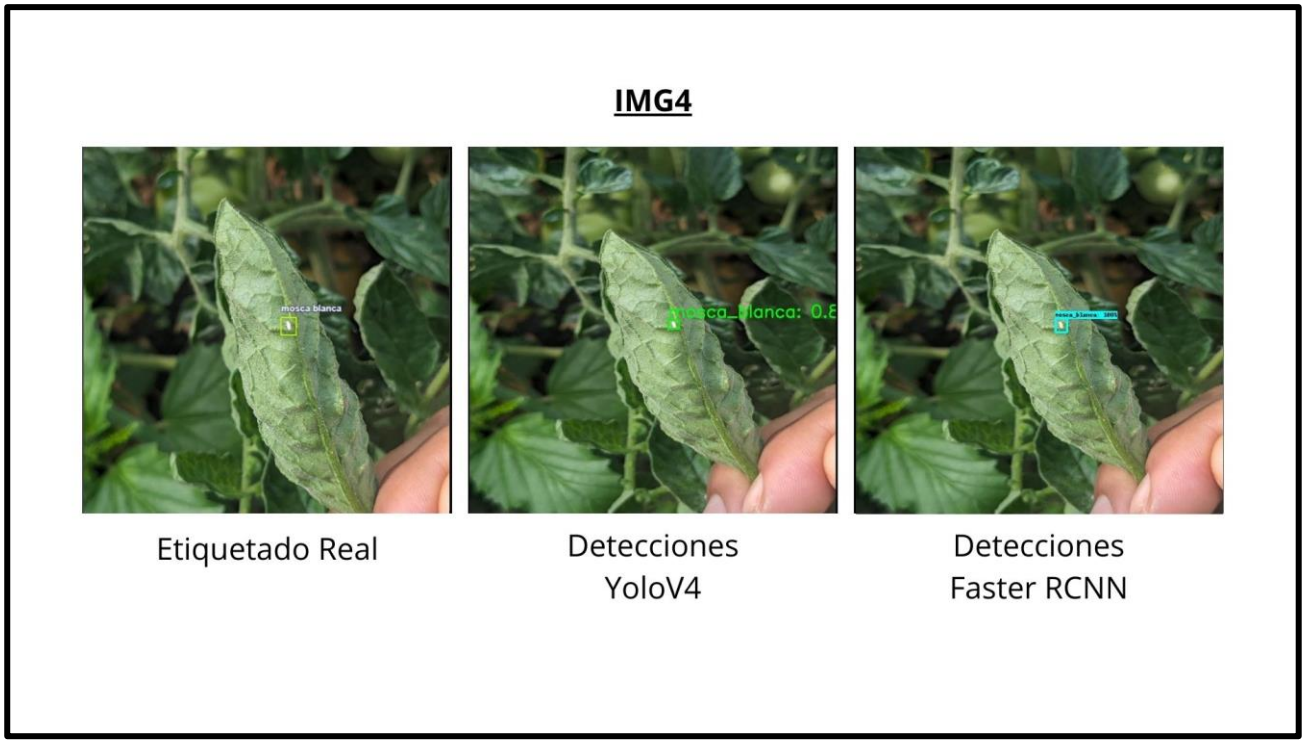


Figura 46— Resultados de las detecciones IMG4

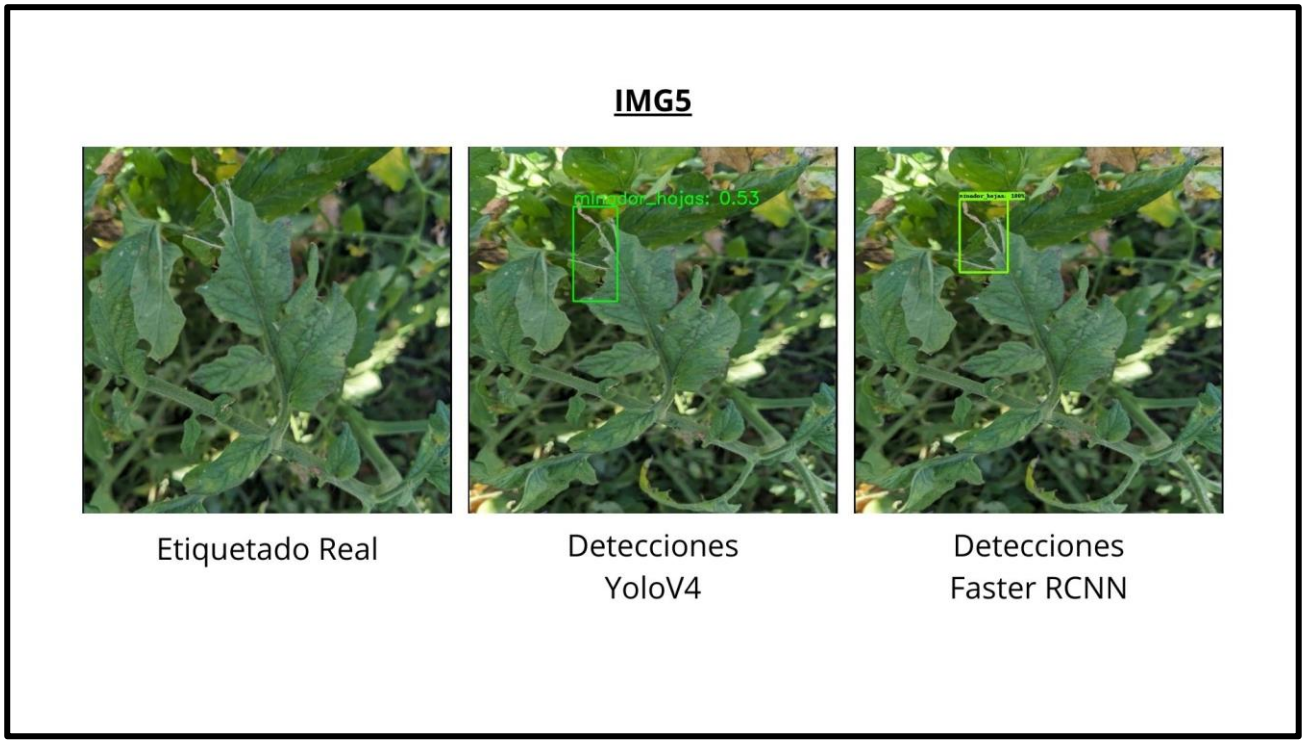


Figura 47 — Resultados de las detecciones IMG5

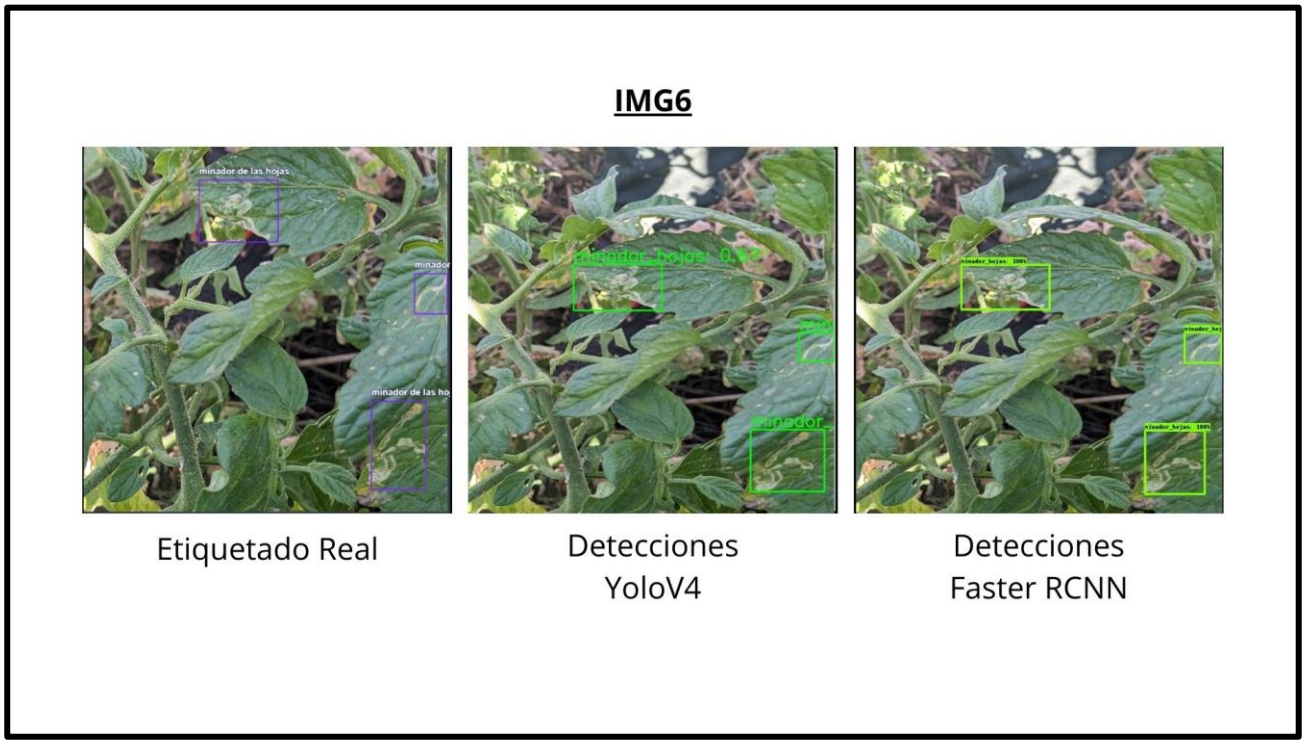


Figura 48— Resultados de las detecciones IMG6

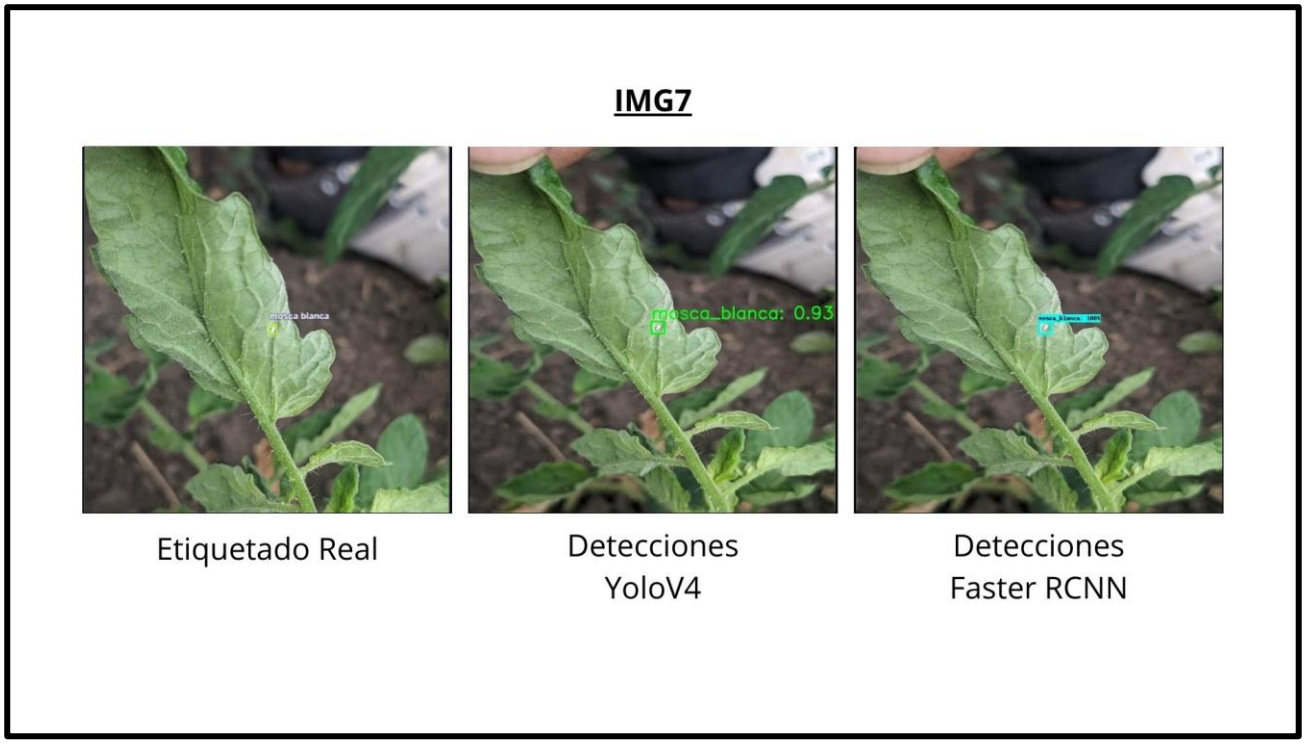


Figura 49 — Resultados de las detecciones IMG7

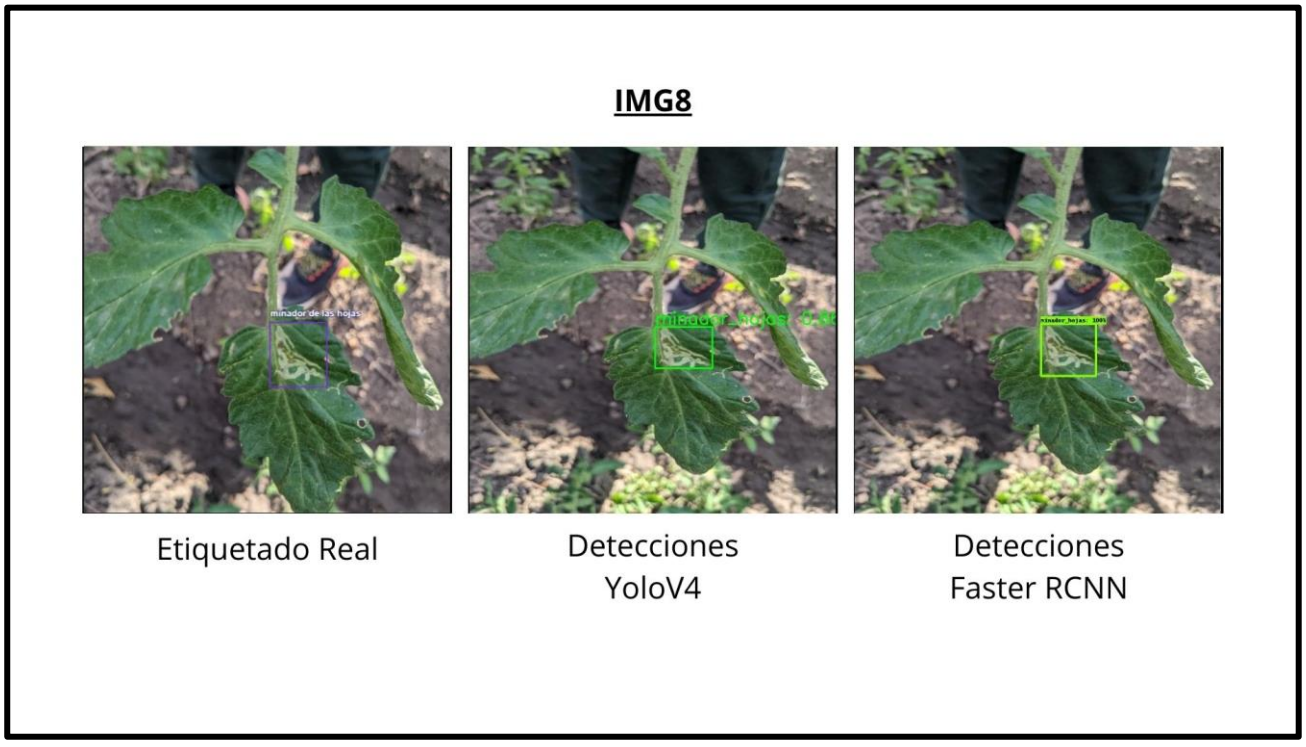


Figura 50 — Resultados de las detecciones IMG8

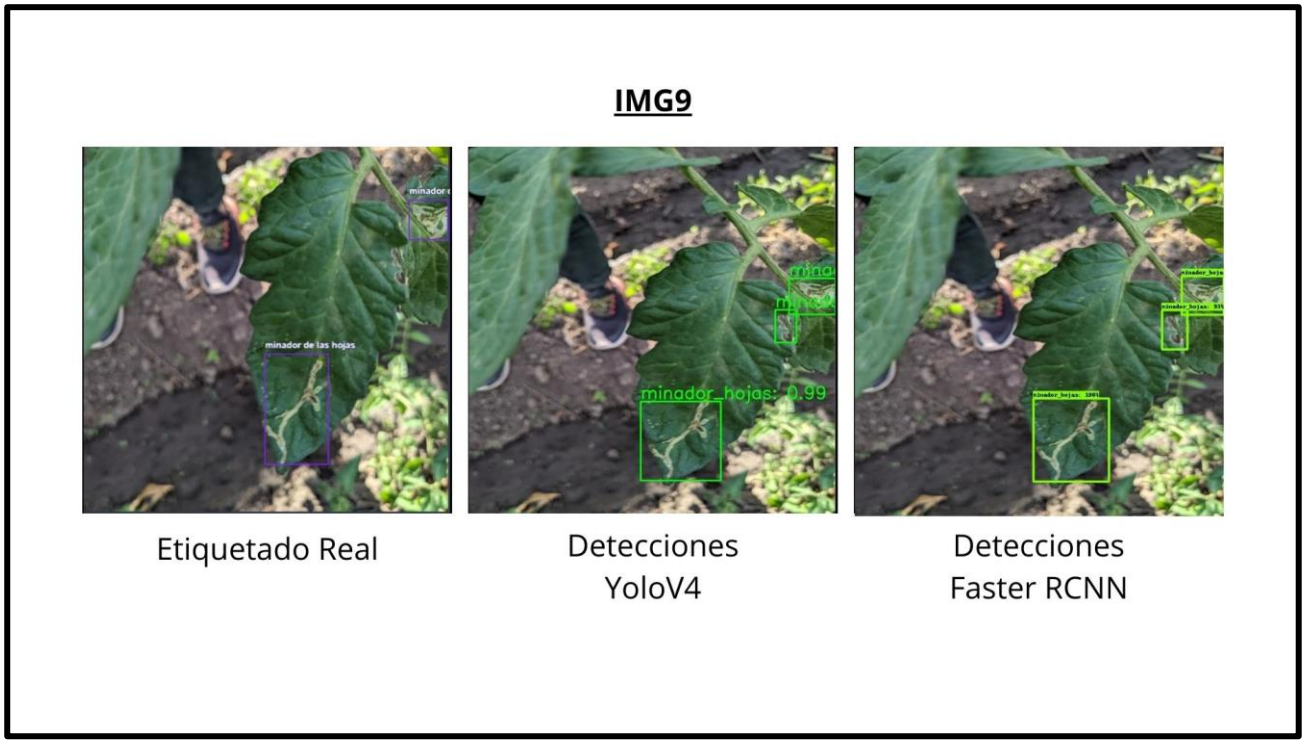


Figura 51 — Resultados de las detecciones IMG9

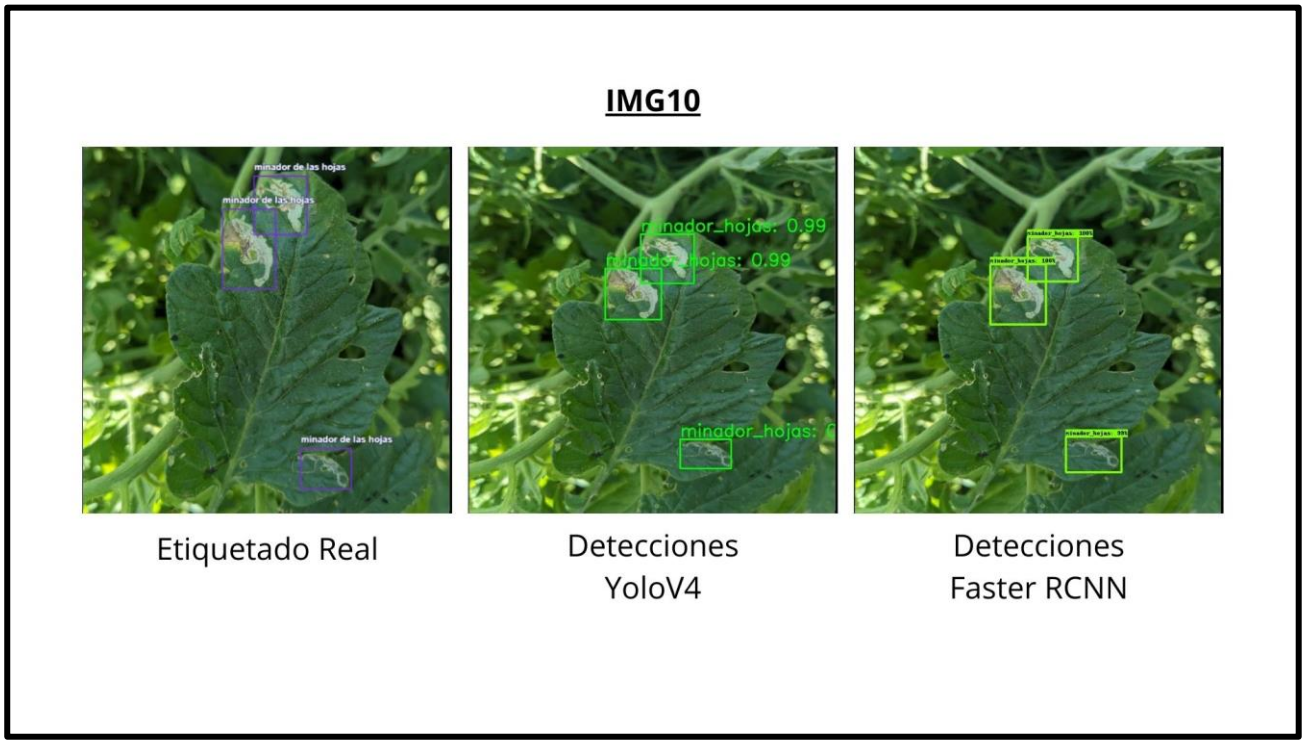


Figura 52 — Resultados de las detecciones IMG10

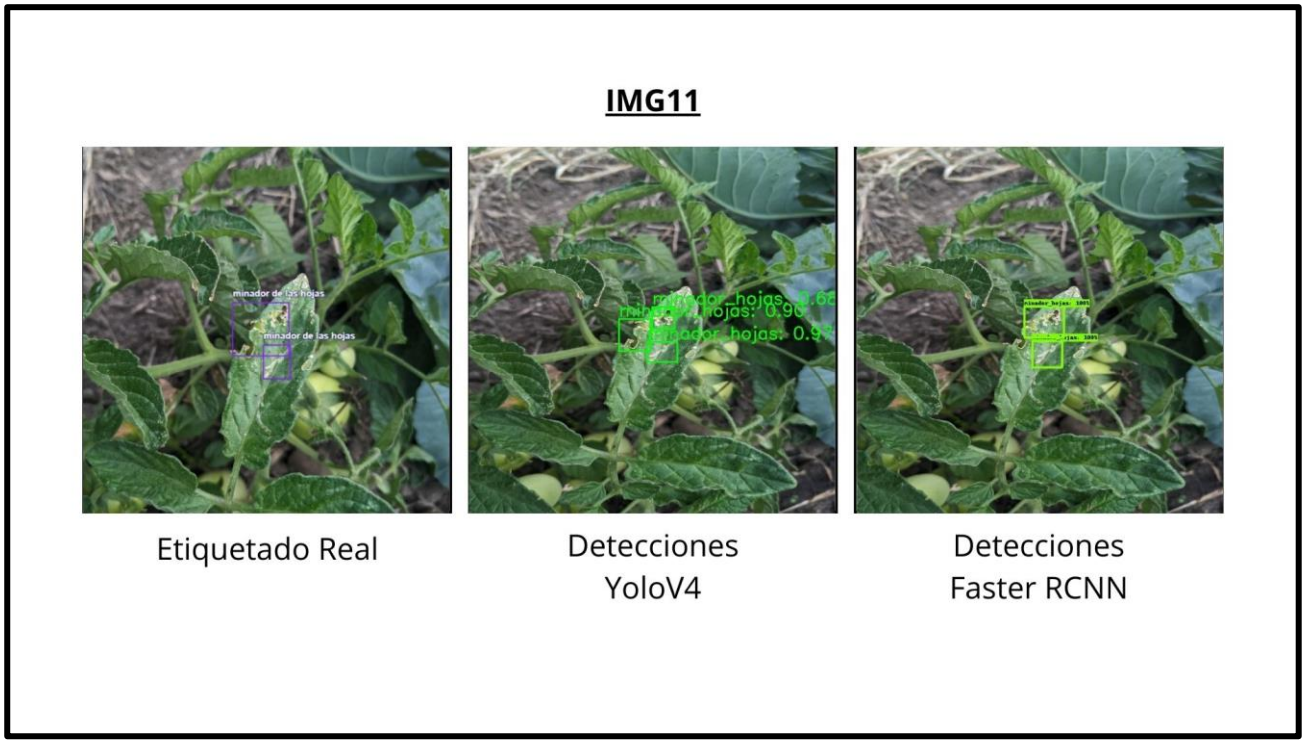


Figura 53 — Resultados de las detecciones IMG11

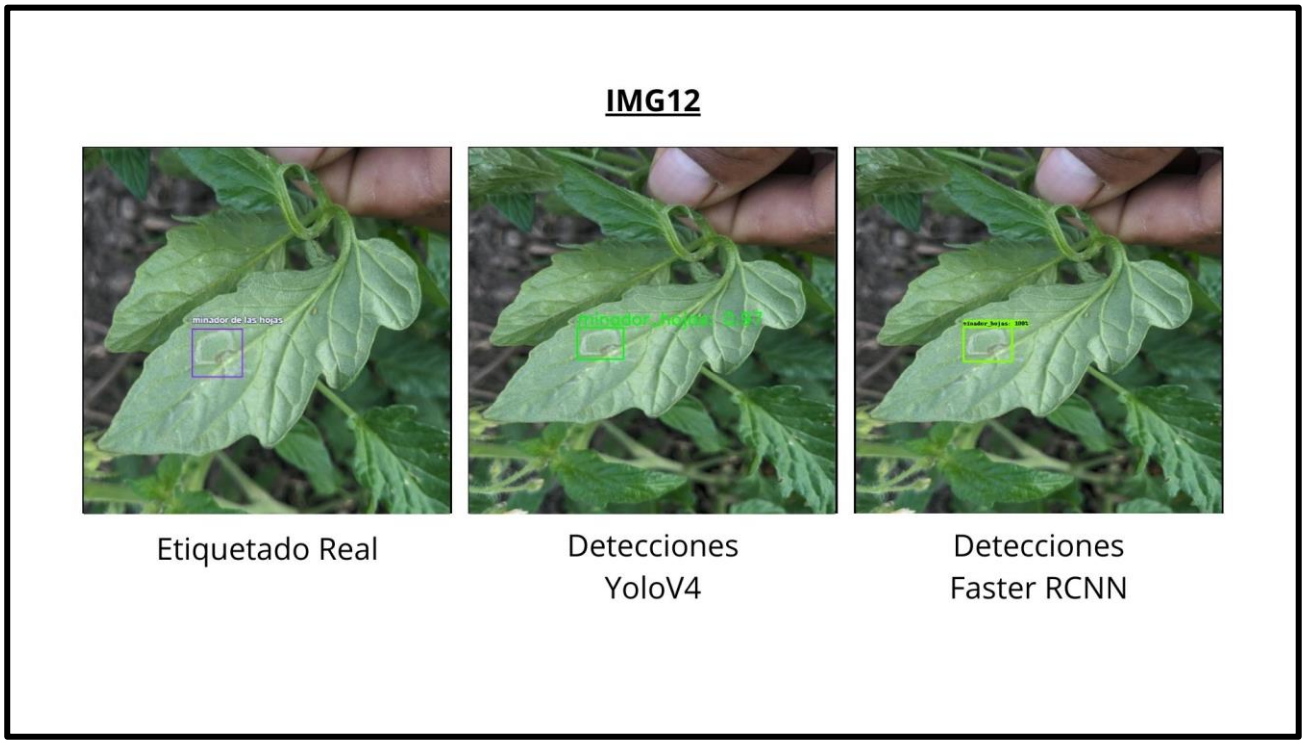


Figura 54 — Resultados de las detecciones IMG12

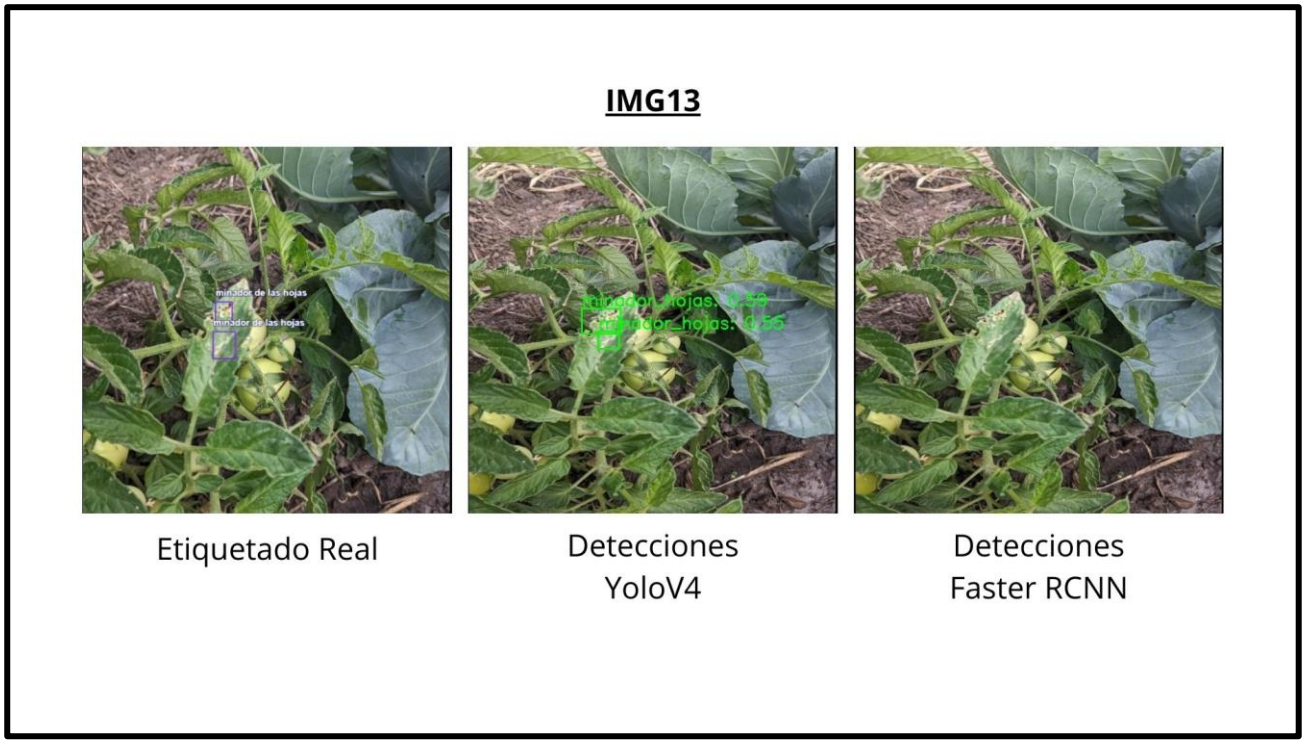


Figura 55 — Resultados de las detecciones IMG13

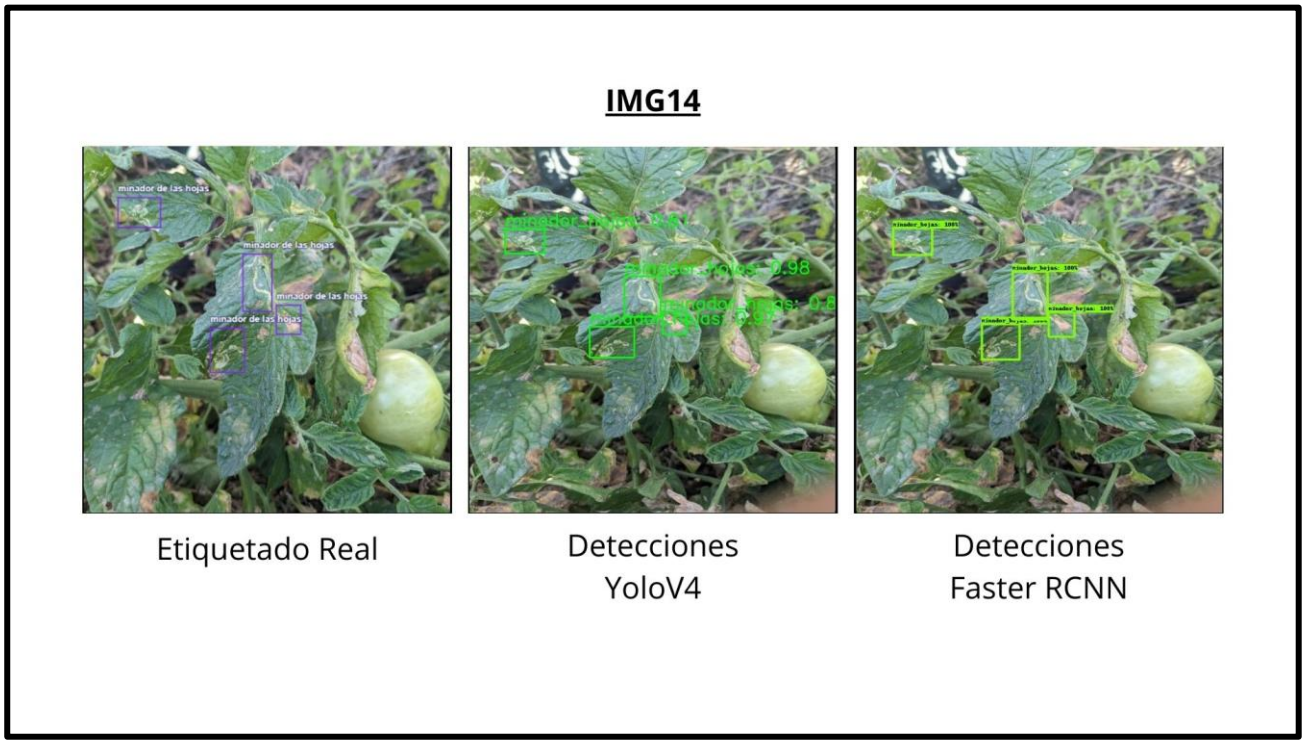


Figura 56 — Resultados de las detecciones IMG14

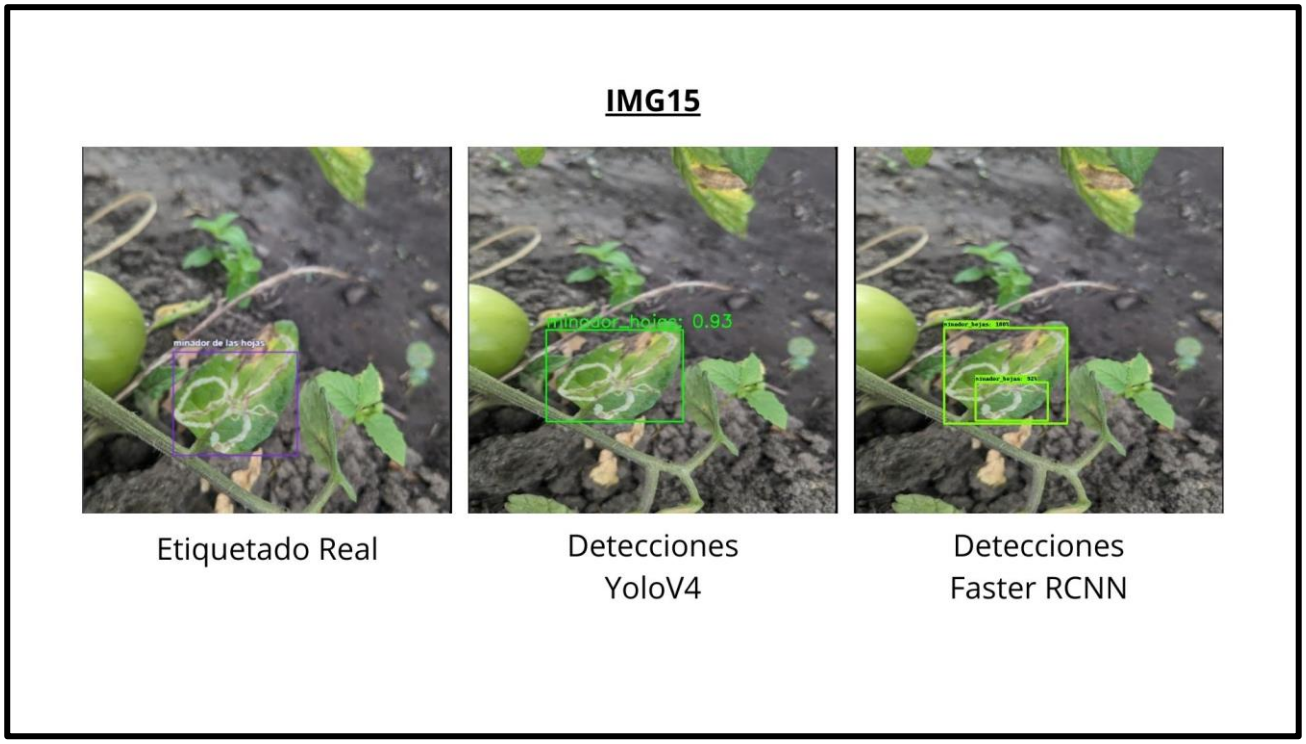


Figura 57— Resultados de las detecciones IMG15

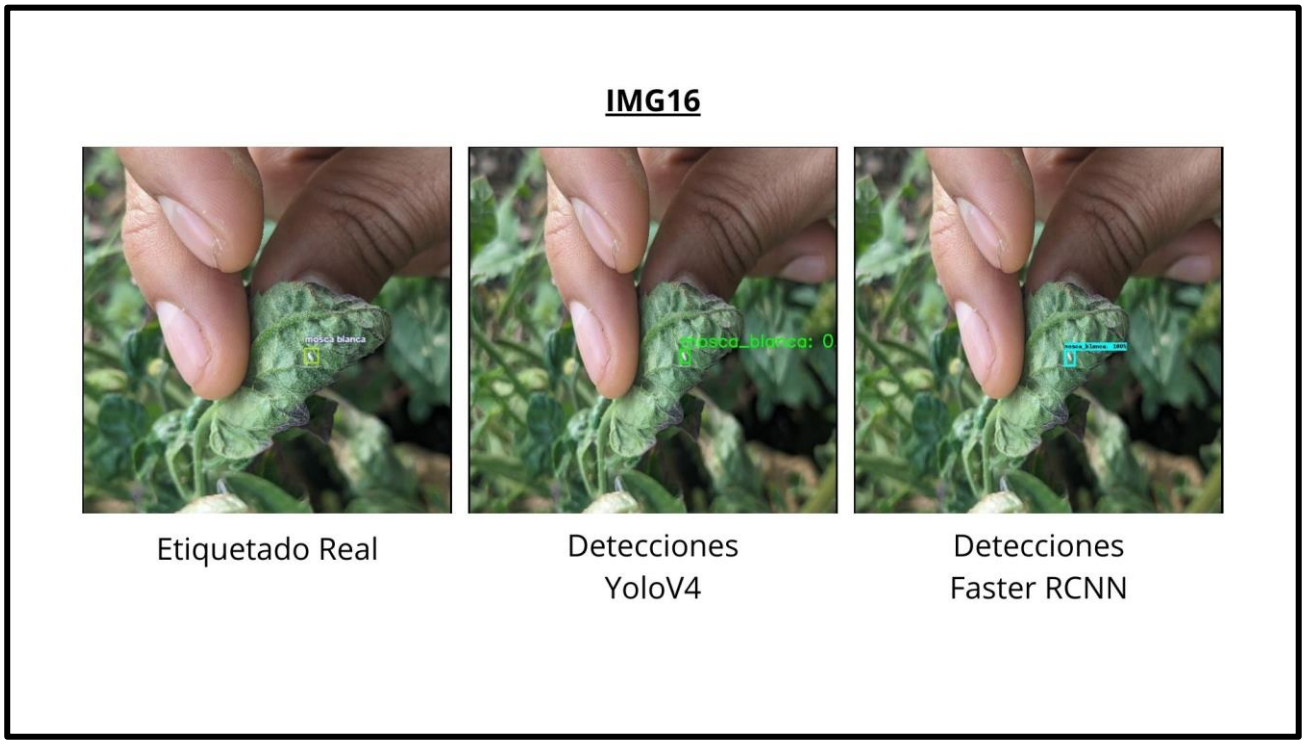


Figura 58 — Resultados de las detecciones IMG16

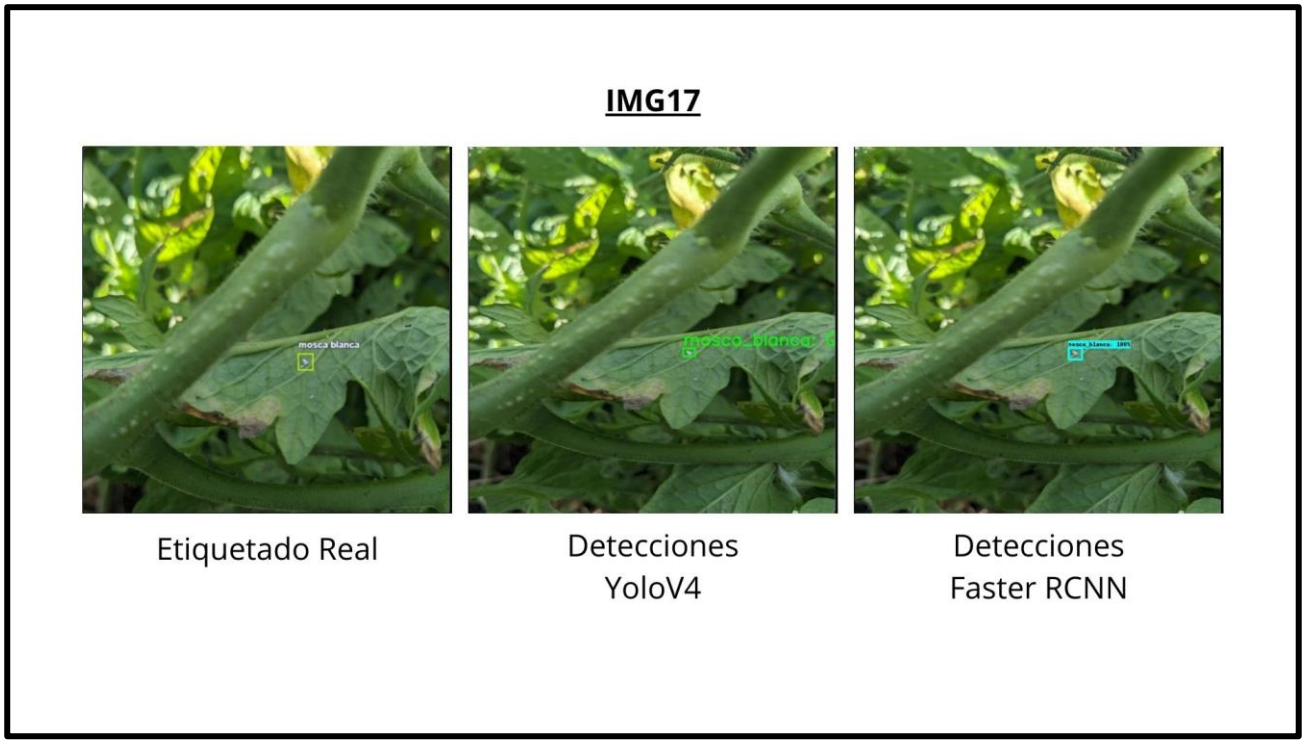


Figura 59 — Resultados de las detecciones IMG17

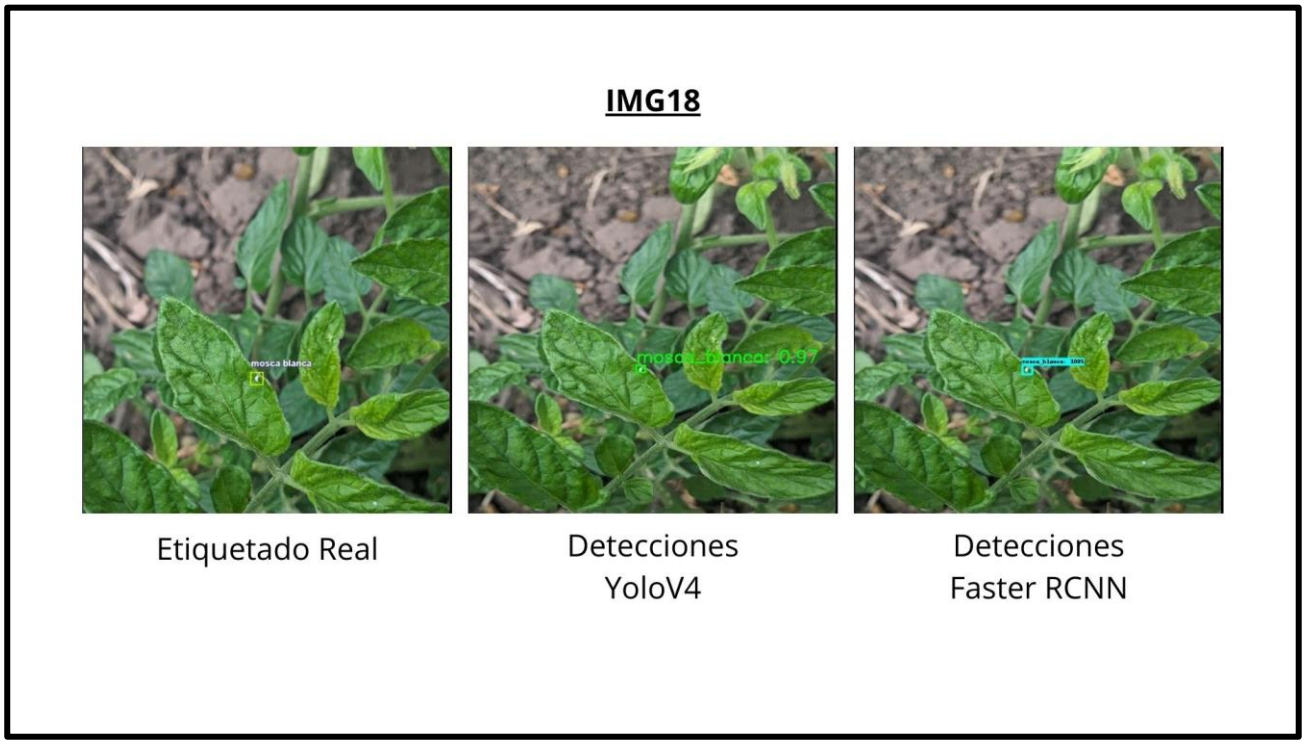


Figura 60 — Resultados de las detecciones IMG18

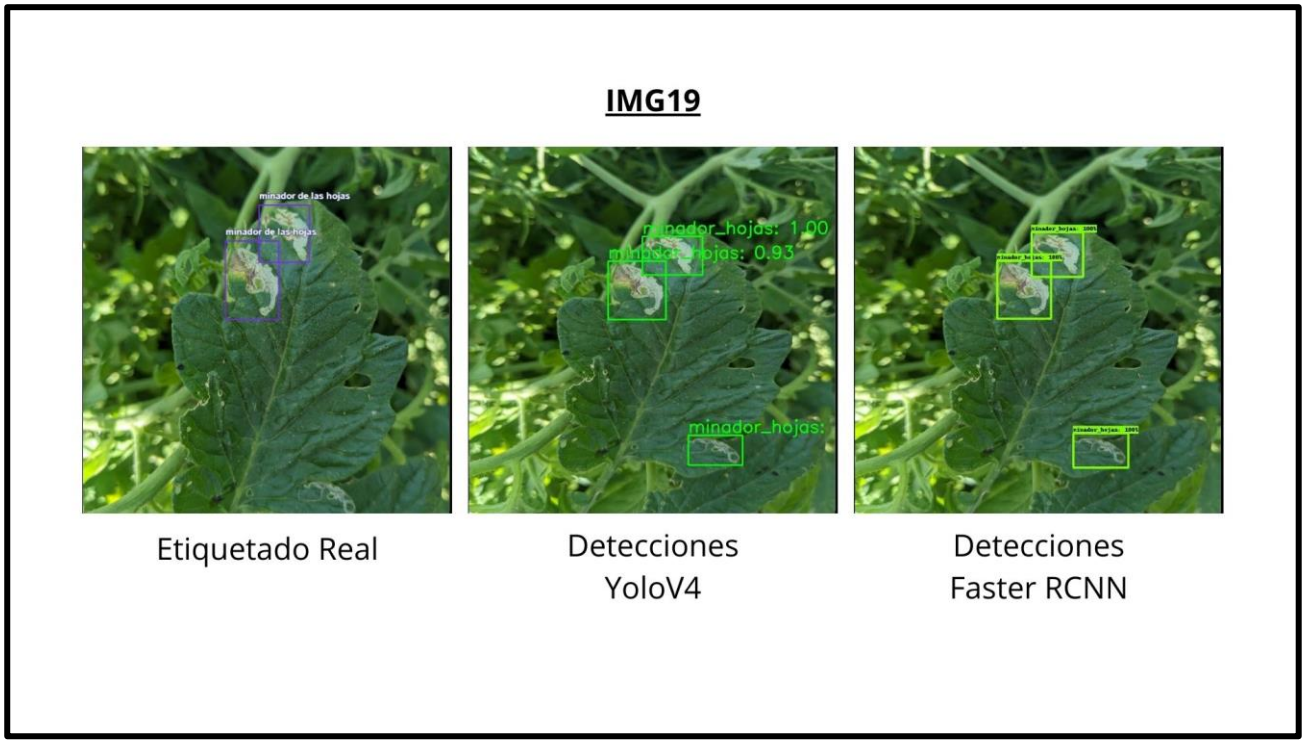


Figura 61 — Resultados de las detecciones IMG19

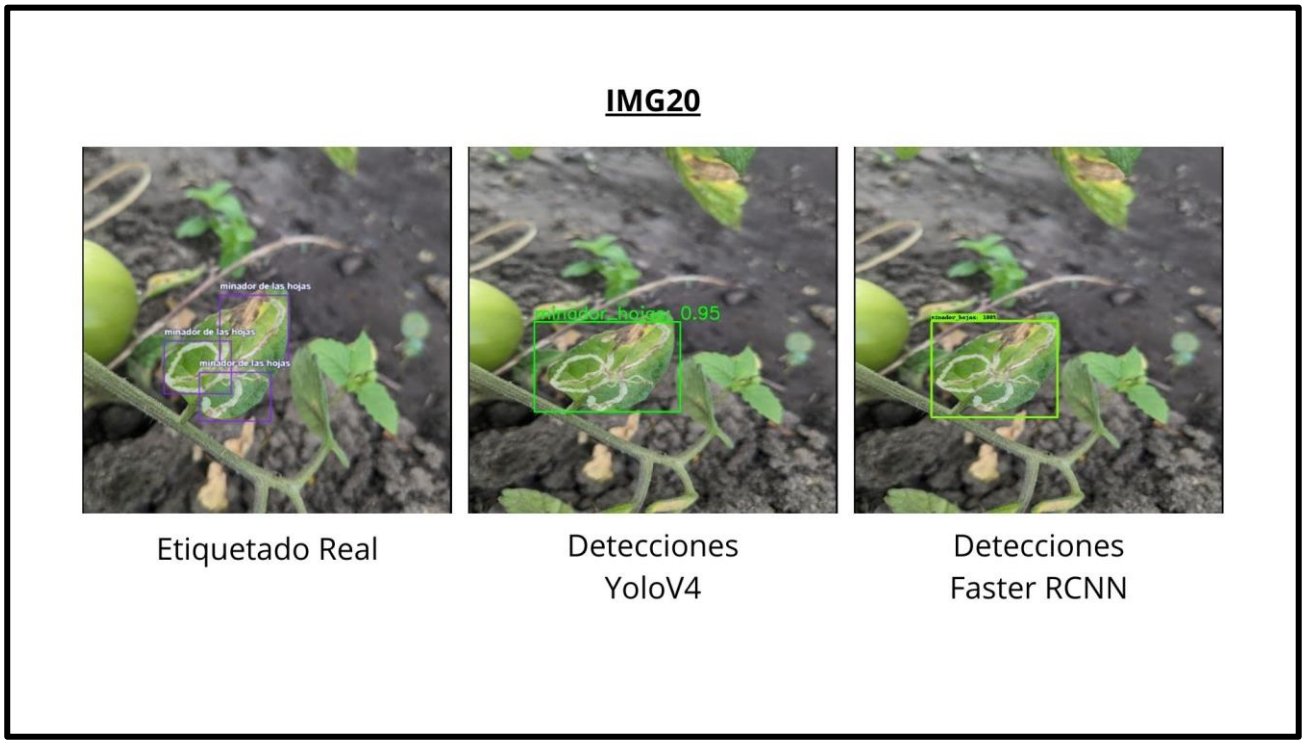


Figura 62 — Resultados de las detecciones IMG20

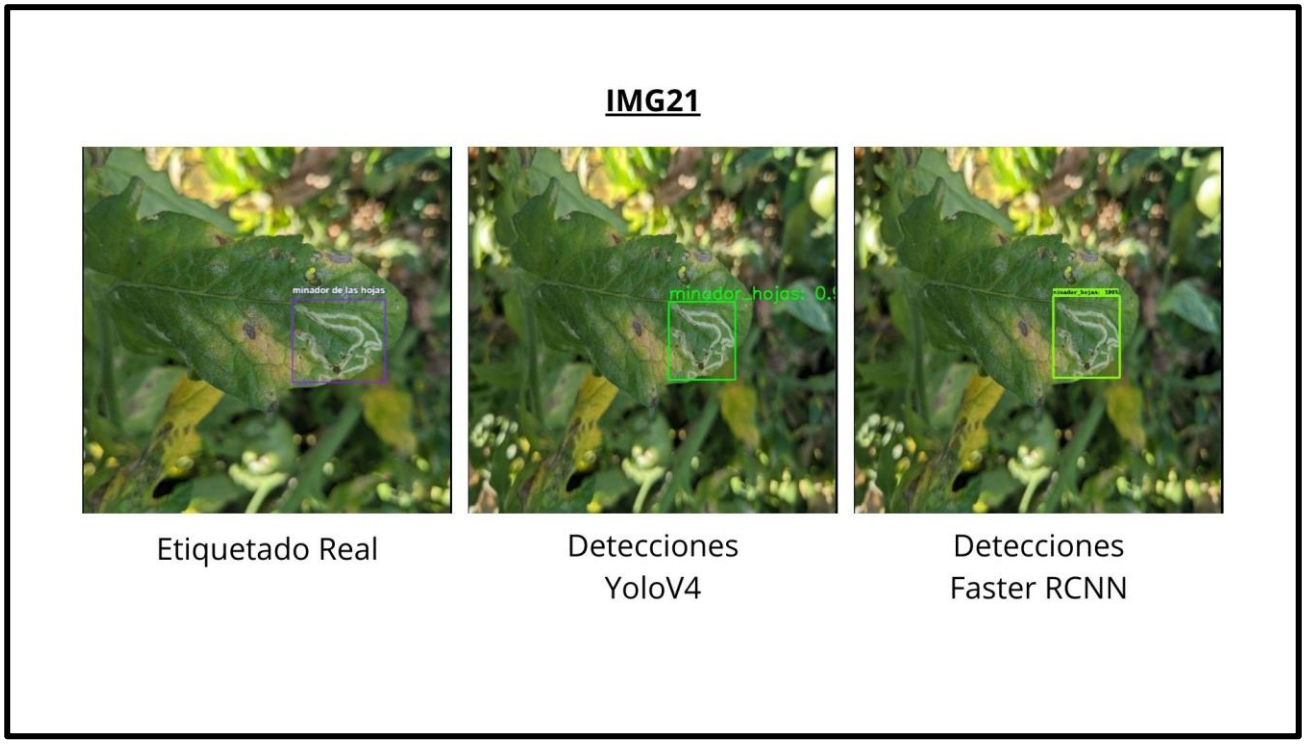


Figura 63 — Resultados de las detecciones IMG21

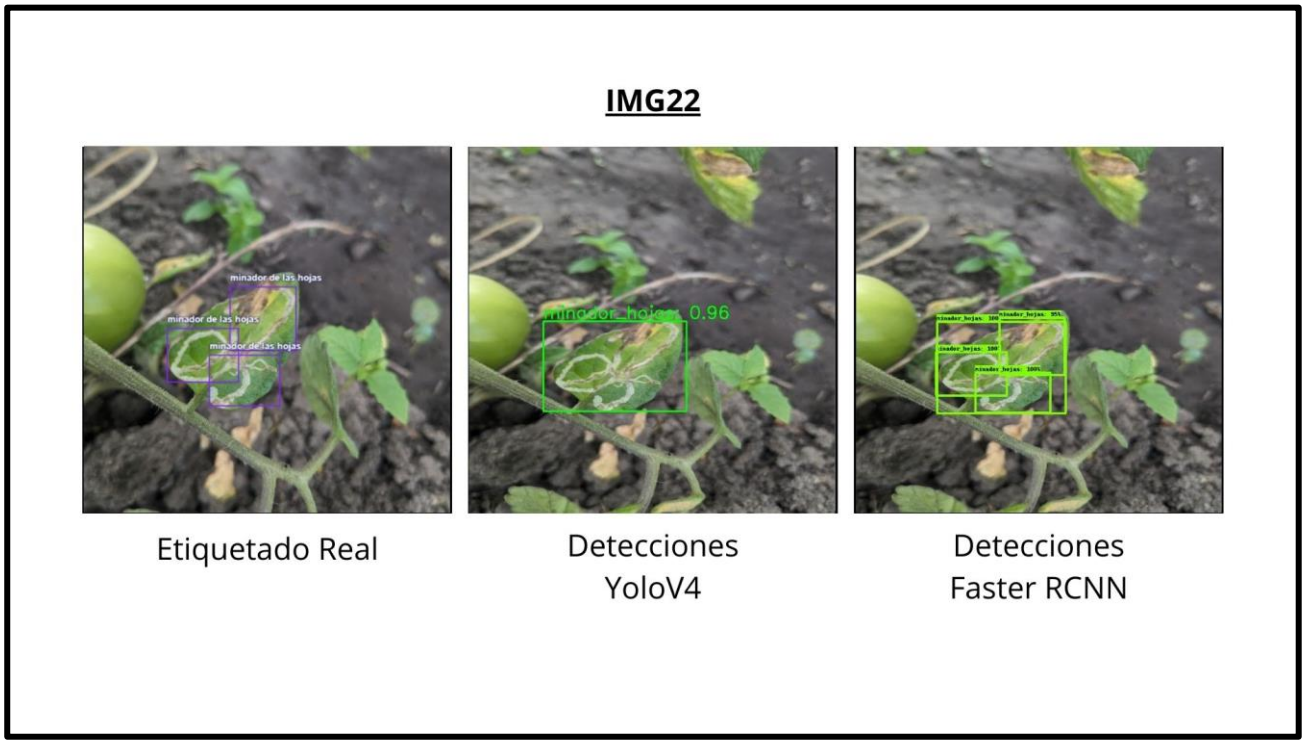


Figura 64 — Resultados de las detecciones IMG22

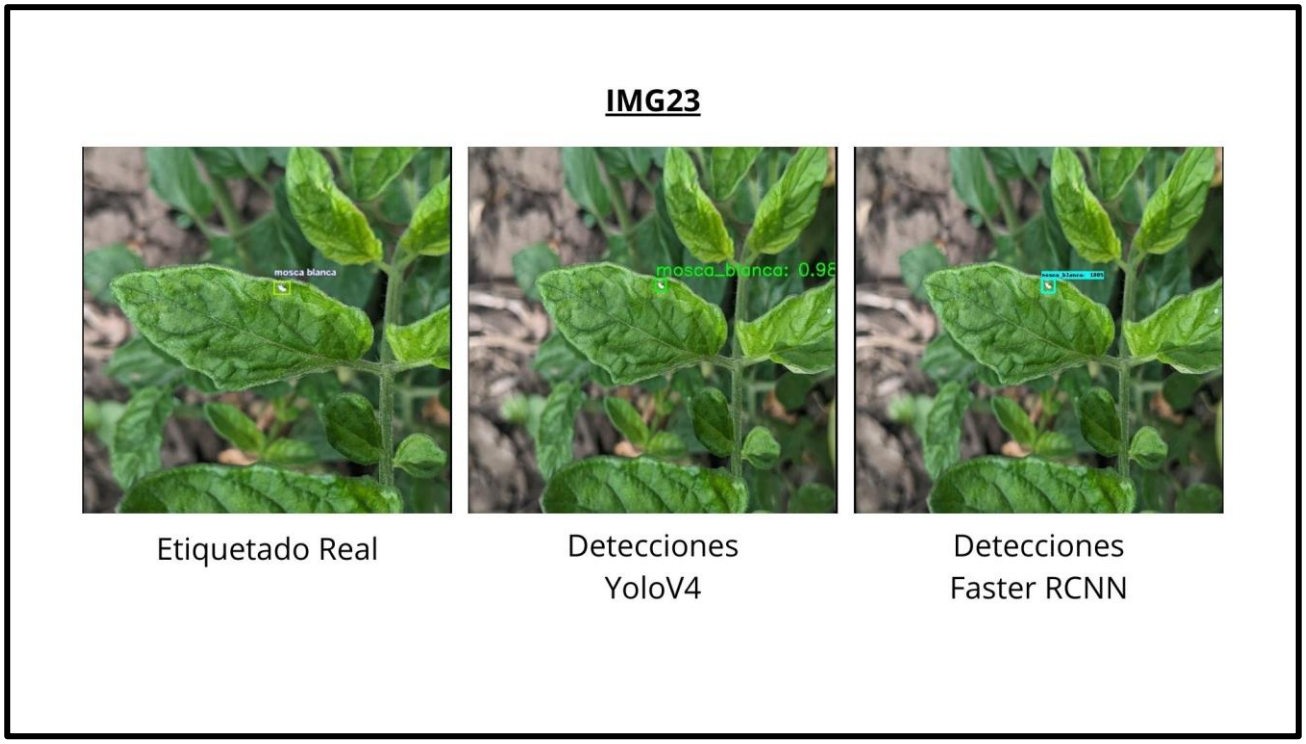


Figura 65 — Resultados de las detecciones IMG23

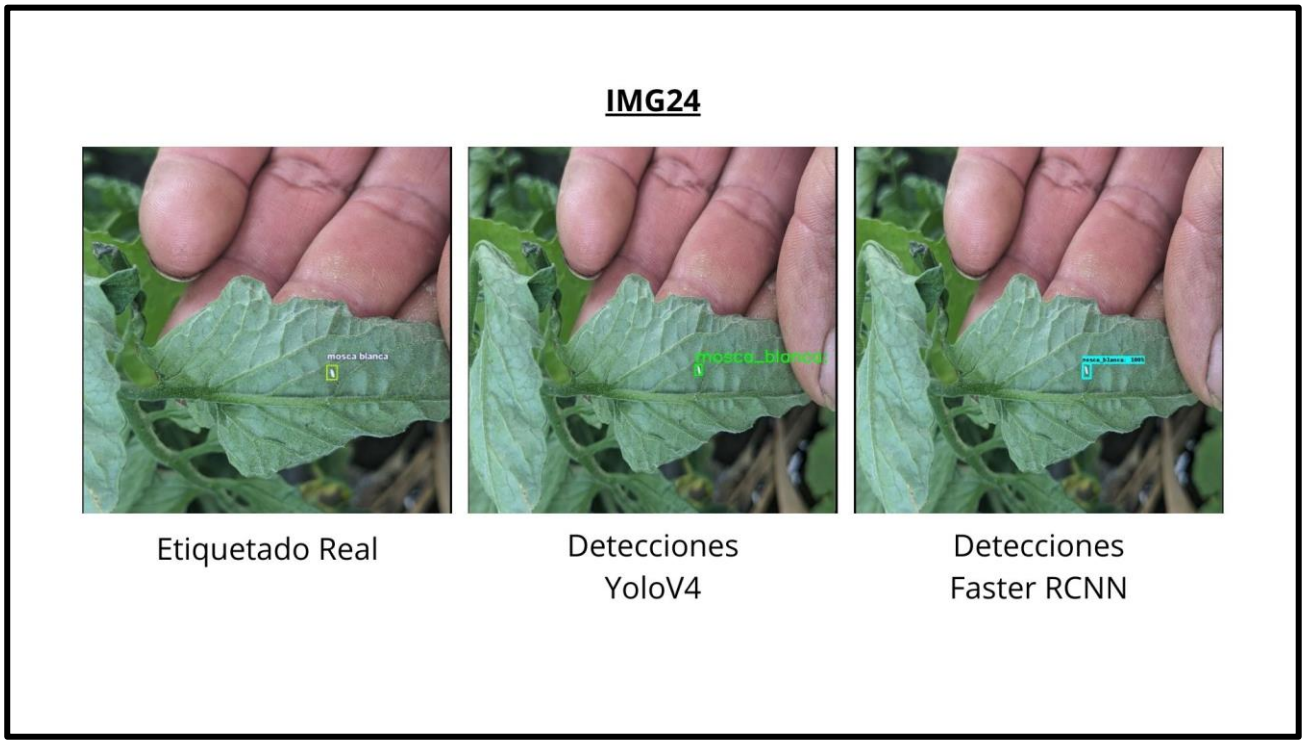


Figura 66 — Resultados de las detecciones IMG24

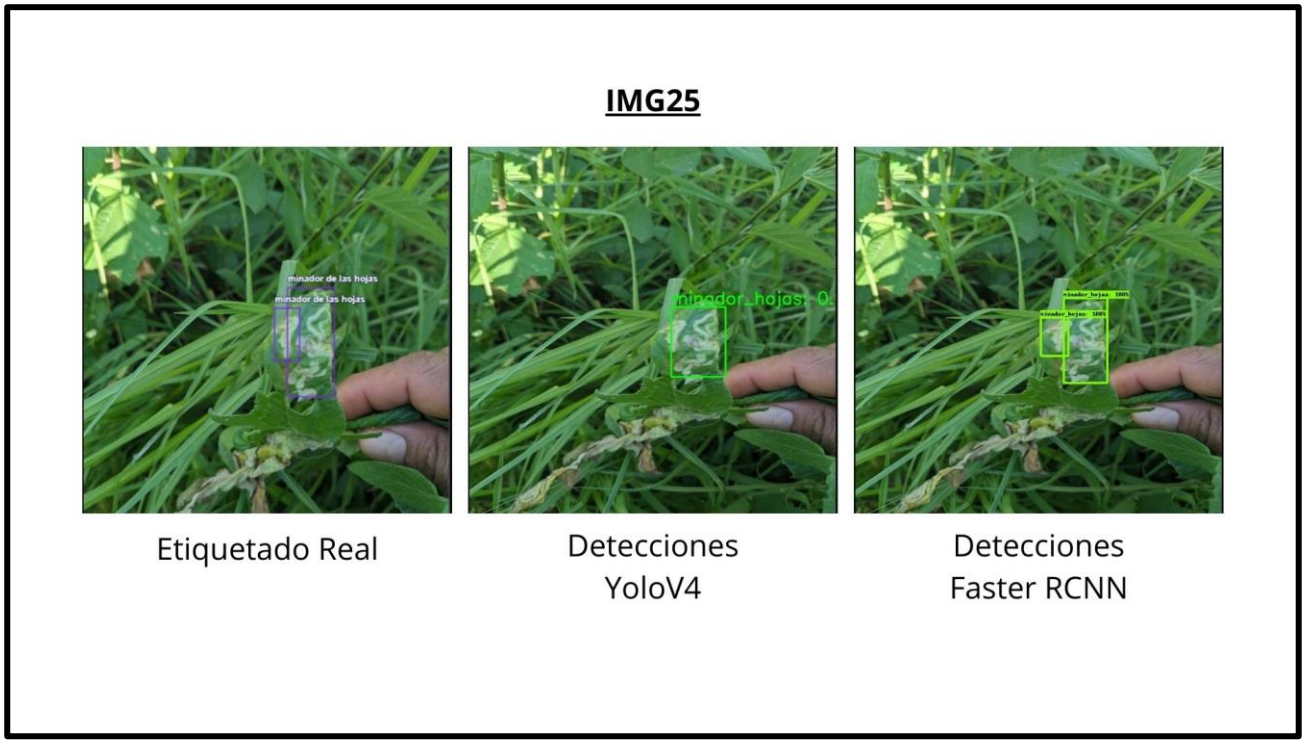


Figura 67 — Resultados de las detecciones IMG25

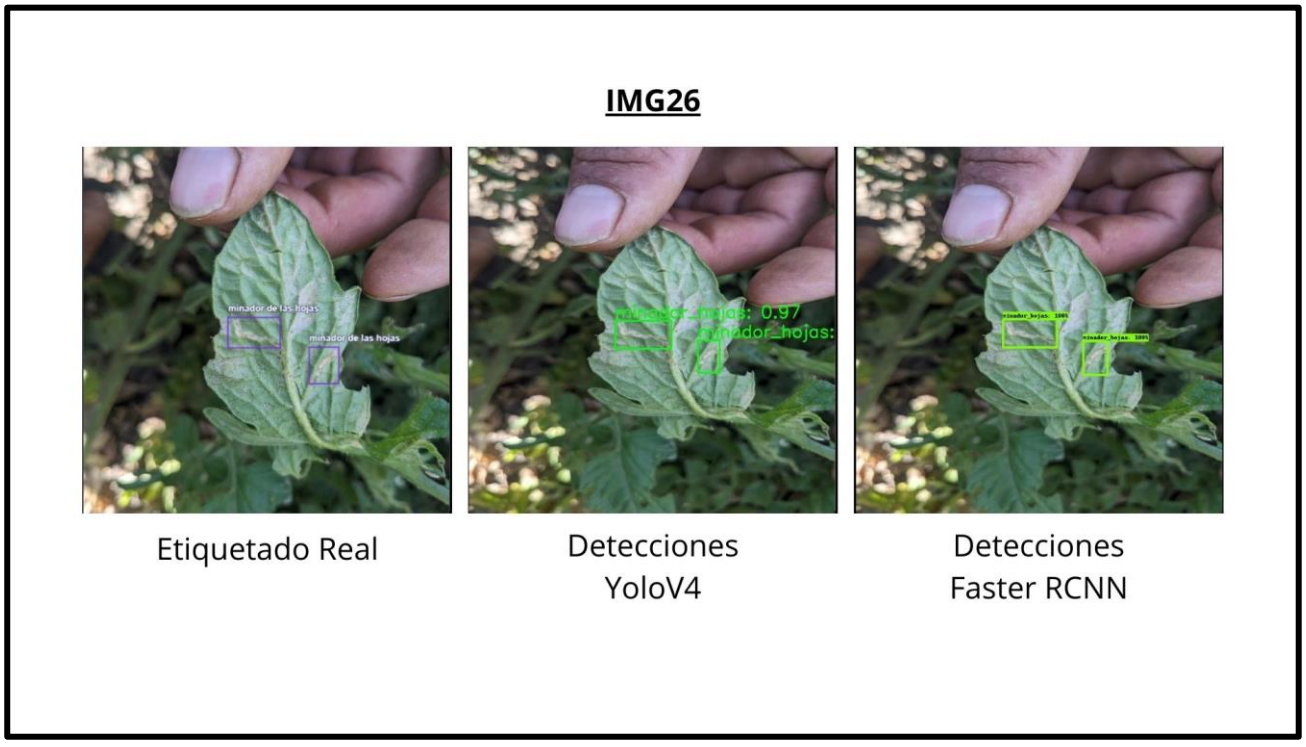


Figura 68 — Resultados de las detecciones IMG26

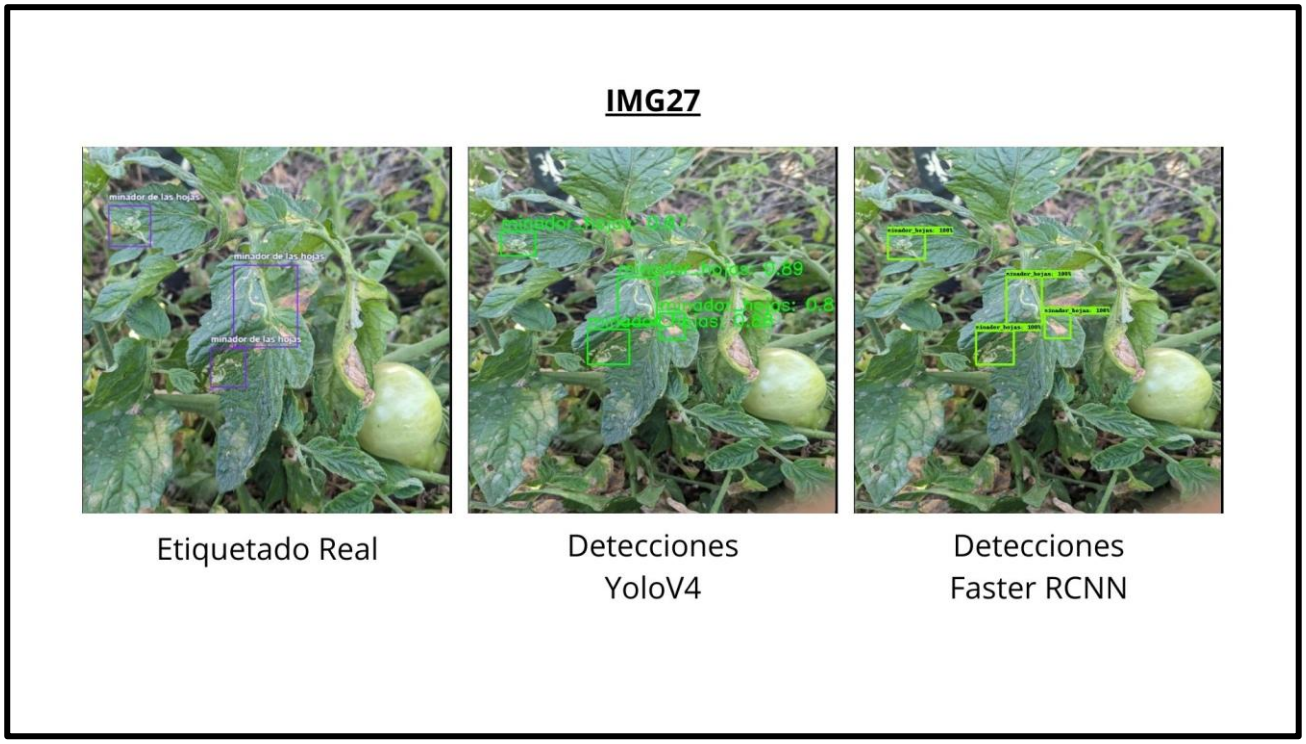


Figura 69 — Resultados de las detecciones IMG27

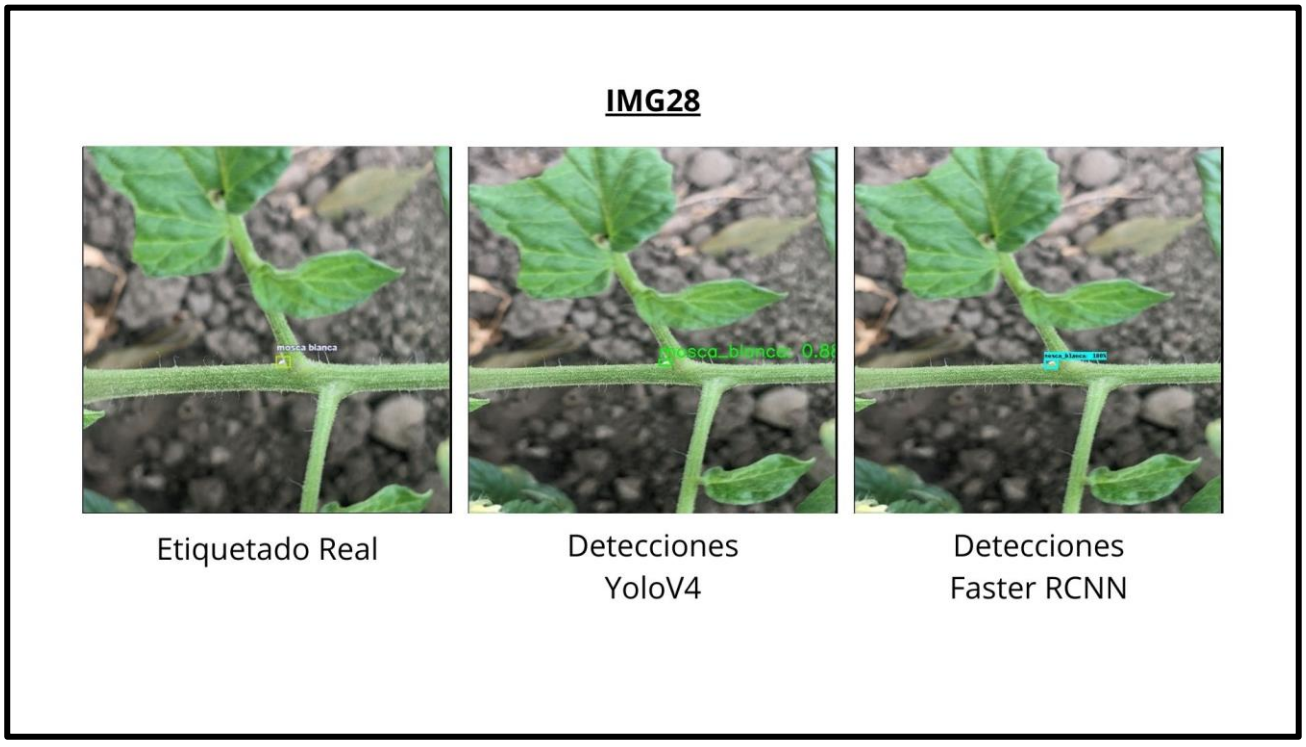


Figura 70 — Resultados de las detecciones IMG28

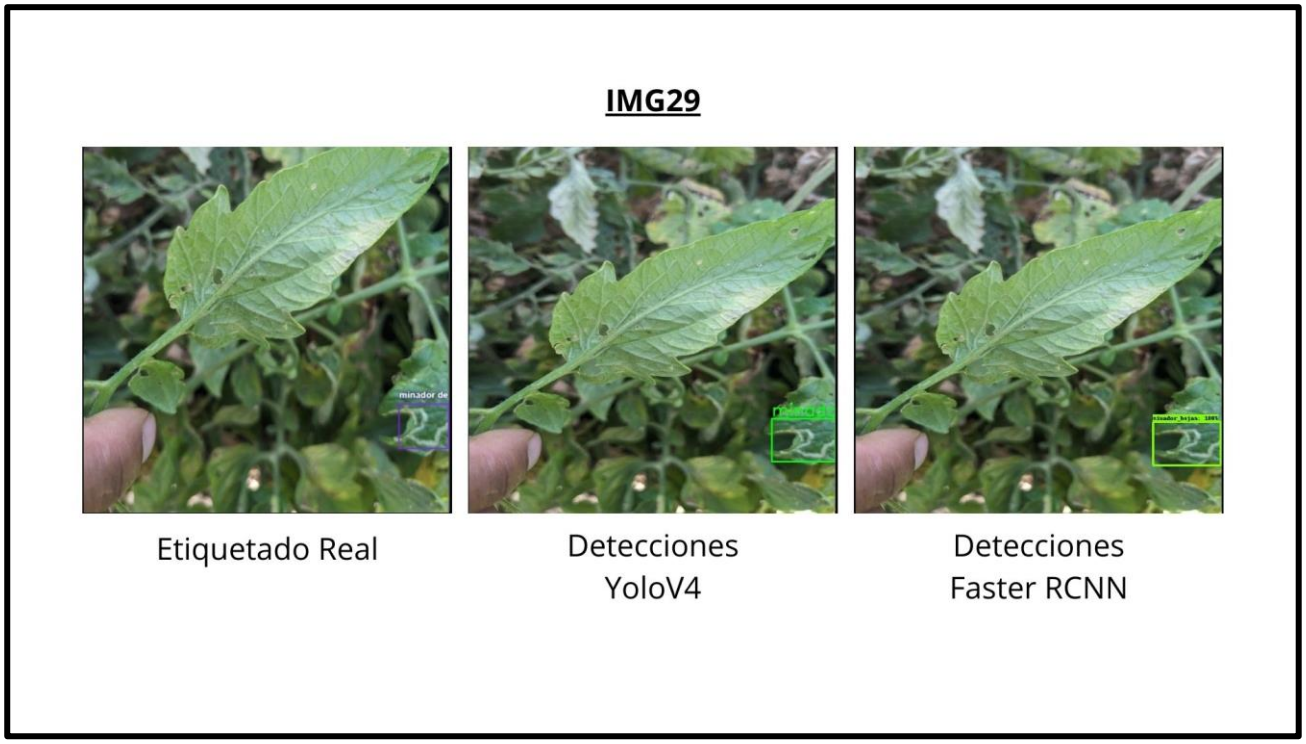


Figura 71 — Resultados de las detecciones IMG29

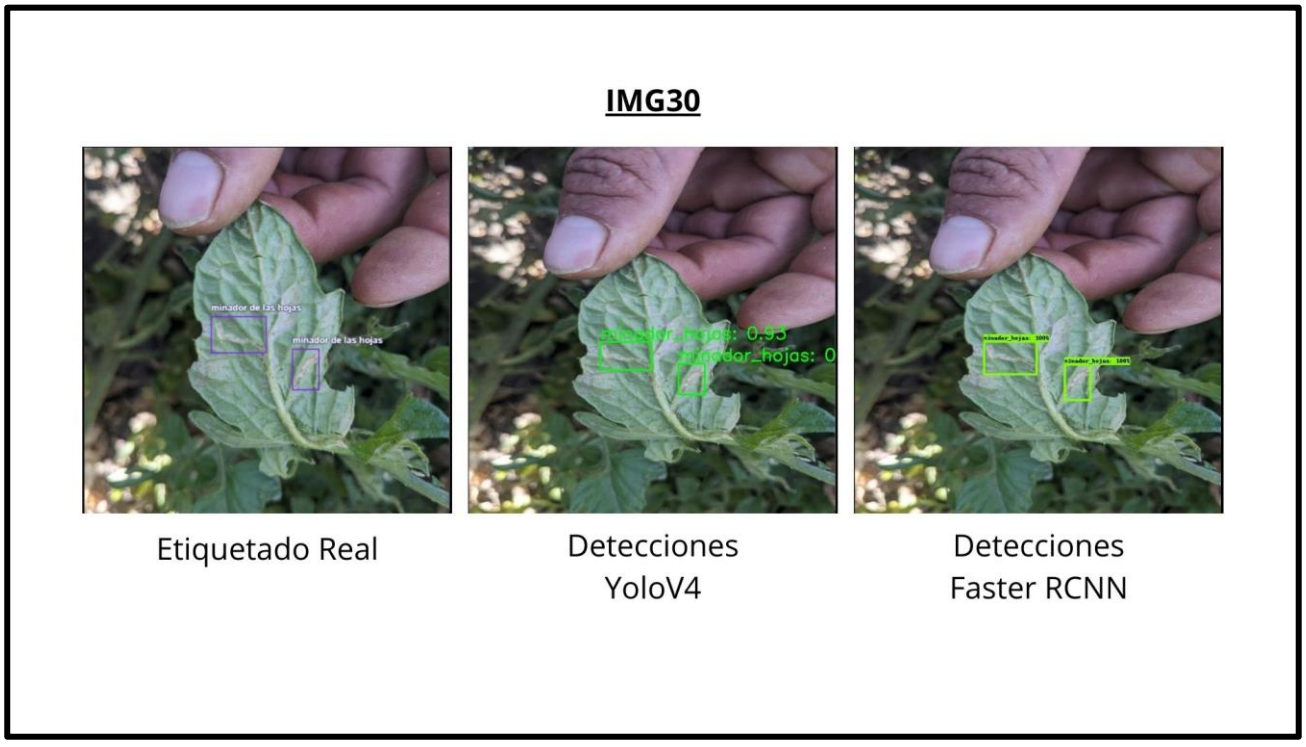


Figura 72 — Resultados de las detecciones IMG30

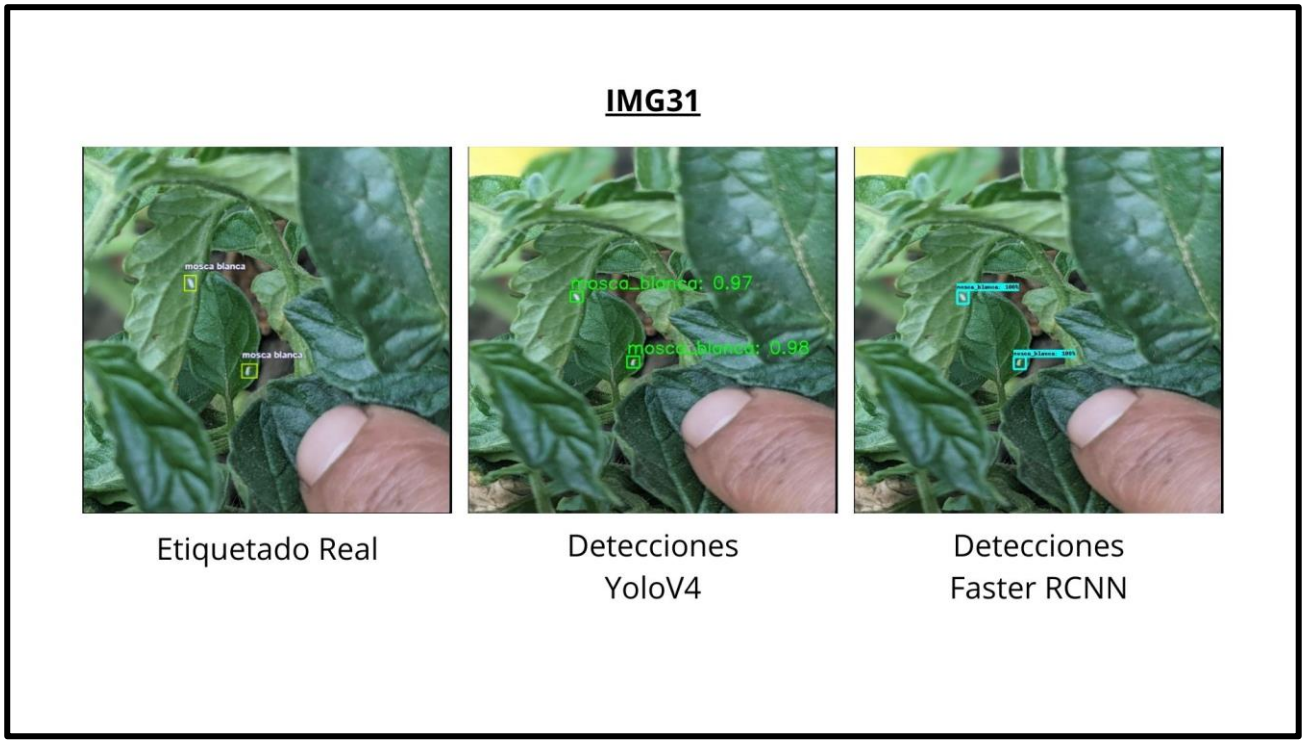


Figura 73 — Resultados de las detecciones IMG31

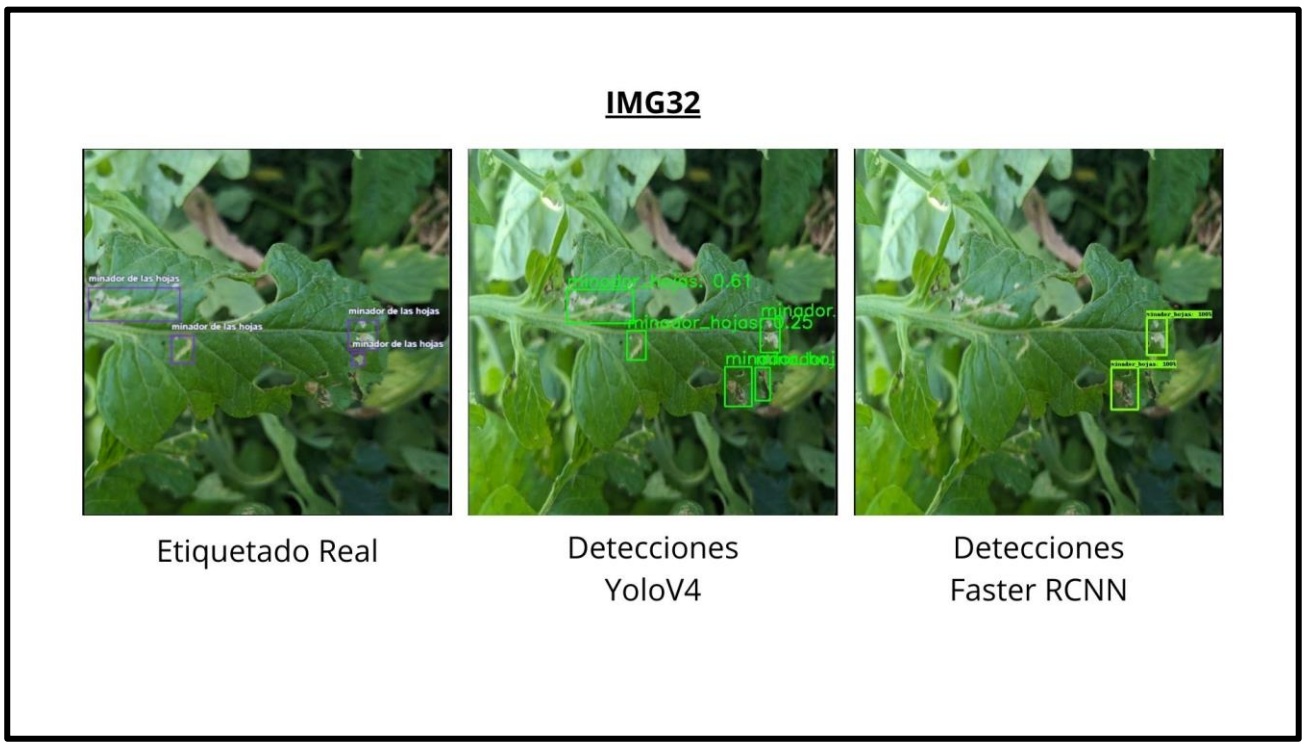


Figura 74 — Resultados de las detecciones IMG32

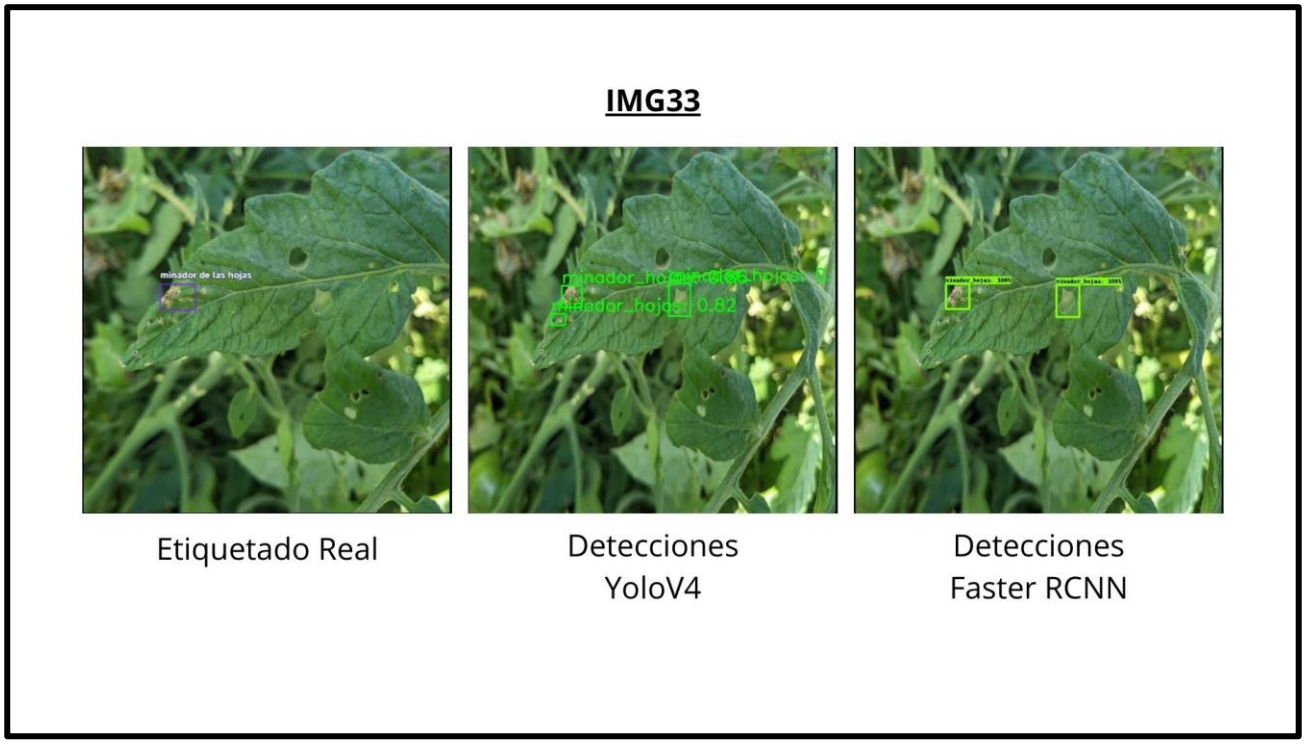


Figura 75 — Resultados de las detecciones IMG33

