UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE GUAYAQUIL

CARRERA DE MECATRÓNICA

DESARROLLO DE UN ALGORITMO DE VISIÓN ARTIFICIAL PARA LA CLASIFICACIÓN DE LAS ESPECIES DE PATOS DENDROCYGNA BICOLOR Y DENDROCYGNA AUTUMNALIS EN ZONAS AEROPORTUARIAS DE GUAYAQUIL

Trabajo de titulación previo a la obtención del
Título de Ingeniero en Mecatrónica

AUTORES: Anibal Sebastian Macero Guerrero
Jeremy Aaron Ronquillo Siguencia

TUTOR: Ing. Jonathan Salvador Paillacho Corredores

Guayaquil - Ecuador
2025

20/08/25
19:00

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, **Anibal Sebastian Macero Guerrero** con documento de identificación Nº **0929181238** y **Jeremy Aaron Ronquillo Siguencia** con documento de identificación Nº **0926569104**; manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que, sin fines de lucro, la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo.

Guayaquil, 20 de agosto del año 2025

Atentamente,

Anibal Sebastian Macero Guerrero
0929181238

Jeremy Aaron Ronquillo Siguencia
0926569104

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, **Anibal Sebastian Macero Guerrero** con documento de identificación Nº **0929181238** y **Jeremy Aaron Ronquillo Siguencia** con documento de identificación Nº **0926569104**, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del **Dispositivo Tecnológico: DESARROLLO DE UN ALGORITMO DE VISIÓN ARTIFICIAL PARA LA CLASIFICACIÓN DE LAS ESPECIES DE PATOS DENDROCYGNA BICOLOR Y DENDROCYGNA AUTUMNALIS EN ZONAS AEROPORTUARIAS DE GUAYAQUIL**, el cual ha sido desarrollado para optar por el título de: Ingeniero en Mecatrónica, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Guayaquil, 20 de agosto del año 2025

Atentamente,

Anibal Sebastian Macero Guerrero
0929181238

Jeremy Aaron Ronquillo Siguencia
0926569104

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, **Jonathan Salvador Paillacho Corredores**, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: **DESARROLLO DE UN ALGORITMO DE VISIÓN ARTIFICIAL PARA LA CLASIFICACIÓN DE LAS ESPECIES DE PATOS DENDROCYGNA BICOLOR Y DENDROCYGNA AUTUMNALIS EN ZONAS AEROPORTUARIAS DE GUAYAQUIL**, realizado por **Anibal Sebastian Macero Guerrero** con documento de identificación Nº **0929181238** y por **Jeremy Aaron Ronquillo Siguencia** con documento de identificación Nº **0926569104**, obteniendo como resultado final el trabajo de titulación bajo la opción **Dispositivo Tecnológico** que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Guayaquil, 20 de agosto del año 2025

Atentamente,

Ing. Jonathan Salvador Paillacho Corredores
1718907874

DEDICATORIA

Este trabajo de titulación está dedicado principalmente a mi familia, a mi padre Lauro Anibal Macero Campoverde ya que fue un pilar fundamental en mi desarrollo personal y educativo, agradezco su disposición a siempre brindarme su apoyo, a mi madre Yanina Magdalena Guerrero Delgado ya que ha sido una compañía durante todo mi proceso universitario, brindándome su apoyo incondicional, siendo la mayor influencia en mi desarrollo como un ser humano de bien, enseñándome a seguir avanzando a pesar de los obstáculos y las circunstancias.

Por último, dedico este proyecto de titulación a cada uno de los estudiantes que siguen avanzando en su preparación profesional, principalmente a los futuros Ingenieros en Mecatrónica, de los cuales en algún futuro espero poderles ser de ayuda y que ellos puedan ser de ayuda para los demás. Espero que nuestra comunidad pueda crecer y que, con esfuerzo, podamos demostrar la valía de nuestra carrera en la industria y, sobre todo, para el desarrollo de nuestro amado país, Ecuador.

Anibal Sebastian Macero Guerrero

A mi mamá, Iliane Siguencia, pilar de mi vida, por su presencia inquebrantable, su amor que todo lo sostiene y el ejemplo que guía cada paso. A mi tía, Irene Muñoz, cuyo apoyo ha cruzado siempre la distancia y cuya fortaleza y generosidad admiro profundamente. A Mirella Moya, figura materna fundamental, por criarme con tanta ternura y enseñarme a mirar el mundo con respeto y gratitud.

Y a mí mismo, por no rendirme cuando el camino se volvió cuesta arriba, por volver a intentarlo cada vez, por la disciplina silenciosa y la constancia que me trajeron hasta aquí. Este logro les pertenece a ustedes y también a la parte de mí que decidió creer y seguir.

Jeremy Aaron Ronquillo Siguencia

AGRADECIMIENTO

Agradezco en primer lugar a Dios que me dio la oportunidad de seguir avanzando de una manera exitosa cada uno de los niveles cursados en la Universidad Politécnica Salesiana, a mis padres que con sus enseñanzas y con su apoyo me permitieron seguir avanzando en mi preparación educativa desde muy pequeño y hasta ahora en la universidad para ser un futuro profesional.

Agradezco a la Universidad Politécnica Salesiana, a los docentes que nos brindaron parte del conocimiento necesario, siempre siendo una guía y un ejemplo para cada uno de sus estudiantes, les agradezco cada reto y cada oportunidad que nos brindaron para desarrollar un carácter profesional. Agradezco de gran manera a nuestro tutor Jonathan Salvador Paillacho Corredores, por brindarnos su apoyo, y sobre todo por la paciencia dispuesta hacia nosotros durante todo este proceso de titulación, espero que siga guiando e influyendo en el futuro de muchos otros estudiantes que buscan desarrollarse en su futuro profesional.

Por último quiero agradecer de gran manera a mi compañero de titulación Jeremy Aaron Ronquillo Siguencia, por ser un principal apoyo durante este proyecto de titulación, agradezco la paciencia y la responsabilidad invertida en nuestro proyecto, espero que su futuro como profesional sea igual de productivo como cada una de las horas invertidas en nuestro Proyecto de Titulación, además espero que siga desarrollándose profesionalmente demostrando que es capaz de lograr grandes objetivos .

Anibal Sebastian Macero Guerrero

A Dios, por la fuerza, la salud y la claridad para llegar hasta aquí. A mi familia y, en especial, a mis padres, por su apoyo incondicional, su paciencia en los días largos y la confianza que me sostuvo en cada etapa. A mi pareja, por el ánimo constante, la comprensión y la compañía que hicieron más llevadero este camino.

A la Universidad Politécnica Salesiana, a sus docentes, y a los compañeros y amigos que conocí a lo largo de este proceso, gracias por las clases, las conversaciones, los trabajos compartidos y los buenos momentos que dieron forma a esta experiencia académica y humana.

A mi tutor, Ing. Jonathan Salvador Paillacho Corredores, por su guía oportuna, sus observaciones claras y la orientación que permitió encaminar el proyecto hacia la meta con orden y propósito.

A todos quienes, de una u otra manera, brindaron tiempo, recursos o palabras de aliento, gracias por ser parte de este logro.

Jeremy Aaron Ronquillo Siguencia

RESUMEN

El presente proyecto diseña e implementa un sistema de visión por computador para detectar dos especies de patos silbadores, Dendrocygna autumnalis y Dendrocygna bicolor, orientado al apoyo del monitoreo en entornos sensibles. Se construyó un conjunto de datos a partir de imágenes públicas obtenidas en la web, se realizó la etiquetación por cada especie usando el programa LabelImg y se organizó en carpetas haciendo particiones para entrenamiento, validación y prueba.

Con esta base se entrenó un modelo ligero YOLOv7 tiny en Google Colab mediante transferencia de aprendizaje y se exportó a ONNX para su ejecución en una Raspberry Pi conectada a una webcam Full HD compatible con UVC. La solución integra, adquisición de imágenes, inferencia y visualización de las especies de patos mediante cajas delimitadoras que son proyectadas en un display, con baja latencia y operación estable.

El proyecto aporta un pipeline reproducible de principio a fin, con modelo YOLOv7-tiny exportado a ONNX y desplegado en Raspberry Pi con webcam Full HD, junto con scripts y pautas de calibración que permiten instalar, operar y extender la solución con rapidez. El sistema mantiene baja latencia, visualiza detecciones en tiempo real y deja sentadas buenas prácticas de datos para futuras ampliaciones.

Palabras claves: Visión por computador, clasificación de patos, YOLOv7 tiny, Raspberry Pi, ONNX, LabelImg.

ABSTRACT

This project designs and implements a computer-vision system to detect two whistling-duck species, Dendrocygna autumnalis and Dendrocygna bicolor, aimed at supporting monitoring in sensitive environments. A dataset was built from public images gathered on the web, labeled by species with LabelImg, and organized into training, validation, and test partitions.

On this basis, a lightweight YOLOv7-tiny model was trained in Google Colab via transfer learning and exported to ONNX for execution on a Raspberry Pi connected to a UVC-compatible Full HD webcam. The solution integrates image acquisition, inference, and display visualization through bounding boxes, achieving low latency and stable operation.

The system exhibits robust behavior across diverse scenes and a clear separation between species. As a practical contribution, the work provides a reproducible end-to-end pipeline, device-ready inference scripts, and calibration guidelines that facilitate field use and extension to new scenarios.