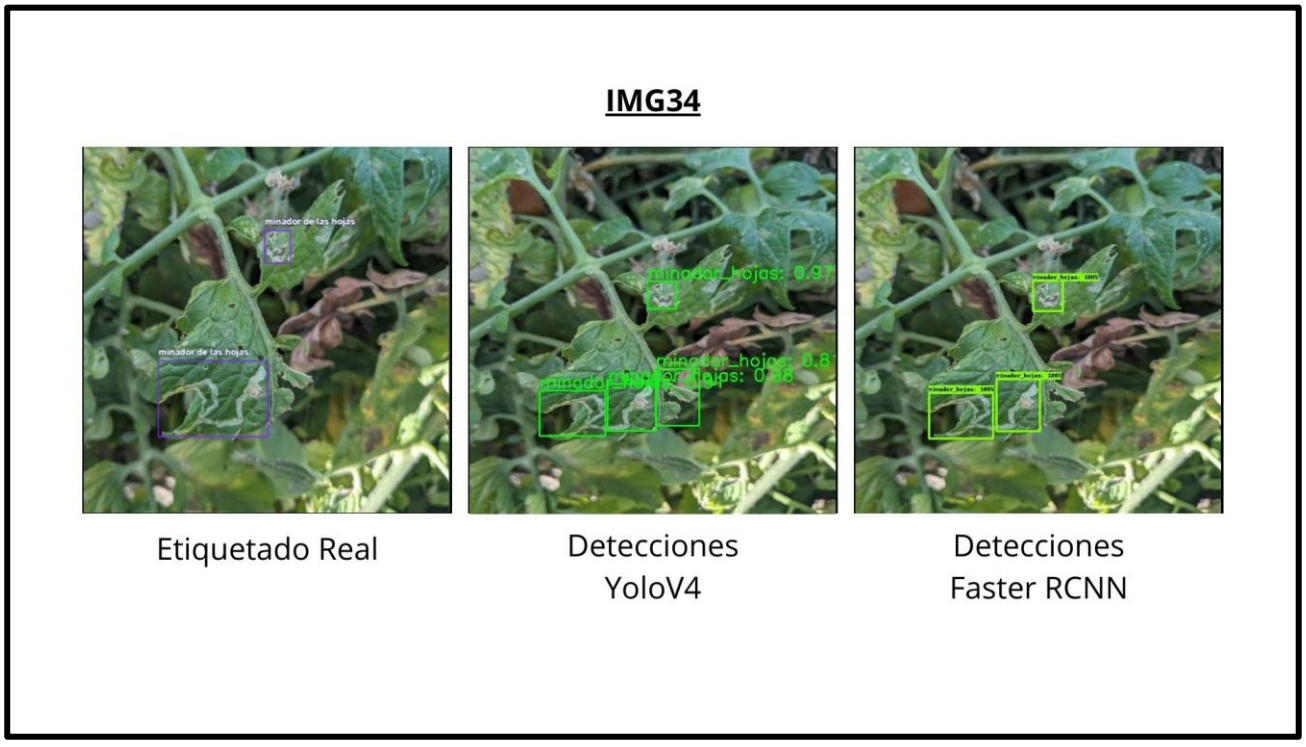


Figura 76 — Resultados de las detecciones IMG34

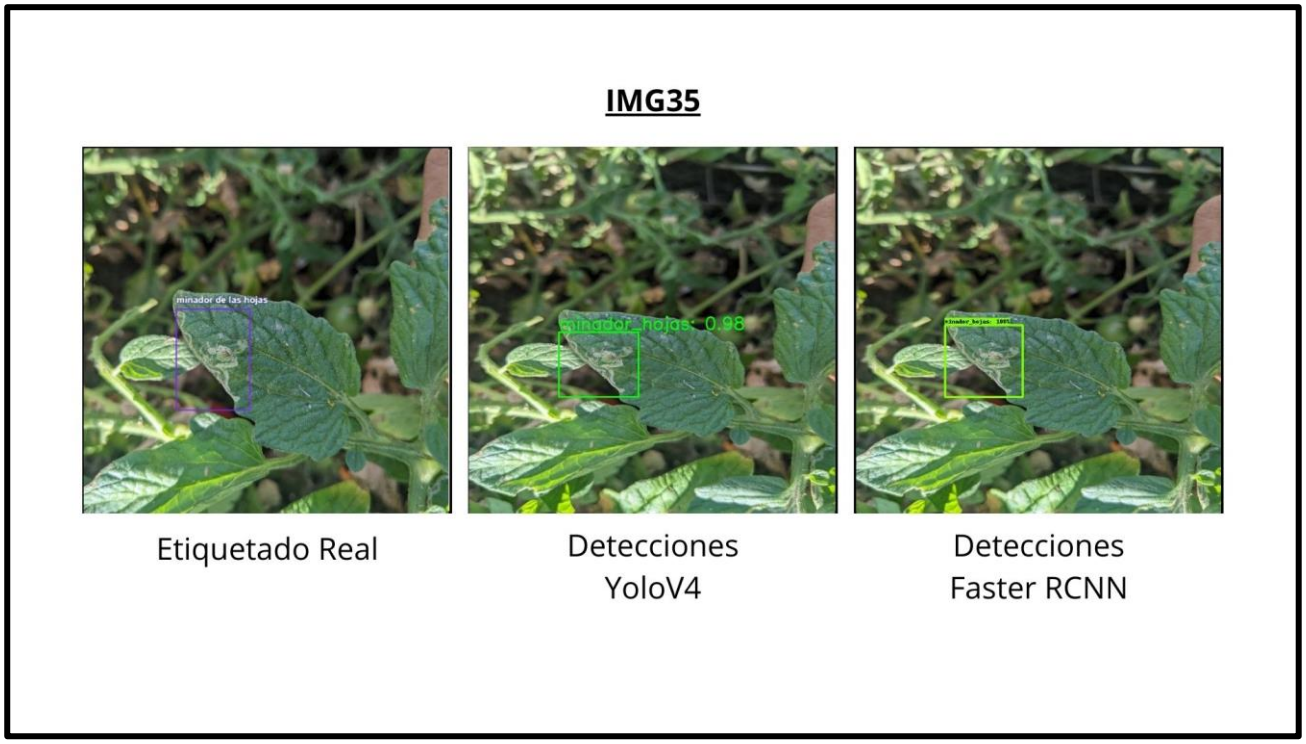


Figura 77 — Resultados de las detecciones IMG35

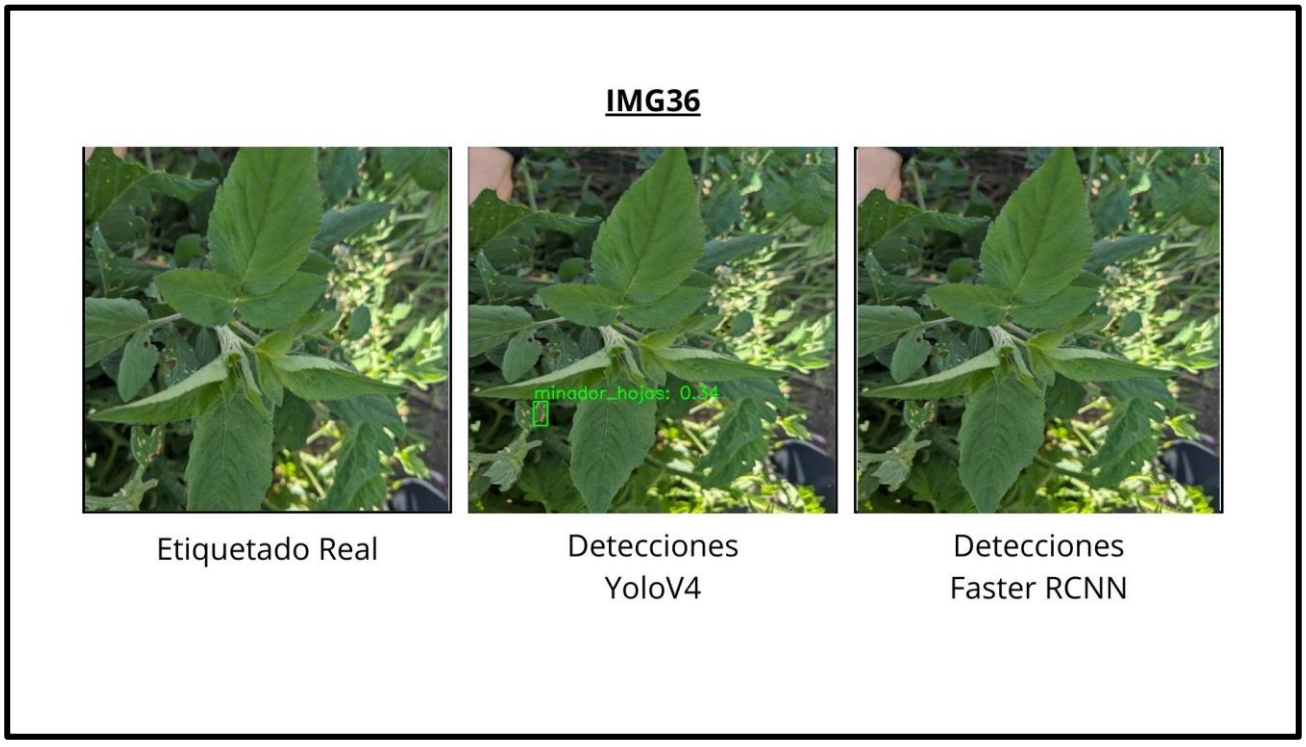


Figura 78 — Resultados de las detecciones IMG36

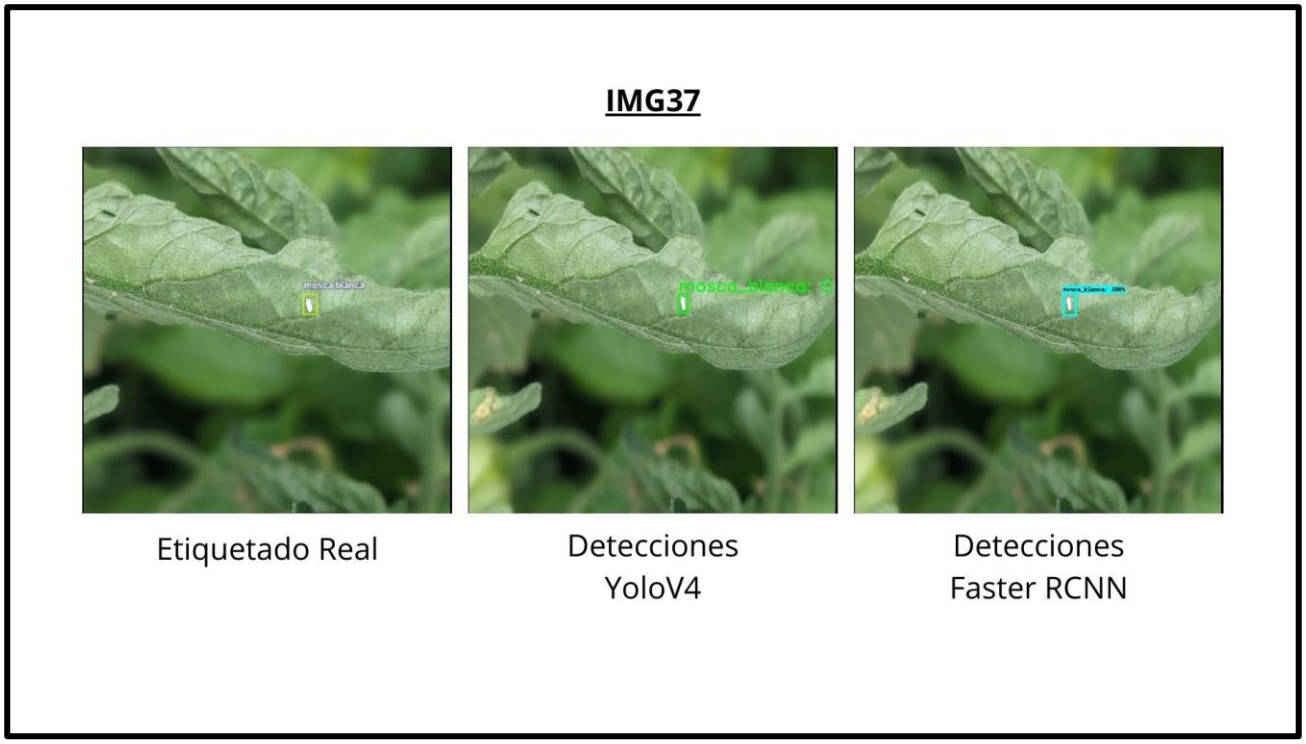


Figura 79 — Resultados de las detecciones IMG37

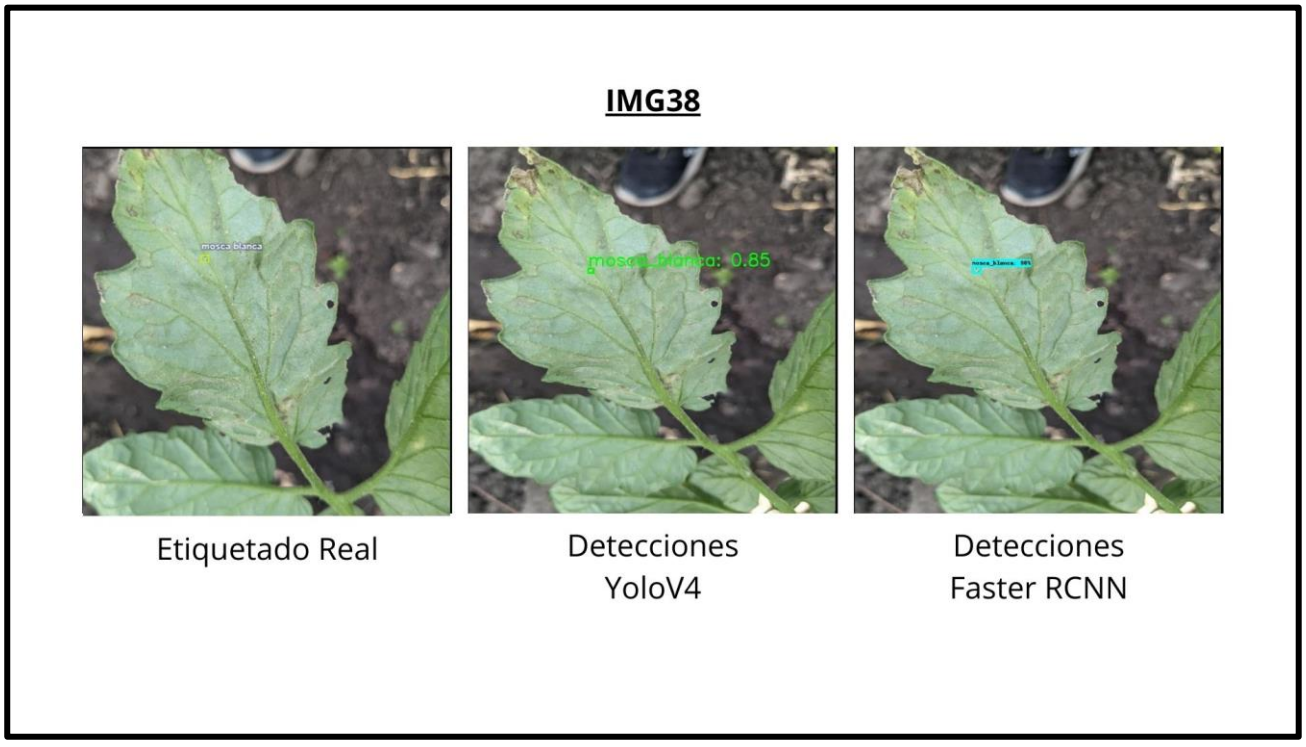


Figura 80 — Resultados de las detecciones IMG38

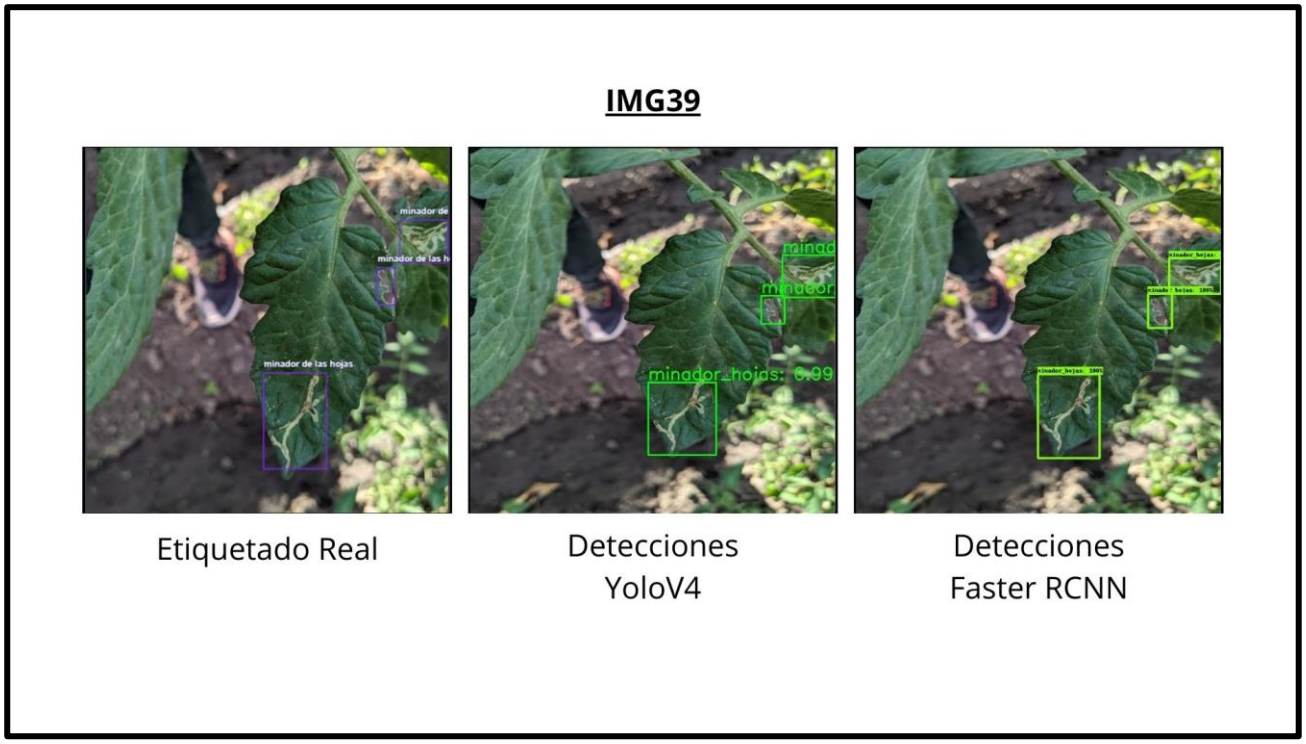


Figura 81 — Resultados de las detecciones IMG39

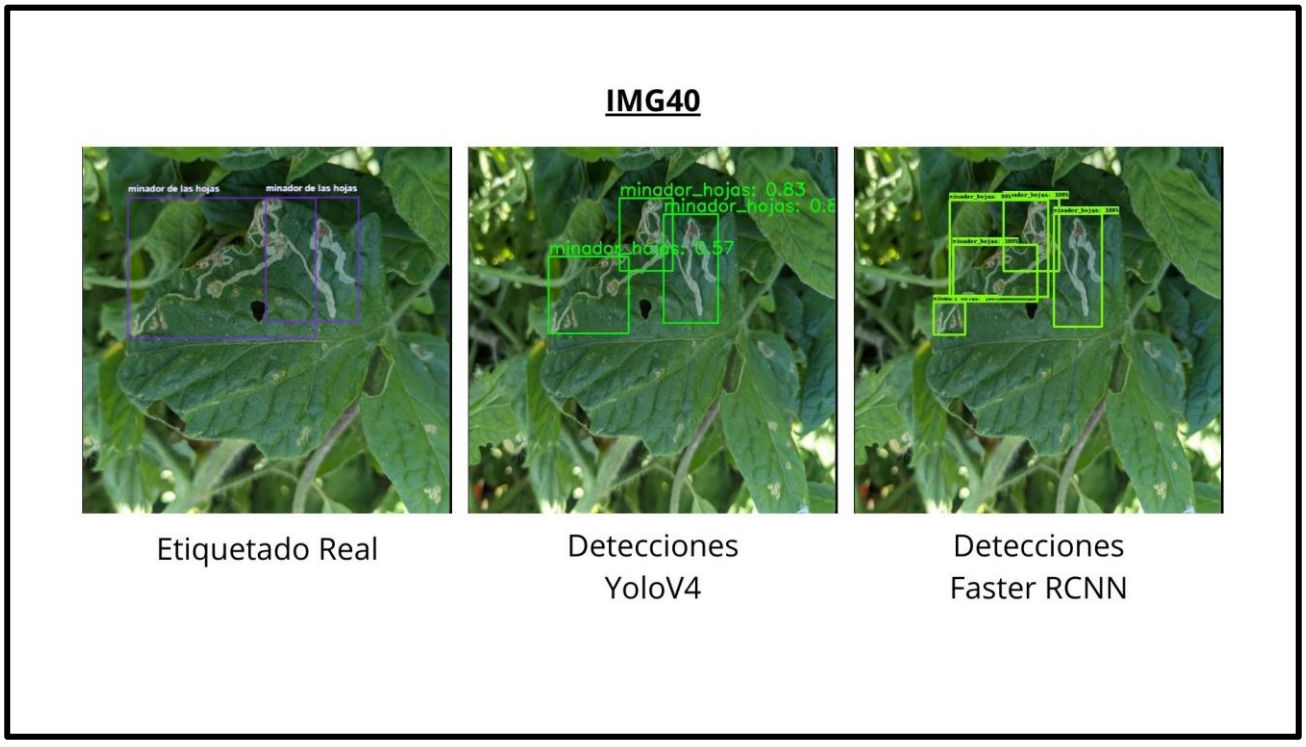


Figura 82 — Resultados de las detecciones IMG40

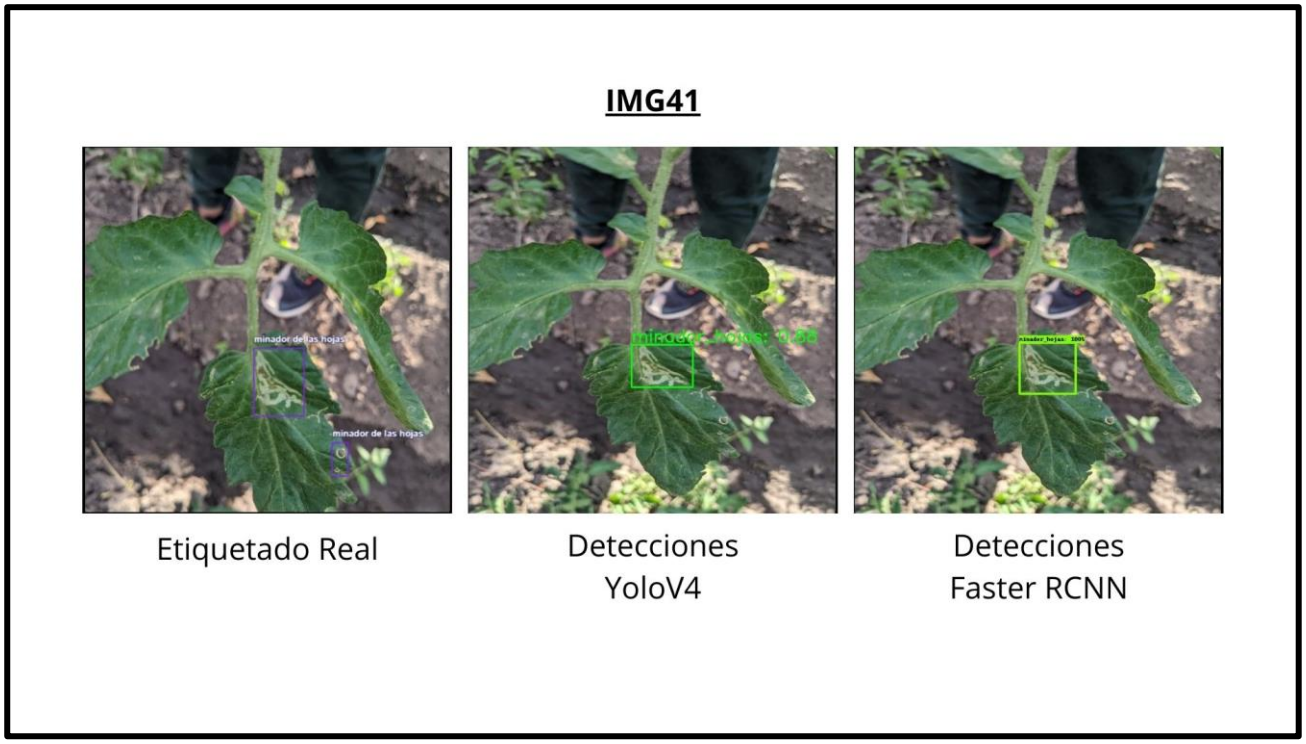


Figura 83 — Resultados de las detecciones IMG41

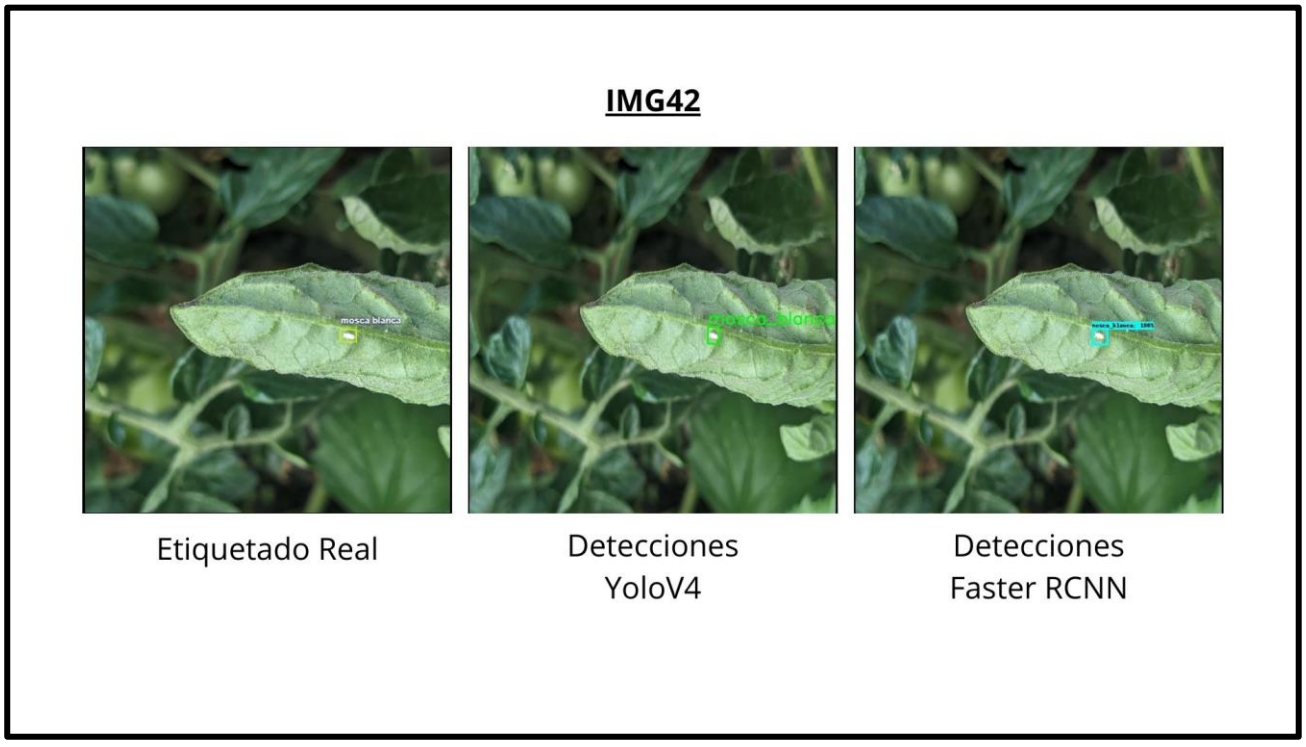


Figura 84 — Resultados de las detecciones IMG42

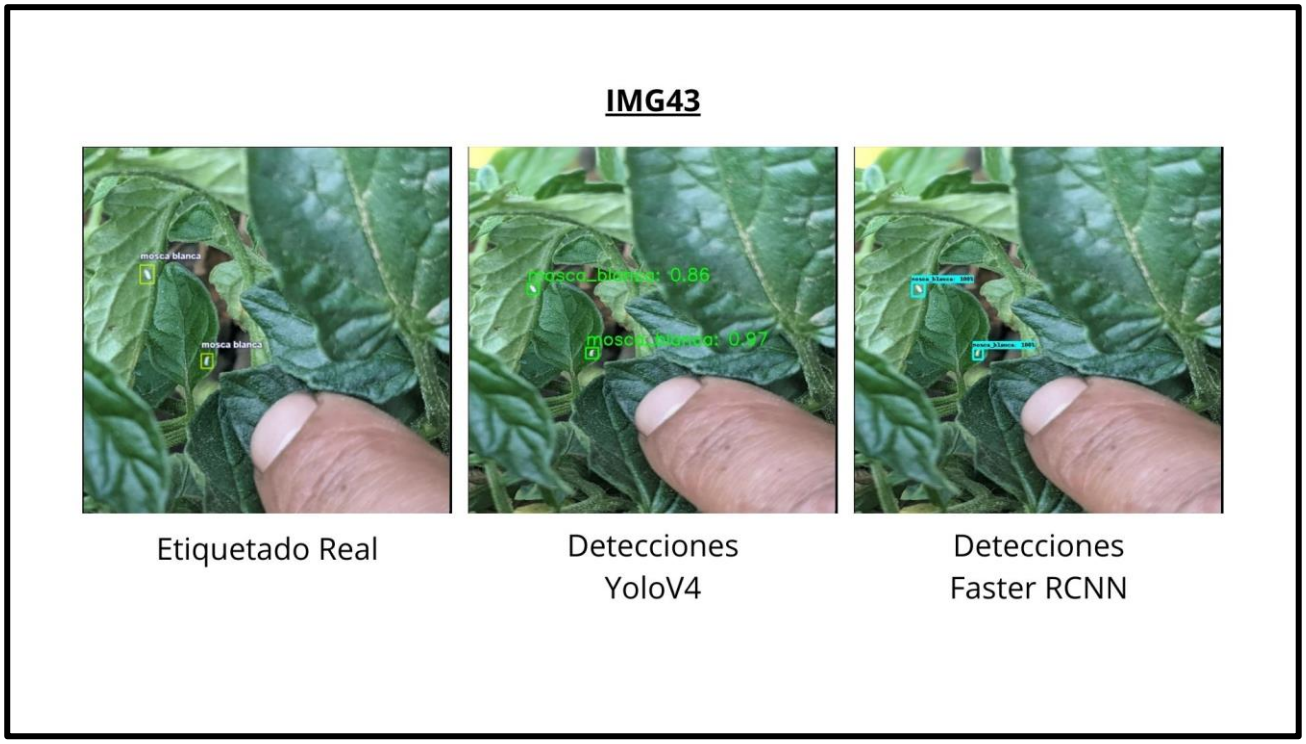


Figura 85 — Resultados de las detecciones IMG43

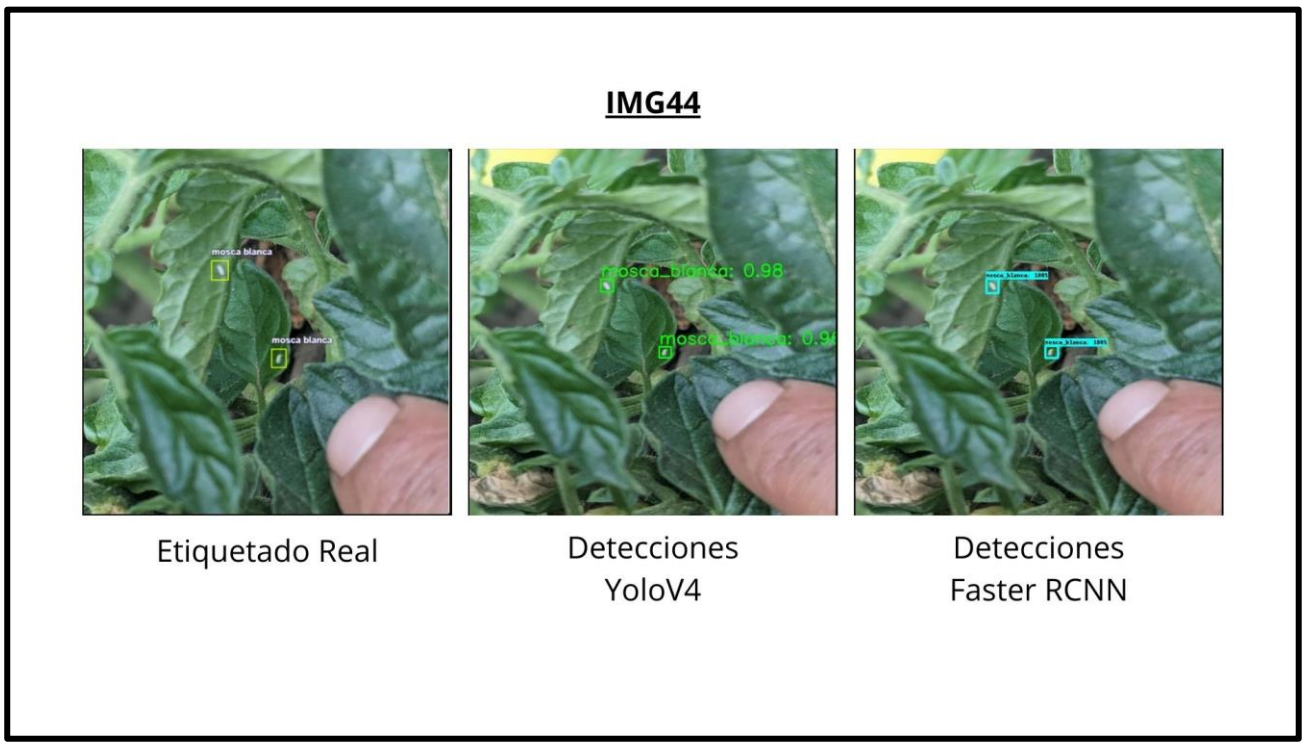


Figura 86 — Resultados de las detecciones IMG44

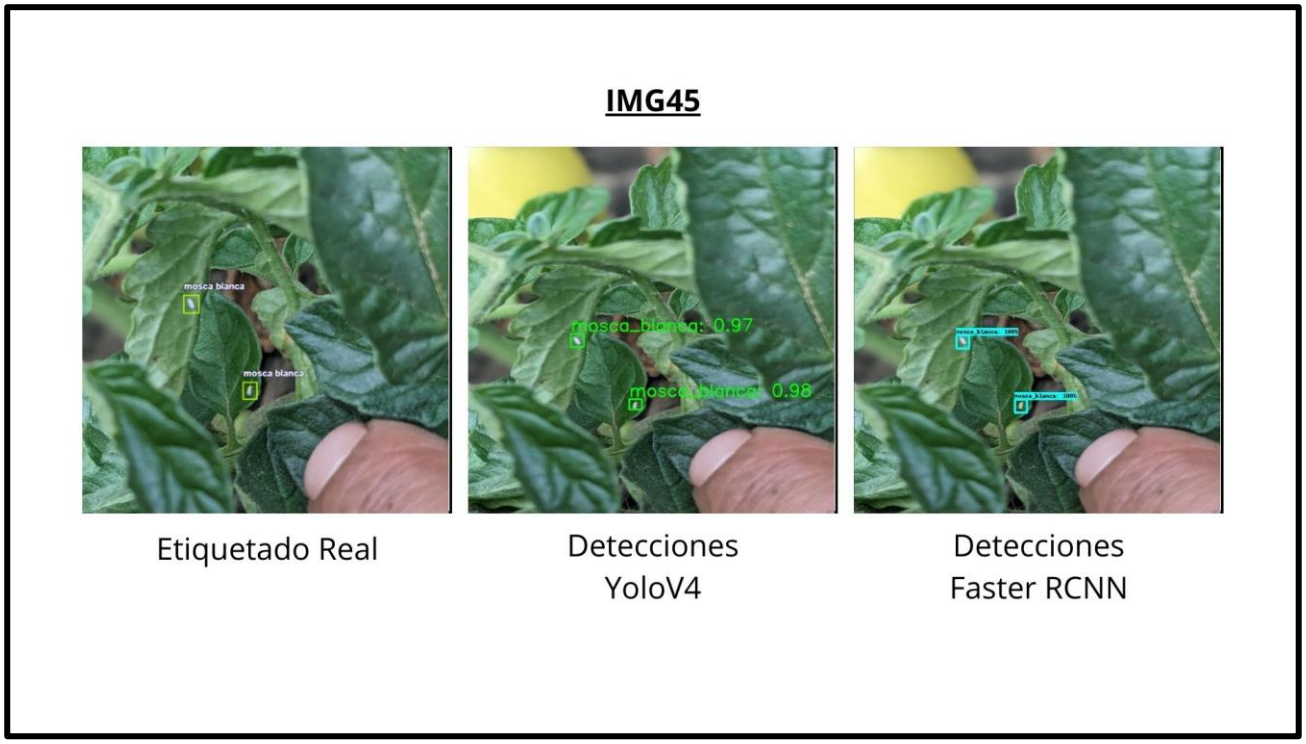


Figura 87 — Resultados de las detecciones IMG45

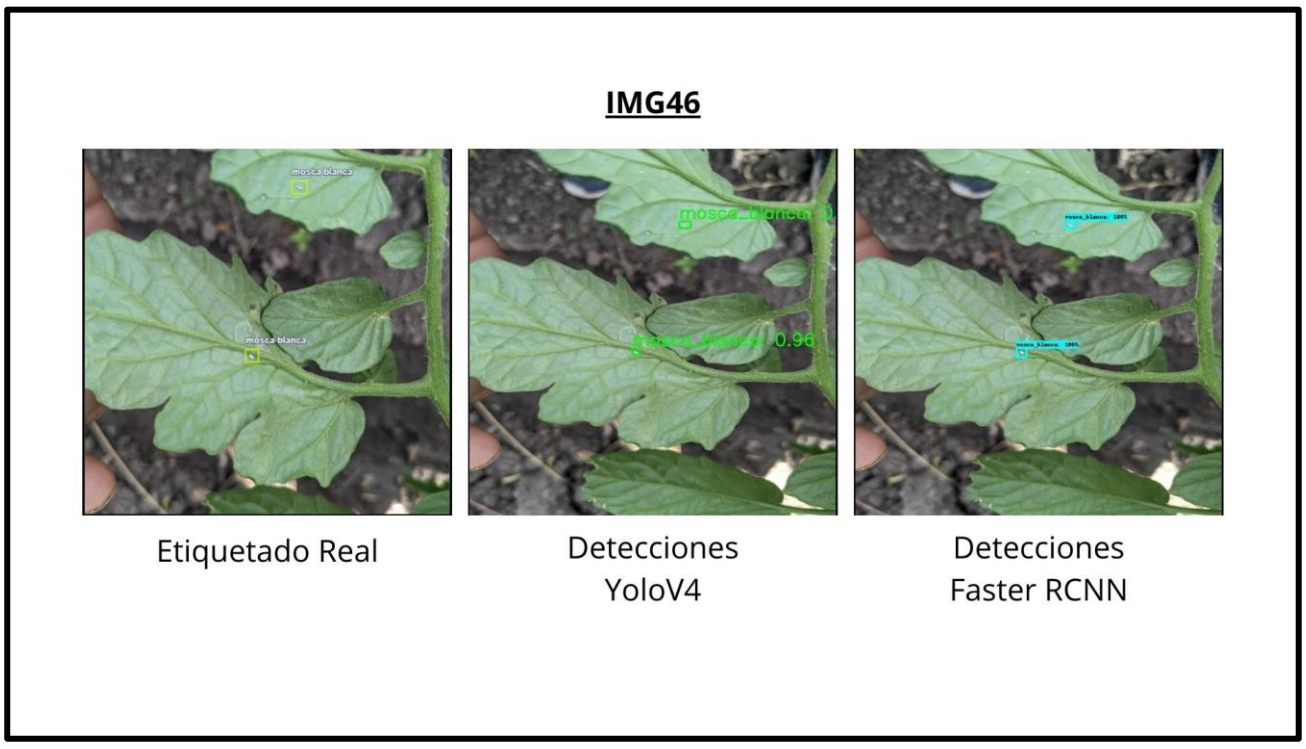


Figura 88 — Resultados de las detecciones IMG46

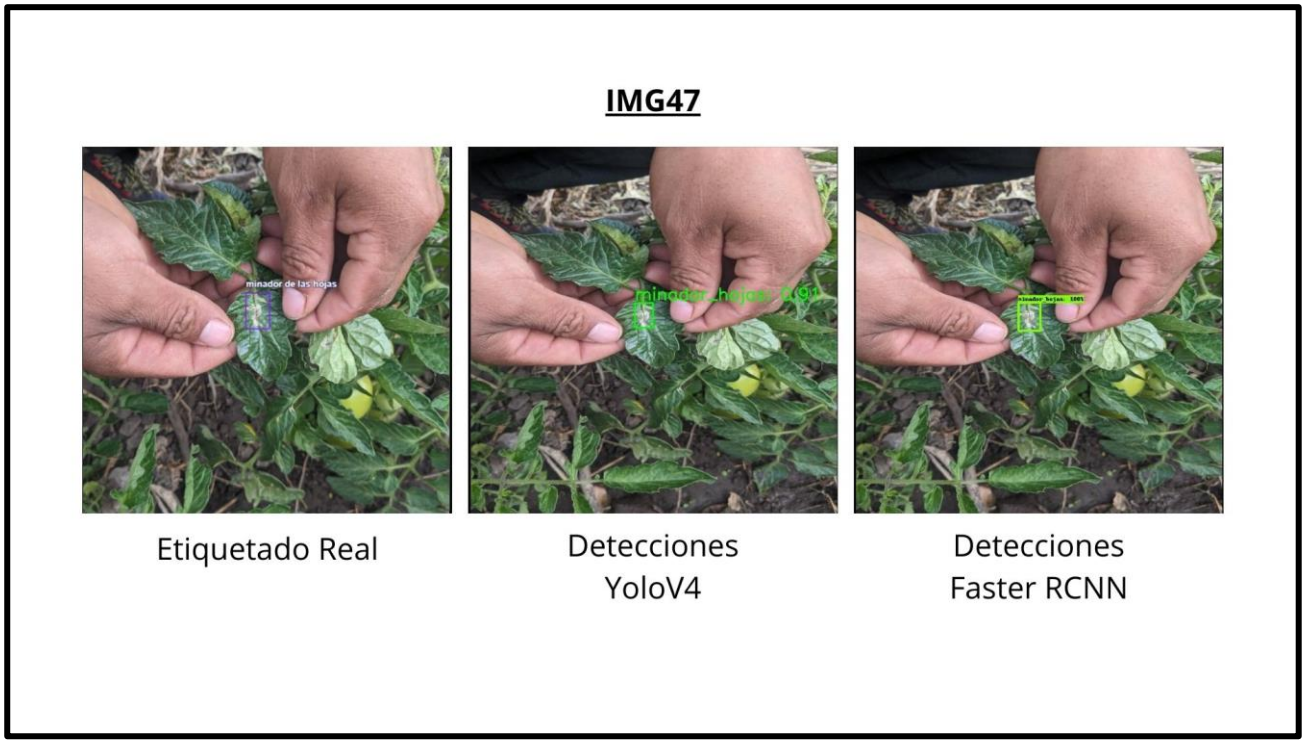


Figura 89 — Resultados de las detecciones IMG47

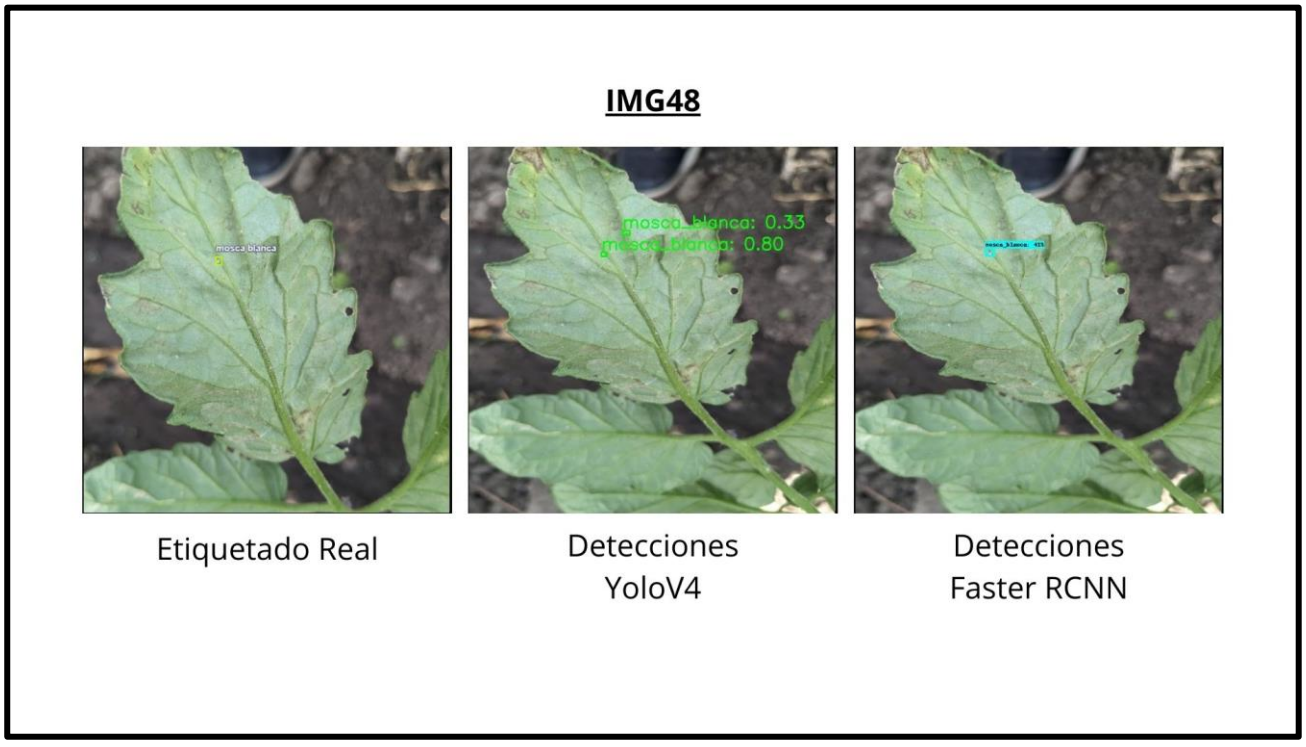


Figura 90 — Resultados de las detecciones IMG48

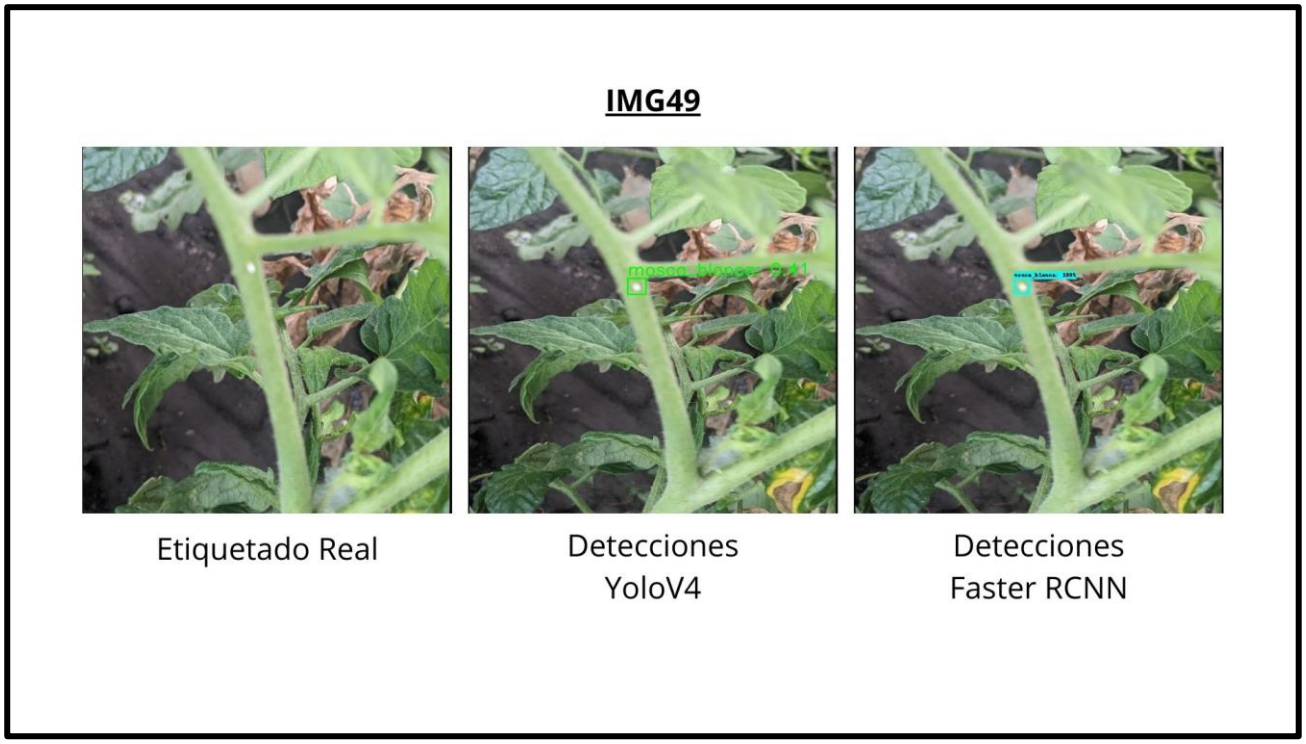


Figura 91 — Resultados de las detecciones IMG49

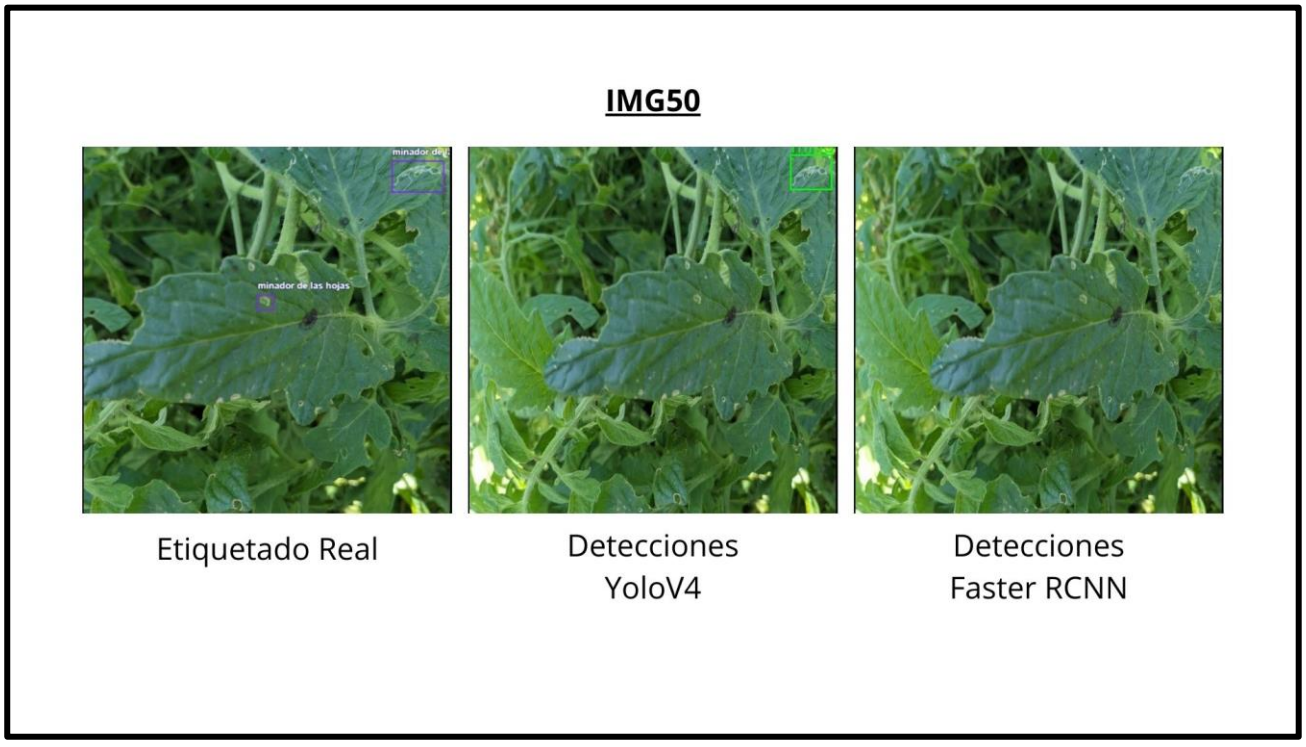


Figura 92 — Resultados de las detecciones IMG50

Anexo 06 — Códigos fuente del modelo de Yolo v4

```
# -*- coding: utf-8 -*-
"""tomatoe_model_train_yolo.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1rdf4CxvV66ynONhmELuUFVKCGh13TPPt

# Dependencies

## OpenCV
"""

pip install opencv-python==4.5.4.58

"""## Yolo V4 - darknet"""

!git clone https://github.com/AlexeyAB/darknet

# Commented out IPython magic to ensure Python compatibility.
# %cd darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile

# Commented out IPython magic to ensure Python compatibility.
# %%capture
# !make

"""# Utils

"""

# Commented out IPython magic to ensure Python compatibility.
def imShow(path):
  import cv2
  import matplotlib.pyplot as plt
#   %matplotlib inline

  image = cv2.imread(path)
  height, width = image.shape[:2]
  resized_image = cv2.resize(image,(3*width, 3*height), interpolation =
cv2.INTER_CUBIC)

  fig = plt.gcf()
  fig.set_size_inches(18, 10)
  plt.axis("off")
  plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
```

```
  plt.show()

# use this to upload files
def upload():
  from google.colab import files
  uploaded = files.upload()
  for name, data in uploaded.items():
    with open(name, 'wb') as f:
      f.write(data)
      print ('saved file', name)

# use this to download a file
def download(path):
  from google.colab import files
  files.download(path)

"""# Connect to Google Drive

## Mount
"""

base_path = '/content/drive/'
tomatoes_project_path = f'{base_path}/My Drive/Educacion/Tesis Unamba/Modelo-
Tomate/Yolo/'

from google.colab import drive
drive.mount(base_path)
!ls -l "{tomatoes_project_path}"

"""## Copy images & config files"""

!cp "{tomatoes_project_path}images_train/train.zip" ../
!cp "{tomatoes_project_path}images_train/test.zip" ../

# Commented out IPython magic to ensure Python compatibility.
# %%capture
# !unzip -o ../train.zip -d data/
# !unzip -o ../test.zip -d data/

!cp "{tomatoes_project_path}config/yolov4-obj.cfg" ./cfg
!cp "{tomatoes_project_path}config/obj.names" ./data
!cp "{tomatoes_project_path}config/obj.data"  ./data
!cp "{tomatoes_project_path}utils/generate_train.py" ./
!cp "{tomatoes_project_path}utils/generate_test.py" ./

"""# Pre - Train

## Generate text files with images data to train model
"""
```

```
!python generate_train.py
!python generate_test.py
!ls data/

"""# Train"""

# Commented out IPython magic to ensure Python compatibility.
# %%capture
# !wget
https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov
4.conv.137

"""## Entrenamiento inicial del modelo
En esta sección se lleva a cabo el entrenamiento del modelo. En esta fase es
importante considerar los siguientes puntos:


*   Se busca que el valor de pérdida **x, y avg loss ** tienda a cero.
*   Cada 100 pasos de entrenamiento se genera un archivo en la carpeta /backup.
Estos archivos se deben de estar descargando manualmente durante el entrenamiento.
* En la iteración 1000 se puede esperar tener un valor menor a 8, de no ser así,
puede haber mejoras al set de datos
* Se debe de asegurar de haber actualizado los valores en la carpeta /config
* Los archivos del modelo entrenado se sobre escriben por lo cual es importante
estarlos descargando y catalogando de manera periodica.



"""

!./darknet detector train data/obj.data cfg/yolov4-obj.cfg yolov4.conv.137 -
dont_show -map

"""## Entrenamiento incremental del modelo
Esta sección es específica a entrenar sobre un modelo que ya fue entrenado
anteriormente. Se requiere contar con el archivo **.weights** que se guardó en el
último entrenamiento del modelo.
"""

#!./darknet detector train data/obj.data cfg/yolov4-obj.cfg
backup/<nombre_del_archivo>.weights -dont_show
!./darknet detector train data/obj.data cfg/yolov4-obj.cfg backup/yolov4-
obj_v4_1400_l1.weights -dont_show

"""## Copy trained files
Asegurarse se descargar los archivos a la computadora para evitar perderlos. El
archivo final son todos aquellos con terminación .weigts
"""

!cp -r backup "{tomatoes_project_path}backup"
```

```