Keywords: Computer vision, duck classification, YOLOv7-tiny, Raspberry Pi, ONNX, LabelImg.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

I. Introducción

El presente estudio tiene como fin el diseño e implementación de un sistema de visión por computador para la detección de dos especies de patos silbadores, Dendrocygna autumnalis y Dendrocygna bicolor, con aplicación en tareas de monitoreo en tiempo cercano al real. La solución opera sobre una Raspberry Pi conectada a una webcam Full HD compatible con UVC y muestra en pantalla las cajas delimitadoras y la etiqueta de especie en cada fotograma, de manera continua y estable.

Para sustentar el modelo, se construye un conjunto de datos a partir de imágenes públicas obtenidas en la web. El material se etiqueta por especie con la herramienta LabelImg y se organiza en particiones de entrenamiento, validación y prueba, manteniendo un flujo reproducible y coherente entre la anotación y el uso final.

El entrenamiento se realiza en Google Colab mediante transfer learning sobre la variante YOLOv7 tiny, priorizando un equilibrio entre precisión y rapidez. Una vez ajustado, el modelo se exporta a ONNX para su ejecución eficiente en el dispositivo. La arquitectura integra adquisición, inferencia y visualización: la Raspberry Pi recibe el flujo de la cámara, procesa las imágenes y superpone las detecciones sobre la salida de video, preservando una latencia reducida adecuada para monitoreo continuo.

El documento describe el procedimiento completo, creación del dataset, etiquetado, configuración del archivo de datos, entrenamiento, evaluación y despliegue en dispositivo, presentando lineamientos de calibración del umbral de confianza y de organización de resultados. Con ello, se ofrece una solución integrada, reproducible y lista para extenderse a nuevos escenarios o fuentes de video según los requerimientos de operación.

II. Problema

El impacto entre aves y aeronaves representa una problemática crítica para la aviación civil y comercial, debido a las consecuencias que genera a nivel económico, ambiental y, especialmente, en la seguridad de los pasajeros y tripulaciones [1]. Desde el punto de vista económico, las colisiones con aves, también conocidas como bird strikes [2], ocasionan pérdidas millonarias cada año a nivel mundial. Las aerolíneas se ven obligadas a afrontar inspecciones no programadas, reparaciones estructurales, sustitución de componentes dañados (como motores, trenes de aterrizaje o sensores), así como la reprogramación de itinerarios, desvíos de rutas y cancelaciones de vuelos [3].

Según la Administración Federal de Aviación (FAA) [4], solo en Estados Unidos se reportan más de 16,000 impactos con fauna silvestre al año, con un costo estimado de más de 300 millones de dólares anuales para la industria [5]. Estos eventos no solo afectan la operatividad de las aerolíneas, sino también la reputación de los aeropuertos, al provocar retrasos masivos y pérdida de confianza por parte de los usuarios [6].

En la ciudad de Guayaquil, Ecuador, la presencia constante de aves en las inmediaciones del Aeropuerto Internacional José Joaquín de Olmedo se ha convertido en un factor de riesgo recurrente para las operaciones aéreas [7]. Específicamente, zonas como el islote El Palmar [8], caracterizadas por su riqueza ecológica y cercanía a las rutas de aproximación, han sido identificadas como puntos críticos de concentración de fauna silvestre [9]. Las especies Dendrocygna bicolor y Dendrocygna autumnalis [10], comúnmente conocidas como patillos, forman bandadas que frecuentan cuerpos de agua y humedales cercanos, desplazándose en horarios que coinciden con las franjas de mayor tráfico aéreo. Esta superposición entre actividad biológica y operación aeroportuaria ha ocasionado incidentes documentados, como los reportados por la aerolínea Equair [11], que informó impactos con aves durante maniobras de despegue y aterrizaje, lo cual motivó solicitudes formales de intervención a las autoridades aeronáuticas.

A pesar de estos antecedentes, la gestión actual del riesgo faunístico en el aeropuerto se limita a mecanismos reactivos y esporádicos, sin un sistema de monitoreo continuo ni una base de datos que permita caracterizar con precisión la dinámica poblacional de las especies involucradas [12]. La ausencia de un control sistemático dificulta la formulación e implementación de estrategias preventivas eficaces y mantiene vigente un riesgo que combina seguridad operacional, efectos ambientales y costos logísticos elevados para las aerolíneas [13]. El incremento de los encuentros entre aves silvestres [14] y aeronaves refuerza la necesidad de incorporar soluciones tecnológicas capaces de detectar, clasificar y registrar su presencia en tiempo real, con el fin de apoyar decisiones informadas y disminuir la probabilidad de colisión.

III. JUSTIFICACIÓN

En las últimas décadas, el progreso de las tecnologías digitales ha transformado de manera profunda la forma de vigilar los entornos naturales. Herramientas como la visión por computadora [15], el aprendizaje automático y los sistemas embebidos han permitido automatizar tareas antes dependientes de la observación humana, con mejoras notables en la precisión, la rapidez de respuesta y la trazabilidad de la información [16]. En el campo ambiental, estas soluciones se han aplicado con éxito al seguimiento de fauna silvestre, posibilitando la identificación y el monitoreo de animales en hábitats complejos o de difícil acceso, sin alterar su comportamiento natural [17].

Frente a las limitaciones de la observación humana, un sistema de visión por computador para el seguimiento de las especies Dendrocygna bicolor y Dendrocygna autumnalis ofrece mayor regularidad y objetividad en la detección de presencia y actividad [18]. Al automatizar el proceso, se reduce la dependencia de operadores cuyas capacidades pueden verse mermadas por la fatiga, las condiciones ambientales o la confusión entre especies de rasgos similares [19]. Entrenado con imágenes reales obtenidas en campo, el modelo aprende los rasgos distintivos de cada especie y ejecuta tareas de localización en tiempo real, habilitando alertas y decisiones oportunas por parte de las autoridades aeroportuarias ante concentraciones de aves en zonas críticas [20].

Desde el punto de vista económico, la reducción de incidentes con fauna implica menos inspecciones de emergencia y reparaciones costosas, así como una disminución de los retrasos y cancelaciones de vuelos [21]. En aeropuertos con alta incidencia de colisiones, se estima que la implementación de soluciones basadas en visión artificial puede reducir los gastos asociados en un 20 % o más al año [22]. Además, al optimizar la asignación de personal y enfocar los recursos humanos en tareas de mantenimiento y respuesta puntual, se logra un uso más racional del presupuesto disponible.

En lo ambiental, el sistema propuesto promueve un método no invasivo de gestión de la fauna [23]. A diferencia de las trampas, el uso intensivo de ultrasonido o la perturbación con métodos físicos que pueden alterar los ritmos de alimentación y descanso de las aves, la visión artificial respeta el comportamiento natural de las poblaciones silvestres [24]. Asimismo, la base de datos histórica generada permitirá a biólogos y autoridades ambientales estudiar las dinámicas migratorias y reproductivas de las especies, identificar posibles causas de fluctuaciones poblacionales y planificar corredores ecológicos o áreas de alimentación alternativas alejadas de las rutas de vuelo.

Finalmente, la flexibilidad del sistema garantiza su escalabilidad y replicabilidad. Reentrenando el modelo con nuevos conjuntos de datos, se puede adaptar la solución a otras especies de aves o entornos críticos [25], como puertos marítimos [26] o instalaciones industriales. La arquitectura modular también permite migrar el procesamiento entre microcontroladores de bajo consumo y servidores en la nube, según las restricciones económicas y de infraestructura. De este modo, el proyecto no solo atiende un problema concreto en Guayaquil, sino que sienta las bases para un marco de referencia en la gestión inteligente de fauna en toda América Latina, fortaleciendo la seguridad aérea, promoviendo la conservación y demostrando el potencial transformador de la inteligencia artificial en la ingeniería mecatrónica.

IV. OBJETIVOS

IV-A. Objetivo general

Desarrollar un algoritmo basado en visión artificial para la clasificación automatizada de las especies de patos Dendrocyna bicolor y Dendrocyna autumnalis, con el fin de contribuir a su control en zonas aeroportuarias.

IV-B. Objetivos específicos

1. Construir una base de datos de imágenes representativas de las especies Dendrocyna bicolor y Dendrocyna autumnalis, considerando condiciones reales de las zonas aeroportuarias.

2. Preprocesar las imágenes recopiladas y desarrollar un modelo de clasificación utilizando técnicas de visión artificial.

3. Validar el rendimiento y la precisión del algoritmo de clasificación mediante pruebas controladas.

V. Marco Teórico

V-A. Medios de transporte

El transporte aéreo desempeña un papel fundamental en la economía global y regional, destacándose por su capacidad para conectar rápidamente distancias largas y facilitar el comercio internacional. Según la Asociación Internacional de Transporte Aéreo (IATA), el sector genera aproximadamente 65,5 millones de empleos y contribuye con 2,7 billones de dólares al producto interno bruto mundial . Además, el 51 por ciento de los turistas internacionales viajan en avión, lo que resalta su importancia en la industria del transporte [27]. En América Latina y el Caribe, la aviación actúa como un habilitador central del desarrollo sostenible: amplía la conectividad entre territorios a escala regional y global, una forma que dinamiza la actividad económica [28].

Tipo	Subsector	2008	2009	2010	2011	2012	2013	2014	2015
Privada	Aéreo	231	40	169	369	2 839	29	4 559	139
	Carreteras	7 656	7 244	4 469	4 229	3 553	8 705	20 978	12 860
	Ferrocarril	1 558	685	2 241	3 933	5 653	6 731	4 979	7 340
	Acuático	2 564	2 374	1 240	2 684	1 351	3 208	184	1 418
Pública	Aéreo	511	561	909	1 390	1 451	1 768	2 100	2 001
	Carreteras	19 381	27 482	32 760	32 505	32 522	37 685	36 487	27 241
	Ferrocarril	1 789	2 090	3 579	3 133	2 927	2 535	2 896	2 927
	Acuático	1 044	1 768	1 834	1 932	2 124	2 388	2 288	2 209
	ND	2 728	3 657	4 434	4 716	5 209	4 621	4 324	4 514
Grand Total		37 461	45 900	51 635	54 891	57 627	67 671	78 795	60 648

Figura 1. Inversión de transportes [28].