"""## Analyze trained file"""

!./darknet detector map data/obj.data cfg/yolov4-obj.cfg backup/yolov4-
obj_best.weights

imShow('/content/darknet/chart.png')

imShow('/content/darknet/chart_yolov4-obj.png')
```

Figura 93 — Código para el entrenamiento del modelo de Yolo v4

```
# -*- coding: utf-8 -*-
"""tomatoe_model_prediction_yolo.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1rdjjKtfHGW8n3F4cuH-ccUIdn7HTViZ4

# Dependencies

## OpenCV
"""

pip install opencv-python==4.5.4.58

"""# Connect to Google Drive

## Mount
"""

base_path = '/content/drive/'
tomatoes_project_path = f'{base_path}/My Drive/Educacion/Tesis Unamba/Modelo-Tomate/Yolo/'
trained_file = 'yolov4-obj_best.weights'
prediction_path = 'images_prediction'

from google.colab import drive
drive.mount(base_path)
!ls -l "{tomatoes_project_path}"

!cp -r "{tomatoes_project_path}{prediction_path}" .
!cp -r "{tomatoes_project_path}config" .
!cp -r "{tomatoes_project_path}backup" .
!mkdir -p "{tomatoes_project_path}images_evaluated"
!mkdir -p "{tomatoes_project_path}images_evaluated_csv"
!mkdir output

"""# Prediction"""

import cv2
import os
import pandas as pd
from google.colab.patches import cv2_imshow

"""## Load trained model"""

with open('config/obj.names', 'r') as f:
    classes = f.read().splitlines()
```

```
net = cv2.dnn.readNetFromDarknet('config/yolov4-obj.cfg',
f'backup/backup/{trained_file}')

model = cv2.dnn_DetectionModel(net)
model.setInputParams(scale=1 / 255, size=(416, 416), swapRB=True)

info = pd.DataFrame(columns=['Imagen','Deteccion','Numero_de_detecciones'])
print('init')
for archivo in os.listdir(f"./{prediction_path}"):
  img = cv2.imread(f"{prediction_path}/{archivo}")
  classIds, scores, boxes = model.detect(img, confThreshold=0.1, nmsThreshold=0.1)
  className = None
  for (classId, score, box) in zip(classIds, scores, boxes):
    cv2.rectangle(img, (box[0], box[1]), (box[0] + box[2], box[1] + box[3]),
                  color=(0, 255, 0), thickness=2)
    className = classes[classId]
    text = '%s: %.2f' % (className, score)
    cv2.putText(img, text, (box[0], box[1] - 5), cv2.FONT_HERSHEY_SIMPLEX, 1,
                color=(0, 255, 0), thickness=2)
    cv2.imwrite("output/{}".format(archivo),img)

  info.loc[len(info.index)] = [archivo, className, len(classIds)]

!cp -r /content/output "{tomatoes_project_path}images_evaluated/"

info

info.to_csv(f"{tomatoes_project_path}images_evaluated_csv/yolo_results.csv")
```

Figura 94 — Código para prueba del modelo de Yolo v4

Anexo 07 — Códigos fuente del modelo de Faster R-CNN

```
# -*- coding: utf-8 -*-
"""tomatoe_model_train_rcnn.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1v6bzQYZnNe366rHep94SUqLYP4QTTRYs

# Dependencies
"""