Según la Comisión Económica para América Latina, el fortalecimiento de este sector resulta decisivo para enfrentar desafíos persistentes, entre ellos los problemas de seguridad del transporte terrestre y las ineficiencias de los procesos logísticos que restringen la competitividad regional [29].

Figura 2. Industria móvil [27].

V-A1. Avión: El avión es uno de los medios de transporte más usados en la sociedad debido a su rapidez y a su gran capacidad para transportar grandes cantidades de personas; hay varios tipos: unos usados para la movilización de carga, otros para movilizar pasajeros y otros usados para la milicia. La mayoría de los aviones están hechos de titanio, aluminio, acero y otros tipos de compuestos ligeros para generar el menor peso posible. Para mantenerse en el aire, los aviones se centran en la fuerza aerodinámica generada en las alas; esta fuerza se origina debido a la

diferencia de presiones tanto en la parte superior como en la inferior de las alas. Cabe recalcar que, mientras más alto vuela el avión, la densidad del aire disminuye y eso genera una menor resistencia aerodinámica [30].

V-B. Fuerza aerodinámica

La aerodinámica es una rama de la mecánica de fluidos que estudia cómo se mueve el aire y qué fuerzas aparecen cuando interactúa con objetos. Analiza, de forma general, fenómenos como la sustentación y el arrastre, que influyen en la estabilidad y la eficiencia de cuerpos en movimiento o expuestos a corrientes de aire. Su concepto se aplica en múltiples ámbitos, desde vehículos y equipos deportivos hasta estructuras y dispositivos, sirve para entender, en términos básicos, cómo el flujo de aire afecta el comportamiento de los objetos. [31].

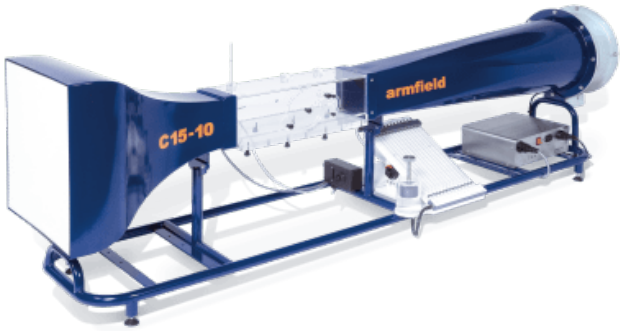


Figura 3. El Túnel de viento o Túnel aerodinámico [31].

V-C. Bird strikes

Las colisiones entre aeronaves y aves, conocidas como ”bird strikes”, representan un desafío significativo para la seguridad aérea. Aunque la mayoría de estos incidentes no causan daños graves, las colisiones con aves de mayor tamaño, como los patos, pueden tener consecuencias catastróficas. Un ejemplo reciente es el accidente del vuelo 2216 de Jeju Air en diciembre de 2024, donde un Boeing 737-800 se estrelló durante un intento de aterrizaje de emergencia en el Aeropuerto Internacional de Muan, Corea del Sur, tras impactar con una bandada de patos. Reportes iniciales señalaron el hallazgo de restos de patos en ambos motores de la aeronave, indicio de que la interacción con aves podría haber desencadenado fallas graves en sus sistemas [32].

A escala mundial, los impactos con fauna (bird strikes) generan pérdidas anuales superiores a 900 millones de dólares para la aviación civil y militar en Estados Unidos y se les atribuyen más de 250 fallecimientos desde 1988. Este escenario refuerza la necesidad de implementar medidas de mitigación eficaces—desde la gestión de hábitats en aeropuertos hasta tecnologías de detección y disuasión—para reducir el riesgo de colisiones y preservar la seguridad de las operaciones aéreas [33].

Figura 4. Accidente por ”birdstrikes”[32].

V-D. Inteligencia Artificial

La inteligencia artificial (IA) es un área de la informática orientada al diseño de sistemas capaces de ejecutar tareas asociadas a la inteligencia humana, tales como aprender, razonar, resolver problemas y comprender el lenguaje natural. Mediante métodos computacionales específicos [34]. A lo largo de su evolución, el campo ha transitado desde esquemas basados en reglas y lógica explícita hacia enfoques de aprendizaje automático y, más recientemente, aprendizaje profundo con redes neuronales, lo que ha ampliado de forma notable su alcance y rendimiento.

V-D1. Visión Artificial: La visión artificial es un ámbito de la inteligencia artificial orientado a que las máquinas extraigan significado de imágenes y videos, aproximando la percepción visual humana. Para lograrlo, integra procedimientos de procesamiento de imágenes, métodos de aprendizaje automático y técnicas de reconocimiento de patrones, con el fin de derivar información útil a partir de datos visuales [35].

Sus aplicaciones abarcan un amplio espectro: desde la inspección de calidad en procesos industriales hasta la conducción autónoma y la seguridad mediante reconocimiento facial. En el plano técnico, los modelos basados en redes neuronales convolucionales (CNN) han marcado el estado del arte en tareas como clasificación y segmentación, aunque su desempeño depende en gran medida de la calidad y representatividad del conjunto de entrenamiento, así como de las condiciones de operación (iluminación, oclusiones, movimiento), factores que influyen de forma directa en la precisión y fiabilidad del sistema [36].

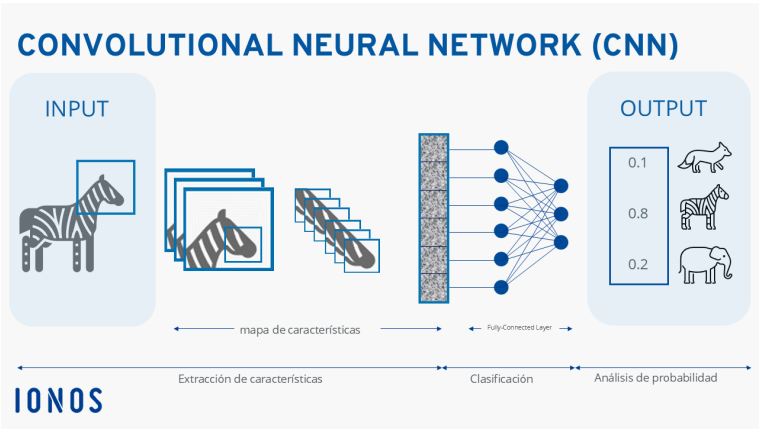


Figura 5. Proceso CNN [36].

V-D2. Procesado de imágenes: El procesamiento digital de imágenes abarca operaciones y análisis sobre datos visuales con el propósito de depurar la señal, realzar rasgos de interés y obtener información útil. Entre las técnicas comunes se incluyen el filtrado, la segmentación, la detección de bordes y la normalización a nivel de píxel. Estas etapas constituyen el pretratamiento indispensable para suministrar entradas consistentes a sistemas de visión por computador y a modelos de aprendizaje profundo [37].

V-D3. Deep learning: Deep Learning es una rama del aprendizaje automático que emplea redes neuronales profundas para aprender representaciones jerárquicas de los datos a partir de múltiples capas de procesamiento. Estas arquitecturas pueden extraer de manera autónoma características complejas, lo que las hace especialmente efectivas en tareas de clasificación, detección y segmentación de imágenes [38].

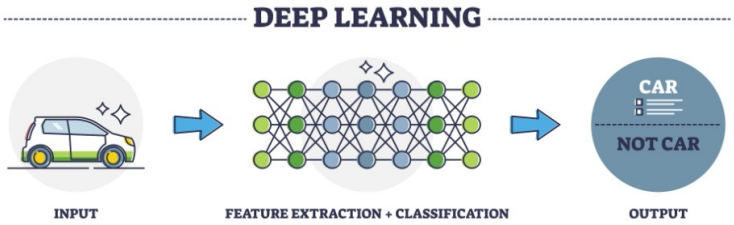


Figura 6. Proceso de deep learning [38].

Una característica distintiva de la IA contemporánea es su habilidad para procesar grandes volúmenes de datos y detectar patrones útiles, lo que habilita aplicaciones en ámbitos como la medicina, el transporte, la industria y las finanzas [39]. No obstante, su despliegue plantea desafíos éticos y sociales, sesgos algorítmicos e impactos sobre el empleo que requieren marcos de gobernanza y evaluación responsable [40].

V-E. YOLO

Familia de métodos de detección de objetos de una sola etapa orientada al tiempo real. El enfoque procesa la imagen completa con una única red neuronal y, en un solo pase, estima cajas delimitadoras, puntajes de confianza y probabilidades de clase por celda de una cuadrícula. Al unificar localización y clasificación en la misma arquitectura, consigue alta velocidad con una precisión competitiva, lo que lo hace adecuado para escenarios de inferencia en campo y dispositivos embebidos. [41].

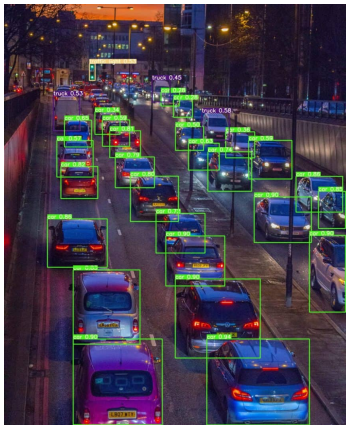


Figura 7. Interfaz YOLO [41].

V-F. Python

Python es un lenguaje de programación de alto nivel e interpretado que prioriza la legibilidad y la productividad. Admite varios paradigmas (orientado a objetos, funcional y procedural), cuenta con tipado dinámico y una biblioteca estándar amplia para manejo de archivos, redes, concurrencia y más. Su ecosistema de paquetes, accesible con pip y entornos virtuales, incluye herramientas como NumPy y Pandas (cómputo y datos), Matplotlib (gráficos), OpenCV (visión por computador) y PyTorch/TensorFlow (aprendizaje automático). Esta combinación lo hace adecuado tanto para prototipado rápido como para despliegues en producción, abarcando automatización, servicios web, ciencia de datos, visión artificial e inteligencia artificial [42].

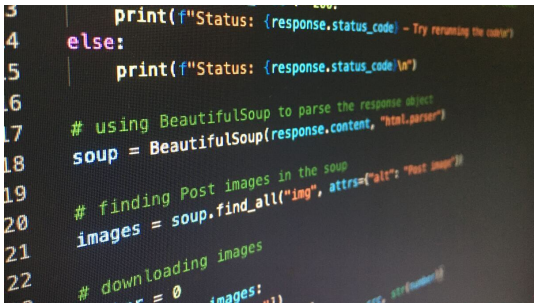


Figura 8. Lenguaje python [42].

V-G. Colab

Google Colab es un entorno de cuadernos en la nube, basado en Jupyter, que permite ejecutar código Python directamente desde el navegador sin configuración local. Ofrece acceso a CPU, GPU y TPU bajo demanda, integra bibliotecas científicas preinstaladas (por ejemplo, NumPy, Pandas, Matplotlib, OpenCV y frameworks de aprendizaje automático) y se conecta con Google Drive para gestionar datasets y resultados. Facilita el prototipado, el análisis de datos, el entrenamiento y la demostración reproducible de modelos, con la posibilidad de compartir cuadernos para revisión y colaboración. En términos de flujo de trabajo, permite importar datos desde la web o repositorios, montar Drive, registrar métricas y generar visualizaciones en un mismo documento.

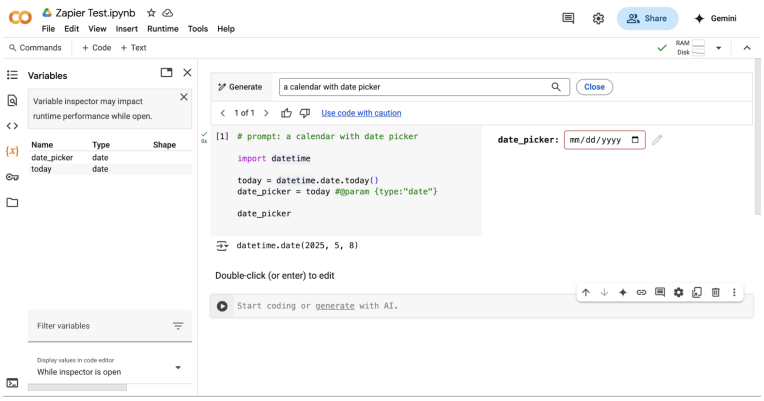


Figura 9. Entorno de trabajo [43].

V-H. Raspberry Pi

La Raspberry Pi es una computadora de placa única, de bajo costo y consumo reducido, impulsada por la Raspberry Pi Foundation. Integra un SoC con arquitectura ARM y ofrece conectividad mediante GPIO, además de interfaces para cámara y pantalla (CSI/DSI) y puertos USB. Gracias a esta combinación de hardware compacto y periféricos, se convierte en una plataforma versátil para implementar proyectos de visión por computador y despliegue de modelos de aprendizaje profundo en el borde [44].

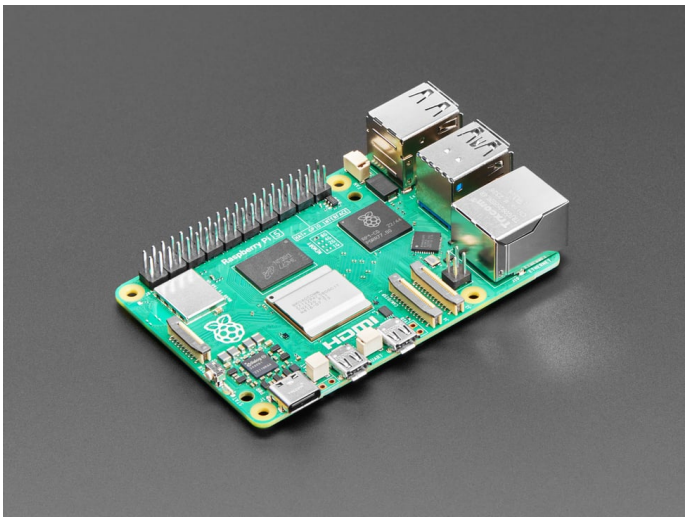


Figura 10. Raspberry Pi 5 [44].

V-H1. Raspberry Pi OS: Raspberry Pi OS es una distribución basada en Debian pensada para aprovechar el hardware de las placas Raspberry Pi[45]. Ofrece imágenes con y sin escritorio, gestor de paquetes APT y utilidades de configuración para habilitar cámara, GPIO, redes y aceleración multimedia. Para proyectos de visión, facilita el uso de libcamera/V4L2, Python y bibliotecas como OpenCV, integrándose con pantallas DSI y módulos de cámara CSI. Puede operar en modo de escritorio o headless (SSH/VNC) y trabajar sobre microSD o unidades más rápidas.

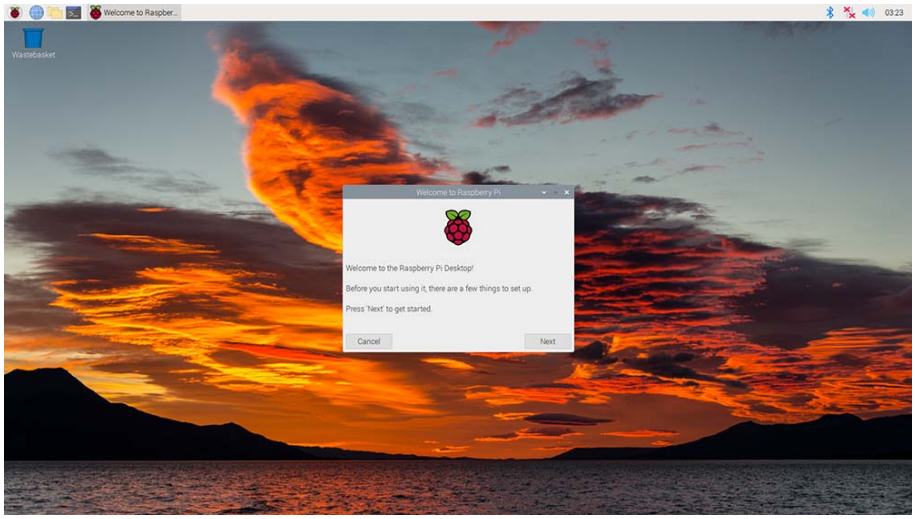


Figura 11. Entorno del sistema operativo [45].

V-H2. Raspberry Pi Touch Display: La pantalla táctil oficial de Raspberry Pi de 7” se conecta por DSI y ofrece una resolución de 800×480 px en un panel TFT con multitáctil de hasta 10 puntos. Se alimenta directamente desde el GPIO y se integra con un cable ribbon, permitiendo interfaces gráficas compactas y de bajo peso para entornos de campo [46].

Figura 12. Raspberry Pi Touch Display [46].

V-I. Webcam Full HD

Una webcam Full HD es una cámara digital USB que entrega video a 1920×1080 píxeles (1080p), comúnmente a 30 fps. La mayoría cumple con la especificación UVC (USB Video Class), por lo que funciona de manera “plug-and-play” en Windows, macOS y sistemas basados en Linux (incluida la Raspberry Pi) [47].

Figura 13. Webcam Full HD [47].

VI. Marco Metodológico

VI-A. Elaboración de Dataset

La conformación del dataset constituye la base del sistema, pues a partir de este conjunto el modelo aprende los patrones visuales de interés. Para el desarrollo del presente proyecto se reúnen imágenes desde distintas fuentes en línea, procurando diversidad de fondos, ángulos y condiciones de iluminación. Cada archivo se anota de forma consistente con su categoría de especie, de modo que el conjunto de entrenamiento resulte representativo y facilite la generalización del modelo en escenarios reales (Figura 14).

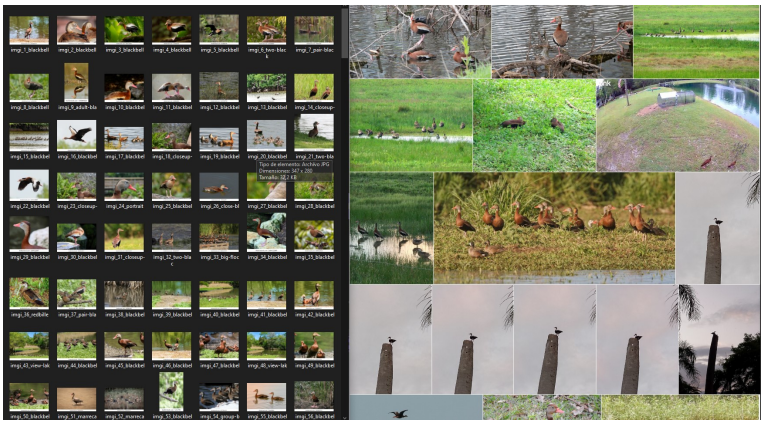


Figura 14. Compilado de imágenes para el dataset Fuente: por A. Macero y J. Ronquillo.