# Use last runtine colab
# Connect to your GPU and go to tools -> command palette -> search for use fallback
runtime version

"""## CUDA"""

!wget
https://developer.download.nvidia.com/compute/cuda/11.8.0/local_installers/cuda-
repo-ubuntu1804-11-8-local_11.8.0-520.61.05-1_amd64.deb

!dpkg -i cuda-repo-ubuntu1804-11-8-local_11.8.0-520.61.05-1_amd64.deb

!ls /var/cuda-repo-ubuntu1804-11-8-local | grep .pub

!apt-key add /var/cuda-repo-ubuntu1804-11-8-local/7D65C20C.pub

!apt-get update

!apt-get install cuda-11-8

!ls /usr/local/cuda-11.8

!export CUDA_PATH=/usr/local/cuda-11.8/

!nvcc --version

"""## Tensorflow"""

!pip install tensorflow[and-cuda]==2.13.0

!python3 -c "import tensorflow as tf; print(tf.reduce_sum(tf.random.normal([1000,
1000])))"

!python3 -c "import tensorflow as tf;
print(tf.config.list_physical_devices('GPU'))"

!git clone --depth 1 https://github.com/tensorflow/models
```

```
"""## Utils"""

#AUX: Solve problem version file

def copy_and_update_version():
  with open('/content/models/research/object_detection/packages/tf2/setup.py') as
f:
    content_list = f.readlines()

  content_list.insert(20, "'tensorflow==2.13.0',")

  with open('/content/models/research/setup.py', 'w+') as f:
    f.writelines(content_list)
  print('Updated!')

"""## Setup model"""

# Commented out IPython magic to ensure Python compatibility.
# %%bash
# cd models/research/
# pwd
# protoc object_detection/protos/*.proto --python_out=.

copy_and_update_version()

# Commented out IPython magic to ensure Python compatibility.
# %%capture
# %%bash
# cd models/research/
# pip install .

"""### Validate version & GPU"""

pip show tensorflow

import tensorflow as tf
tf.test.gpu_device_name()

"""## Test Model"""

!python
/content/models/research/object_detection/builders/model_builder_tf2_test.py

"""# Connect to Google Drive

## Mount
"""

base_path = '/content/drive/'
```

```
tomatoes_project_path = f'{base_path}/My Drive/Educacion/Tesis Unamba/Modelo-Tomate/Faster-RCNN/'

from google.colab import drive
drive.mount(base_path)
!ls -l "{tomatoes_project_path}"

"""## Copy images & config files"""

!cp "{tomatoes_project_path}images_train/train.zip" .
!cp "{tomatoes_project_path}images_train/test.zip" .
!cp -r "{tomatoes_project_path}config" .

# Commented out IPython magic to ensure Python compatibility.
# %%capture
# !unzip -o ./train.zip -d data/
# !unzip -o ./test.zip -d data/

!mkdir ./scripts
!cp "{tomatoes_project_path}utils/generate_tfrecord.py" ./scripts/

"""# Pre-train

## Generate text files with images data to train model
"""

!python ./scripts/generate_tfrecord.py -x ./data/train -l ./config/label_map.pbtxt -o ./config/train.record
!python ./scripts/generate_tfrecord.py -x ./data/test -l ./config/label_map.pbtxt -o ./config/test.record

"""## Download Model"""