VI-B. Organización por especie y particiones

Para organizar las imágenes por especie y por conjunto, se estructurará el directorio del dataset (Figura 15). De este modo, cada fotografía se colocará en la carpeta correspondiente a su especie, dentro del subdirectorio de entrenamiento o de prueba. Esta convención facilita tanto el etiquetado automático como la carga del dataset en los scripts de entrenamiento y evaluación, garantizando que el modelo reciba claramente diferenciadas las instancias de cada especie en cada fase.

Figura 15. Organización de imágenes para el dataset Fuente: por A. Macero y J. Ronquillo.

VI-C. Pruebas en modelo pre-entrenado

Con el fin de comprender el flujo extremo a extremo y validar tempranamente el comportamiento del modelo, se realizaron pruebas exploratorias de interfaz y precisión sobre un prototipo inicial. Estas pruebas no buscan establecer desempeño definitivo, sino detectar patrones de error y orientar ajustes en el dataset, el umbral de decisión y la arquitectura.

En la primera prueba (Figura 16), se evaluaron imágenes con múltiples individuos en la misma escena. Se observó que la predicción pierde consistencia cuando existen solapamientos, variaciones de escala y fondos heterogéneos. Este resultado es coherente con prototipos basados en clasificación por imagen completa, que no localizan instancias de manera explícita y, por tanto, pueden mezclar señales de varias aves.

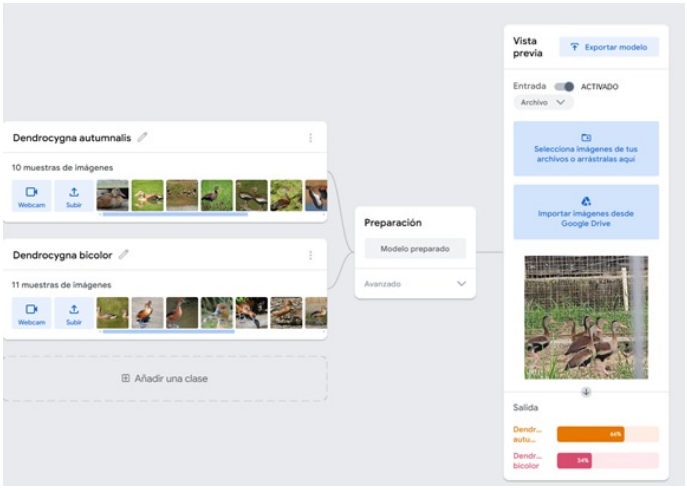


Figura 16. Primera prueba de modelo de aprendizaje Fuente: por A. Macero y J. Ronquillo.

En la segunda prueba (Figura 17), se empleó una toma nítida y con un solo individuo de Dendrocygna autumnalis. En estas condiciones, la salida alcanzó alta confianza, evidenciando que la claridad del contorno, el contraste con el fondo y la ausencia de oclusiones mejoran la precisión.

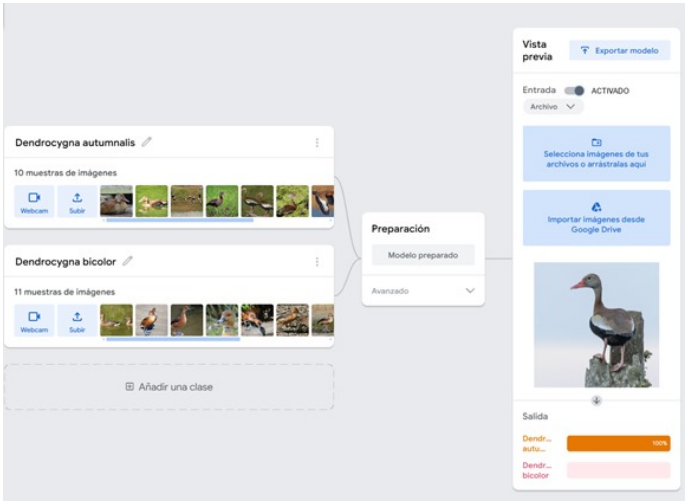


Figura 17. Prueba con porcentaje mas preciso del D. autumnalis Fuente: por A. Macero y J. Ronquillo.

La tercera prueba (Figura 18) se centró en Dendrocygna bicolor bajo un escenario similar, obteniendo igualmente porcentajes elevados de acierto. Estos casos confirman que, con imágenes bien encuadradas y un único sujeto, el prototipo responde de forma estable.

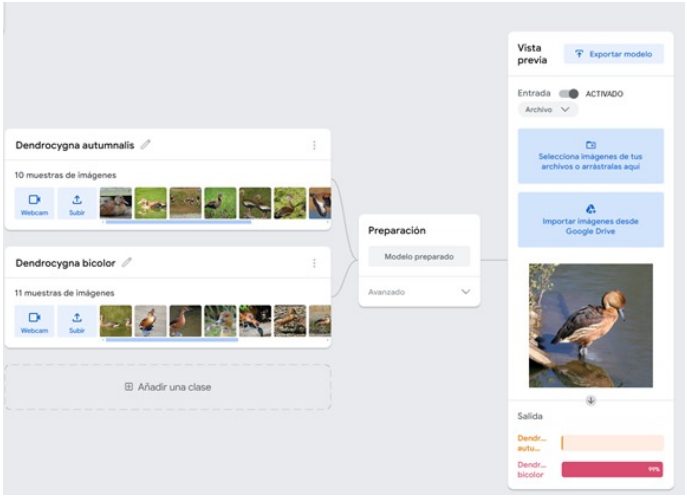


Figura 18. Prueba con mayor porcentaje de acierto en la especia D. bicolor Fuente: por A. Macero y J. Ronquillo.

VI-D. Etiquetado de imágenes

Se anotan manualmente las instancias de cada ave mediante cajas delimitadoras y su clase correspondiente (Figura 19). Se adopta un protocolo de consistencia como el tamaño mínimo de la caja, evitar solapamientos innecesarios, centrado sobre el sujeto y se valida una muestra de anotaciones para control de calidad (Figura 20).

Figura 19. Etiquetado enfocado en la especie dendrocygna autumnalis Fuente: por A. Macero y J. Ronquillo.

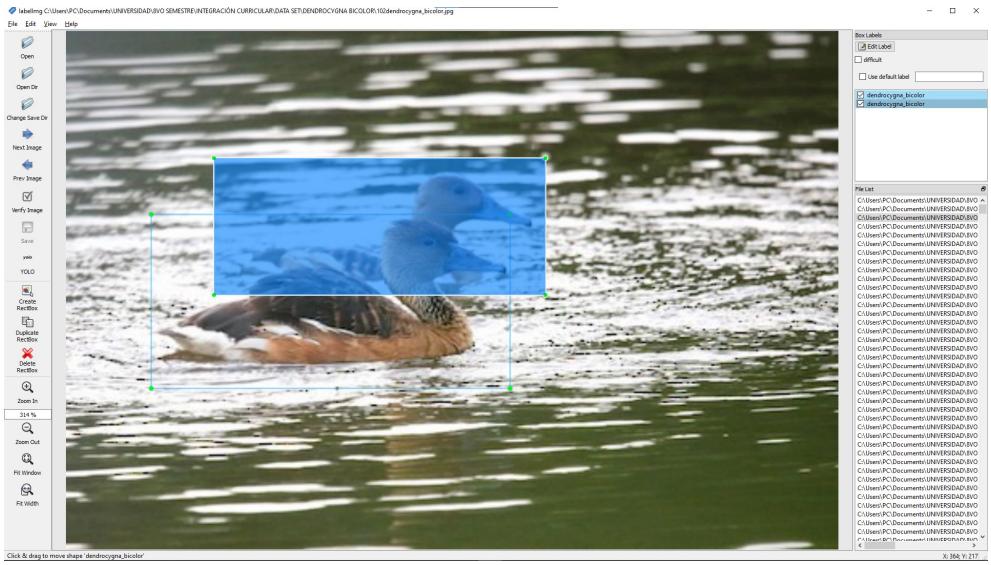


Figura 20. Etiquetado enfocado en la especie dendrocygna bicolor Fuente: por A. Macero y J. Ronquillo.

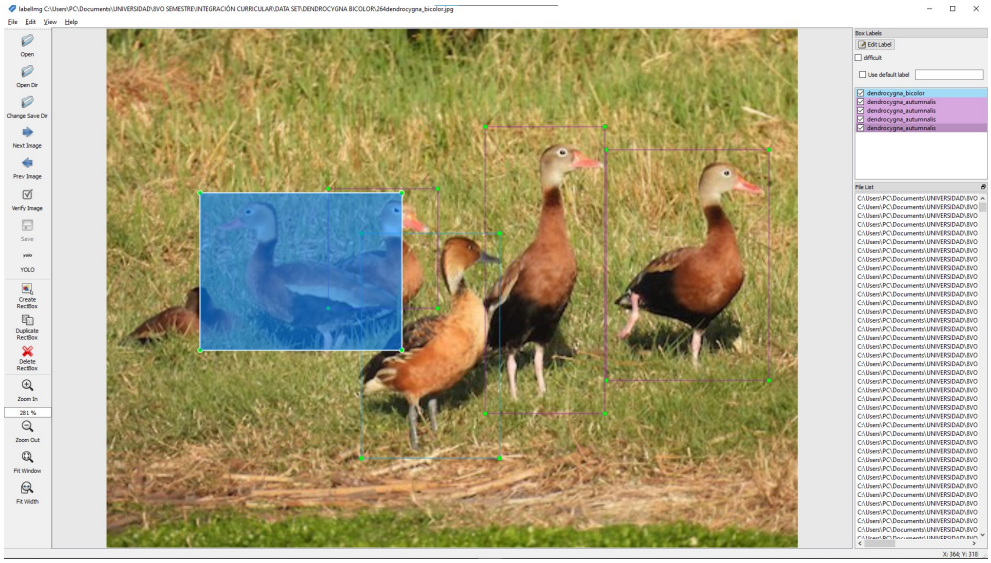


Figura 21. Etiquetado para las dos especies Fuente: por A. Macero y J. Ronquillo.