!wget http://download.tensorflow.org/models/object_detection/tf2/20200711/faster_rcnn_resnet101_v1_640x640_coco17_tpu-8.tar.gz

"""## Unzip model"""

!mkdir pre-trained-models
!tar -zxf  faster_rcnn_resnet101_v1_640x640_coco17_tpu-8.tar.gz --directory ./pre-trained-models/

#Copy main TF
!cp /content/models/research/object_detection/model_main_tf2.py .

"""# Train"""

model_dir = 'config/models/faster_rcnn_resnet101_v1'
pipeline_config_path = 'config/models/faster_rcnn_resnet101_v1/pipeline.config'
```

```
!python /content/models/research/object_detection/model_main_tf2.py \
  --model_dir={model_dir}\
  --pipeline_config_path={pipeline_config_path} \
  --alsologtostderr

"""## Evaluating"""

# Commented out IPython magic to ensure Python compatibility.
# %load_ext tensorboard
# %tensorboard --logdir {model_dir}

tomatoes_project_path

#Backup model trained
!pwd
!cp -r {model_dir} "{tomatoes_project_path}/backup"

#Evaluating model
!python model_main_tf2.py --model_dir={model_dir} --
pipeline_config_path={model_dir}/pipeline.config --checkpoint_dir={model_dir}

"""# Export Model"""

!cp /content/models/research/object_detection/exporter_main_v2.py .

!python exporter_main_v2.py --input_type image_tensor --pipeline_config_path
{pipeline_config_path} --trained_checkpoint_dir {model_dir} --output_directory
exported-models/faster_rcnn_resnet101_v1

!mkdir -p "{tomatoes_project_path}exported-models/faster_rcnn_resnet101_v1"

# Commented out IPython magic to ensure Python compatibility.
#SAVE EXPORTED MODEL
# %cd /content
!pwd
!cp -r exported-models/faster_rcnn_resnet101_v1 "{tomatoes_project_path}exported-
models/faster_rcnn_resnet101_v1"
```

Figura 95 — Código para el entrenamiento del modelo de Faster R-CNN

```
# -*- coding: utf-8 -*-
"""tomatoe_model_prediction_rcnn.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1v6wisnVKv_hPqkCUK072wkSg-FF_GJDm

# Dependencies

## Tensorflow
"""

# Commented out IPython magic to ensure Python compatibility.
# %%capture
# !pip install tensorflow[and-cuda]==2.13.0

!git clone --depth 1 https://github.com/tensorflow/models

"""## Utils"""

#AUX: Solve problem version file

def copy_and_update_version():
  with open('/content/models/research/object_detection/packages/tf2/setup.py') as
f:
    content_list = f.readlines()

  content_list.insert(20, "'tensorflow==2.13.0',")

  with open('/content/models/research/setup.py', 'w+') as f:
    f.writelines(content_list)
  print('Updated!')

"""## Setup model"""

# Commented out IPython magic to ensure Python compatibility.
# %%bash
# cd models/research/
# pwd
# protoc object_detection/protos/*.proto --python_out=.

copy_and_update_version()

# Commented out IPython magic to ensure Python compatibility.
# %%capture
# %%bash
# cd models/research/
# pip install .
```