VI-E. Exportación a formato de entrenamiento

Se genera la carpeta de exportación en formato YOLO: imágenes + etiquetas .txt por imagen, más archivo de configuración con clases (Figura 23). Se comprime el dataset (.zip) para su uso directo en Colab (Figura 24).

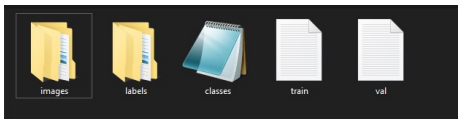


Figura 22. División de archivos para exportar Fuente: por A. Macero y J. Ronquillo.

Tabla I
NÚMERO DE IMÁGENES PARA EL DATASET.

Especie	**Numero**
Dendrocygna autumnalis	1740
Dendrocygna bicolor	1620
Global (todas)	3360

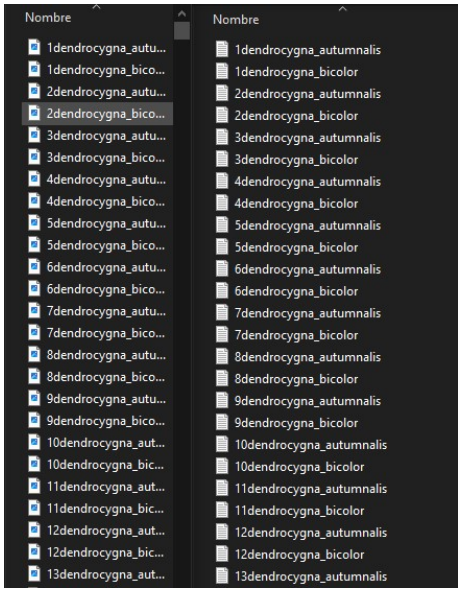


Figura 23. Imágenes y su formato .txt Fuente: por A. Macero y J. Ronquillo.

```
train: /content/dataset/train/images
val:   /content/dataset/val/images
test:  /content/dataset/test/images
names: [Dendrocygna_bicolor, Dendrocygna_autumnalis]
```

Figura 24. Ruta del dataset exportado a colab Fuente: por A. Macero y J. Ronquillo.

VI-F. Preparación del entorno en Google Colab

Se habilita un entorno de ejecución con aceleración por GPU y se actualizan utilidades básicas de Python. A continuación, se clona el repositorio oficial de YOLOv7 y se instalan sus dependencias desde requirements.txt. Para gestionar el dataset, se habilita la descarga y descompresión del paquete de imágenes y etiquetas en un directorio de trabajo. Con el fin de evitar interferencias en el registro de métricas, se desactiva Weights & Biases (W&B) durante el entrenamiento.

Para habilitar el entrenamiento en la nube, el flujo de trabajo inicia con la verificación del entorno y la actualización de utilidades base. Se comprueba la disponibilidad de GPU y la versión de Python, y posteriormente se actualizan los gestores de paquetes (pip, wheel y setuptools). Esta puesta a punto reduce conflictos de dependencias y asegura instalaciones limpias (Figura 25).

```
[ ]  # Limpio estado y actualizo herramientas básicas
     !nvidia-smi -L || true
     !python -V
     !pip -q install --upgrade pip wheel setuptools
```

Figura 25. Verificación del entorno Fuente: por A. Macero y J. Ronquillo.

A continuación, se prepara el conjunto de datos. Se crea el directorio de trabajo en /content, se instala la herramienta gdown y se descarga el paquete del dataset desde Google Drive, que luego se descomprime en una carpeta dedicada para mantener separadas imágenes y etiquetas de acuerdo con el formato YOLO (Figura 26). Esta etapa deja el material listo para su referencia desde el archivo de configuración data.yaml.

```
[ ]  %cd /content
     !rm -rf yolov7
     !git clone https://github.com/WongKinYiu/yolov7.git
     %cd /content/yolov7

     # Instala deps del repo (sin forzar torch/cuda: Colab ya trae)
     !pip -q install -r requirements.txt
```

Figura 26. Descarga dataset desde Google Drive Fuente: por A. Macero y J. Ronquillo.

Finalmente, se procede con el clonado del repositorio de YOLOv7 y la instalación de sus dependencias. Dado que Colab incluye binarios de torch y CUDA, se evita forzar versiones externas para mantener la compatibilidad del entorno. Con ello, se deja operativo el repositorio para ejecutar los scripts de entrenamiento, validación y exportación (Figura 27).

```
%cd /content
%pip install -q gdown
!gdown --fuzzy "https://drive.google.com/file/d/18PJ2KzfQMP4h4V7-ckJA4TKyytXs6cMJ/view?usp=sharing" -O data_yolov7.zip
!unzip -q data_yolov7.zip -d data_yolov7
!rm data_yolov7.zip
```

Figura 27. Clonación repositorio Yolov7 Fuente: por A. Macero y J. Ronquillo.

VI-G. Configuración del archivo data.yaml

El archivo data.yaml centraliza la ubicación del dataset y el mapa de clases que utilizará el detector (Figura 28). En este proyecto se define la ruta absoluta a los subconjuntos train y val y se declara nc: 2 con los nombres de especie en el mismo orden que se usó al anotar en LabelImg (Figura 29). Este orden determina los ID de clase durante el entrenamiento (0 = dendrocygna_autumnalis, 1 = dendrocygna_bicolor). Las rutas de images/ deben tener su carpeta espejo labels/ con archivos .txt en formato YOLO y nombre idéntico.

```
%cd /content/yolov7
!ls -1 cfg/training | grep -i tiny || true
!ls -l data | sed -n '1,200p'
!sed -n '1,200p' data/custom_tiny.yaml
```

Figura 28. Definición y ubicación de rutas Fuente: por A. Macero y J. Ronquillo.

```
train: /content/data_yolov7/data_yolov7/images/train
val: /content/data_yolov7/data_yolov7/images/val
nc: 2
names:
- dendrocygna_autumnalis
- dendrocygna_bicolor
```

Figura 29. Contenido del .yaml Fuente: por A. Macero y J. Ronquillo.

VI-H. Entrenamiento del modelo

Se opta por YOLOv7-tiny porque ofrece un equilibrio adecuado entre rapidez de inferencia y precisión para un escenario con recursos limitados y necesidad de respuesta casi en tiempo real (Figura 30). El entrenamiento se ejecuta en GPU utilizando el archivo data.yaml del proyecto y el conjunto de hiperparámetros base (hyp.scratch.tiny.yaml).

```
%cd /content/yolov7
!python train.py --weights yolov7-tiny.pt
--cfg cfg/training/yolov7-tiny.yaml
--data /content/yolov7/data/custom_tiny.yaml
--hyp data/hyp.scratch.tiny.yaml
--epochs 300
--batch-size 16
--img-size 640 640
--device 0
--workers 8
--name exp_tiny
```

Figura 30. Ajustes para el entrenamiento de YOLOv7-tiny Fuente: por A. Macero y J. Ronquillo.

VI-I. Diagrama de funcionamiento

El diagrama mostrado en la figura 31 explica el flujo de trabajo de nuestro sistema de visión, se captura la escena y se envía el fotograma a la Raspberry Pi, donde se ejecuta el modelo de detección. El módulo de inferencia analiza la imagen, genera las coordenadas de la caja delimitadora y la etiqueta de la especie detectada, y a continuación transmite ese resultado a la interfaz de salida para su visualización en tiempo real.

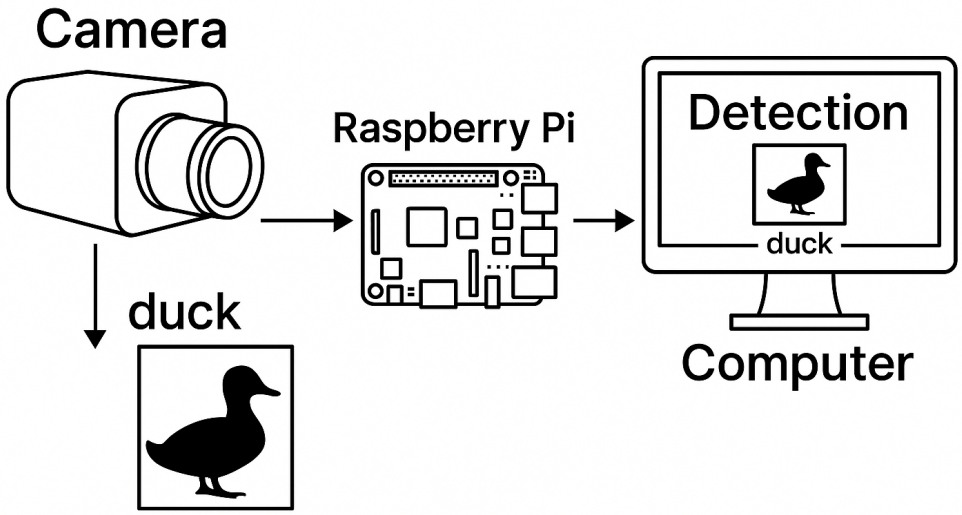


Figura 31. Esquema gráfico del proyecto Fuente: por A. Macero y J. Ronquillo.