```
"""### Validate version & GPU"""

pip show tensorflow

import tensorflow as tf
tf.test.gpu_device_name()

"""# Connect to Google Drive

## Mount
"""

base_path = '/content/drive/'
tomatoes_project_path = f'{base_path}/My Drive/Educacion/Tesis Unamba/Modelo-
Tomate/Faster-RCNN/'

from google.colab import drive
drive.mount(base_path)
!ls -l "{tomatoes_project_path}"

"""## Copy images & config files"""

!cp -r "{tomatoes_project_path}images_prediction" .
!cp -r "{tomatoes_project_path}exported-
models/faster_rcnn_resnet101_v1/faster_rcnn_resnet101_v1" .
!cp -r "{tomatoes_project_path}config" .

"""#Test"""

# Commented out IPython magic to ensure Python compatibility.
# %matplotlib inline

# Commented out IPython magic to ensure Python compatibility.
# %cd /content/models/research/

import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'    # Suppress TensorFlow logging (1)
import pathlib
import tensorflow as tf

import time
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils

tf.get_logger().setLevel('ERROR')           # Suppress TensorFlow logging (2)

# Enable GPU dynamic memory allocation
gpus = tf.config.experimental.list_physical_devices('GPU')
for gpu in gpus:
```

```
    tf.config.experimental.set_memory_growth(gpu, True)

PATH_TO_TEST_IMAGES_DIR = pathlib.Path('/content/images_prediction/')
IMAGE_PATHS = sorted(list(PATH_TO_TEST_IMAGES_DIR.glob("*.jpg")))
# IMAGE_PATHS

MODEL_PATH = '/content/faster_rcnn_resnet101_v1'
LABEL_PATH = '/content/config/label_map.pbtxt'

#Load Model

PATH_TO_SAVED_MODEL = MODEL_PATH + "/saved_model"

print('Loading model...', end='')
start_time = time.time()

# Load saved model and build the detection function
detect_fn = tf.saved_model.load(PATH_TO_SAVED_MODEL)

end_time = time.time()
elapsed_time = end_time - start_time
print('Done! Took {} seconds'.format(elapsed_time))

category_index = label_map_util.create_category_index_from_labelmap(LABEL_PATH,
                                                                    use_display_nam
e=True)
category_index

!mkdir /content/images_prediction_evaluated
!mkdir -p "{tomatoes_project_path}images_prediction_evaluated"

!mkdir -p "{tomatoes_project_path}images_prediction_evaluated_csv"

from collections import defaultdict

def getClassesDetectedByImage(
    image,
    boxes,
    classes,
    scores,
    category_index,
    min_score_thresh=.5
    ):
  classesDetected = []

  for i in range(boxes.shape[0]):

    if scores is None or scores[i] > min_score_thresh:
      box = tuple(boxes[i].tolist())
```

```
      if scores is None:
        raise ValueError("scores should not be None")
      else:
        class_name = category_index[classes[i]]['name']
        display_str = str(class_name)
        # if not skip_scores:
        #   if not display_str:
        #     display_str = '{}%'.format(round(100*scores[i]))
        #   else:
        #     display_str = '{}: {}%'.format(display_str, round(100*scores[i]))

        classesDetected.append((image, display_str))

  counts = defaultdict(int)
  for _, label in classesDetected:
      counts[(image, label)] += 1

  result = [(image_name, label, count) for (image_name, label), count in
counts.items()]

  return result

import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
import matplotlib.image as mpimg  # Importar matplotlib.image para guardar la
imagen
import warnings
warnings.filterwarnings('ignore')   # Suprimir advertencias de Matplotlib

def load_image_into_numpy_array(path):
    return np.array(Image.open(path))

info_data = []

for image_path in IMAGE_PATHS:

    print('Running inference for {}... '.format(image_path), end='')

    image_np = load_image_into_numpy_array(image_path)

    input_tensor = tf.convert_to_tensor(image_np)
    input_tensor = input_tensor[tf.newaxis, ...]

    detections = detect_fn(input_tensor)

    num_detections = int(detections.pop('num_detections'))
    detections = {key: value[0, :num_detections].numpy()
                  for key, value in detections.items()}
    detections['num_detections'] = num_detections
```

```
    # Las clases de detección deben ser enteros.
    detections['detection_classes'] =
detections['detection_classes'].astype(np.int64)

    image_np_with_detections = image_np.copy()

    viz_utils.visualize_boxes_and_labels_on_image_array(
          image_np_with_detections,
          detections['detection_boxes'],
          detections['detection_classes'],
          detections['detection_scores'],
          category_index,
          use_normalized_coordinates=True,
          max_boxes_to_draw=200,
          min_score_thresh=.30,
          agnostic_mode=False)

    # Guardar la imagen modificada
    info_data += getClassesDetectedByImage(
          image_path.name,
          detections['detection_boxes'],
          detections['detection_classes'],
          detections['detection_scores'],
          category_index,
          min_score_thresh=.30)

    output_image_path = str(image_path).replace('images_prediction',
'images_prediction_evaluated')
    mpimg.imsave(output_image_path, image_np_with_detections)  # Guardar la imagen
modificada

    print('Done')

!cp -r /content/images_prediction_evaluated
"{tomatoes_project_path}images_prediction_evaluated/"

import pandas as pd
info = pd.DataFrame(info_data, columns=['Image', 'Deteccion',
'Numero_de_detecciones'])

info

info.to_csv(f"{tomatoes_project_path}images_prediction_evaluated_csv/rcnn_results.c
sv")
```

Figura 96 — Código para prueba del modelo de Faster R-CNN