VI-J. Métricas de evaluación

VI-J1. Accuracy-Precisión General: Accuracy como primera métrica de evaluación la aplicaremos al medir el porcentaje de predicciones correctas que hace el modelo respecto al total de predicciones realizadas (Figura 1). Si el modelo clasificó correctamente 90 imágenes de 100, la precisión será del 90 por ciento.

$$Accuracy = \frac{PrediccionesCorrectas}{TotalDePredicciones} \tag{1}$$

VI-J2. Precisión por clase: En esta métrica de evaluación buscamos evaluar la precisión de nuestro algoritmo enfocada en las dos especies de patos, lo que se busca encontrar son los falsos positivos. Mientras menos falsos positivos tenga nuestro algoritmo, mayor precisión tendrá (Figura 2).

$$Precisión = \frac{VerdaderosPositivos(TP)}{VerdaderosPositivos(TP) + FalsosPositivos(FP)} \tag{2}$$

Tabla II
RESUMEN DE MÉTRICAS POR CLASE EN EL CONJUNTO DE PRUEBA.

Clase	**mAP@0.5**	**mAP@0.5:0.95**
Dendrocygna autumnalis	0.847	—
Dendrocygna bicolor	0.870	—
Global (todas)	0.858	—

VI-J3. Recall o Exhaustividad: En esta métrica de evaluación buscamos medir el porcentaje de positivos reales de una de las clases de patos, tomando como base en la que centrarse en los falsos negativos. Un alto recall significa que el modelo identifica casi todos los positivos verdaderos (Figura 3).

$$Precisión = \frac{VerdaderosPositivos(TP)}{VerdaderosPositivos(TP) + FalsosNegativos(FP)} \tag{3}$$

Tabla III
PUNTO OPERATIVO RECOMENDADO A PARTIR DE LA CURVA F1.

Métrica	**Valor**	**Comentario**
Umbral de confianza óptimo	0.33	Máximo de F1 observado
F1 (global) @ 0.33	0.81	Compromiso precisión–recall

VI-K. Resultados del entrenamiento

VI-K1. Resultados cualitativos del entrenamiento: Se presentan mosaicos con predicciones del detector sobre el conjunto de validación/prueba, donde cada individuo detectado aparece marcado con cajas delimitadoras coloreadas por clase(Figura 32). Las imágenes incluyen escenas variadas (agua, vegetación densa, grupos en movimiento, individuos aislados y bandadas a distancia), lo que permite observar fortalezas y límites del modelo más allá de los indicadores numéricos (Figura 34).

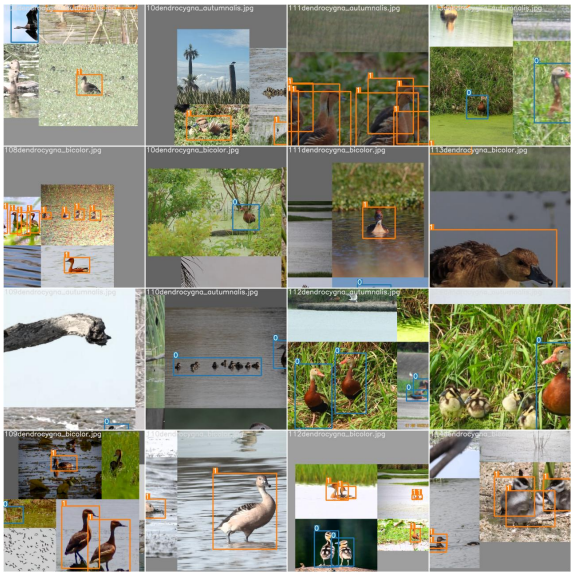


Figura 32. Predicciones en validación Fuente: por A. Macero y J. Ronquillo.

Figura 33. Predicciones con diferente iluminación Fuente: por A. Macero y J. Ronquillo.

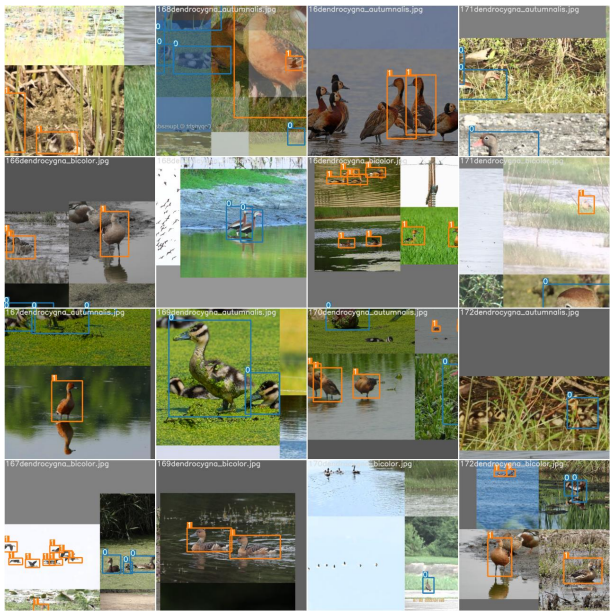


Figura 34. Predicciones en escenarios desafiantes Fuente: por A. Macero y J. Ronquillo.

VI-K2. Resultados cuantitativos del entrenamiento: En términos cuantitativos, el modelo parte en la primera época con P=46.74 %, R=25.23 % y mAP@0.5=9.63 %, y culmina con valores consolidados en las últimas iteraciones. El mejor desempeño promedio se registra alrededor de la época 115, con P=78.76 %, R=84.88 % y mAP@0.5=48.9 %, lo que evidencia una buena capacidad para acertar cuando se reporta una detección y una cobertura alta de verdaderos positivos en el conjunto de validación. En la última época (cierre de entrenamiento) los indicadores se mantienen estables: P=81.87 %, R=85.57 % y mAP@0.5=48.56 %, confirmando que el modelo converge sin degradación(Figura 35).

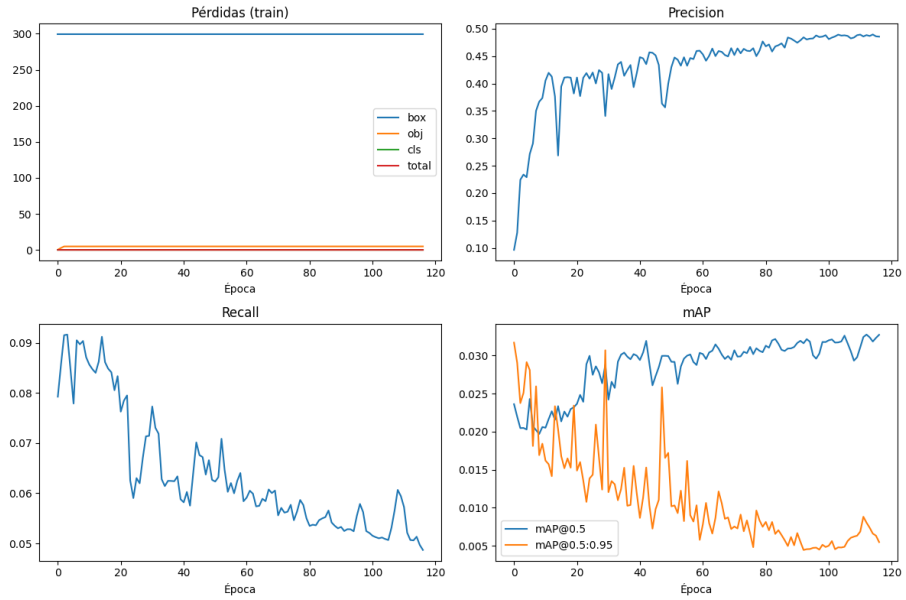


Figura 35. Gráficas de los resultados en el entrenamiento Fuente: por A. Macero y J. Ronquillo.

VI-L. Preparación del entorno en Raspberry Pi

Se crea un entorno virtual, se instalan dependencias mínimas para inferencia (onnxruntime, opencv-python-headless, numpy, PyYAML, pillow, protobuf, etc.), y se copian los artefactos best.onnx, custom.yaml y scripts de inferencia (Figura 36).

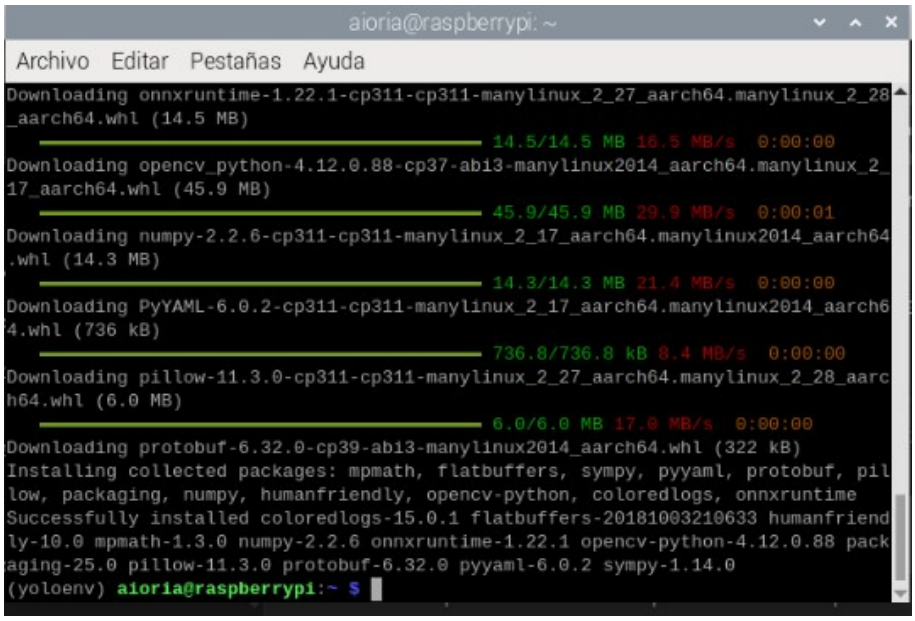


Figura 36. Instalación de dependencias para inferencia en la Raspberry Pi Fuente: por A. Macero y J. Ronquillo.

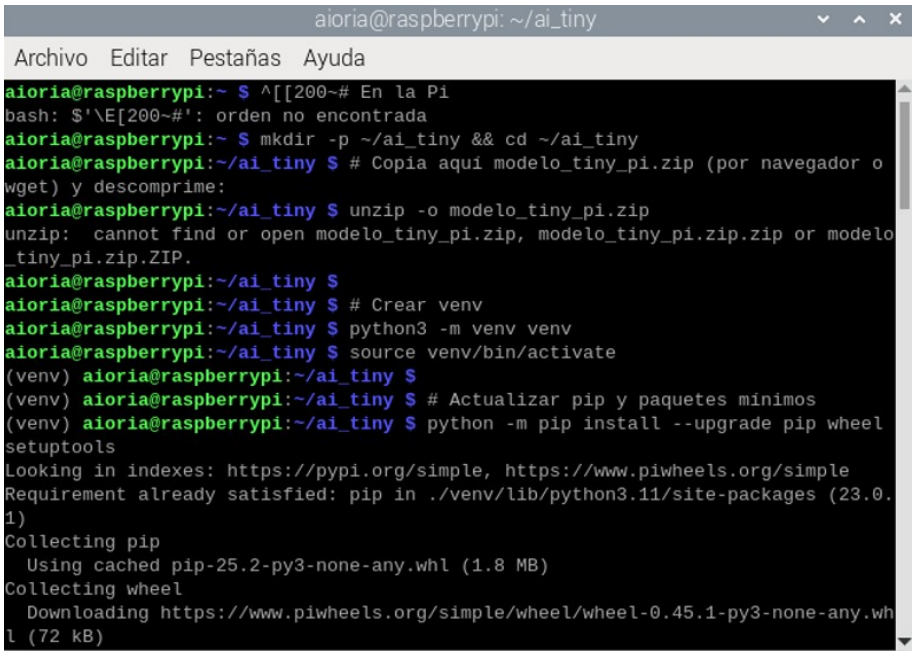


Figura 37. Descarga de artefactos desde Drive y creación del entorno virtual para aislar dependencias Fuente: por A. Macero y J. Ronquillo.

VI-M. Carga del modelo ONNX y verificación de periféricos

Se organiza la carpeta de trabajo (Figura 38), se verifican dispositivos de vídeo (v4l2), y se valida la disponibilidad de los ficheros (best.onnx, custom.yaml). Se documentan incidencias típicas (falta de custom.yaml, aviso del plugin Qt al usar –view) y sus mitigaciones (ejecución headless con –save).

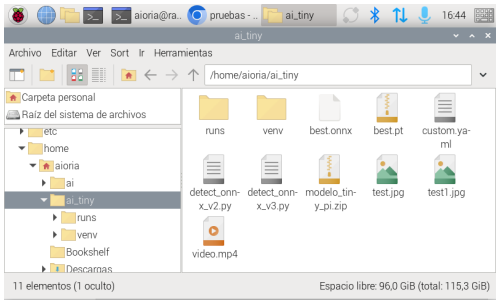


Figura 38. Estructura del proyecto en la Raspberry Pi con el modelo ONNX, configuración y scripts de inferencia Fuente: por A. Macero y J. Ronquillo.

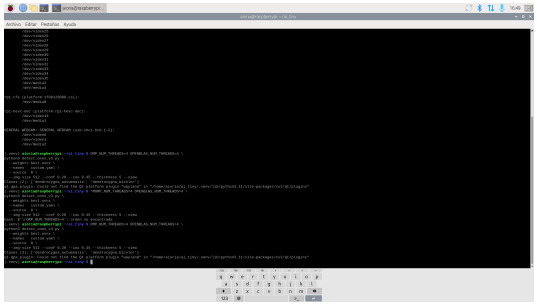


Figura 39. Verificación de dispositivos de vídeo y prueba de carga del modelo Fuente: por A. Macero y J. Ronquillo.

Se documenta el chequeo previo al despliegue en la Raspberry Pi. Primero, se listan los dispositivos de vídeo disponibles (/dev/video*) y se ejecuta una carga de prueba del modelo en ONNX para confirmar que la webcam UVC es reconocida y que las dependencias (onnxruntime, OpenCV) inicializan sin errores (Figura 39). Luego, el diagnóstico inicial revela un problema de rutas: el script no encuentra archivos de configuración (p. ej., custom.yaml) ni el paquete del modelo en el directorio de trabajo. Tras descomprimir el paquete del proyecto y colocar best.onnx y custom.yaml en la carpeta correcta, además de verificar las rutas pasadas por línea de comandos, la prueba se repite con éxito (Figura 40).

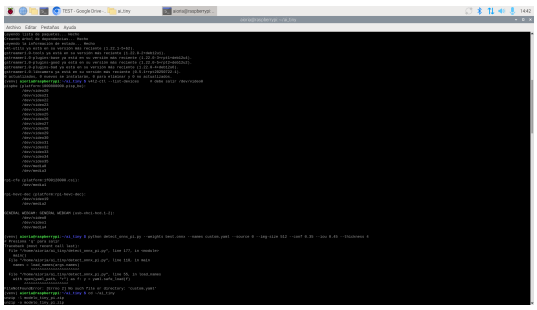


Figura 40. Diagnóstico inicial: resolución de rutas y archivos requeridos Fuente: por A. Macero y J. Ronquillo.

VI-N. Diagrama de flujo del sistema

La figura resume la lógica de operación del prototipo desde la activación hasta la finalización. Al iniciar, se habilita la cámara (webcam Full HD) y comienza la captura continua de video(Figura 41). Cada fotograma pasa por el procesamiento de imagen e inferencia del modelo YOLOv7-tiny (ONNX).

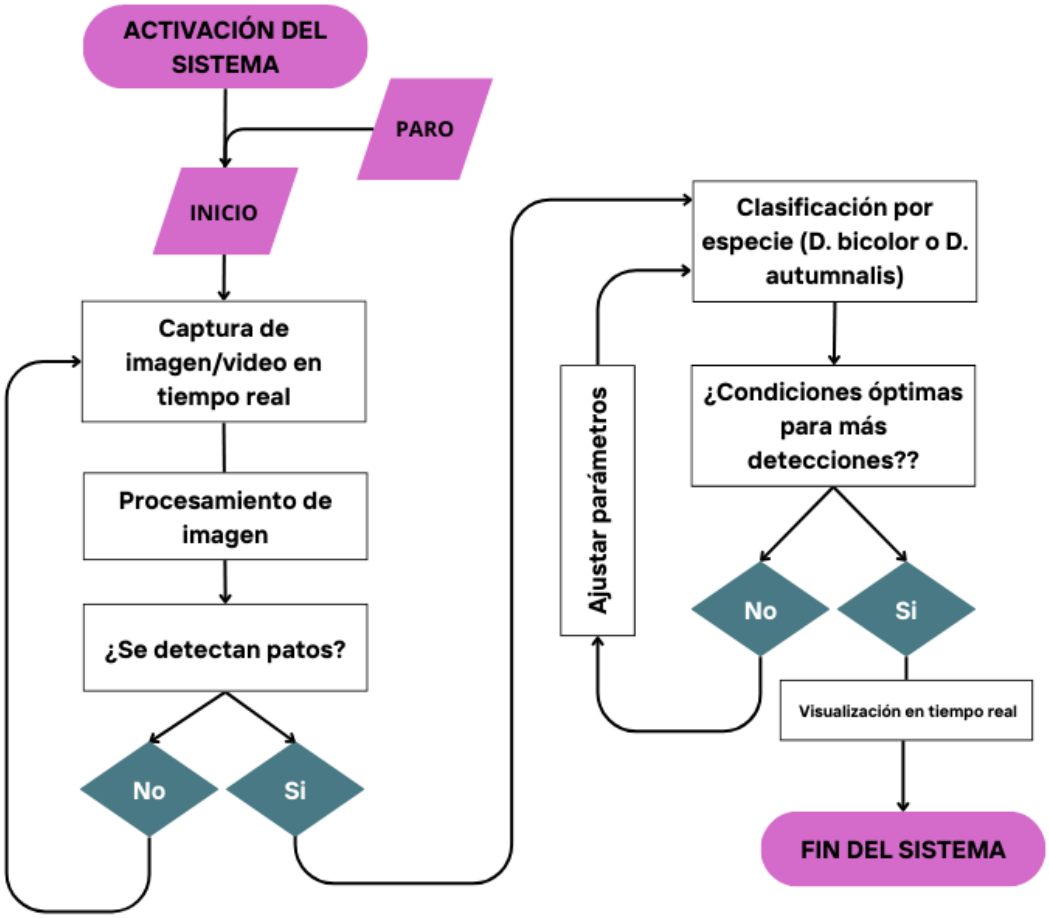


Figura 41. Diagrama de flujo Fuente: por A. Macero y J. Ronquillo.

VII. Resultados

VII-A. Desempeño cuantitativo global

Con base en la evaluación sobre el conjunto de prueba, el modelo presenta un desempeño competitivo para ambas clases. El indicador principal alcanza un mAP@0.5 = 0,858 (global), con valores cercanos a 0,847 para D. autumnalis y 0,870 para D. bicolor (Figura 42). Estas cifras reflejan una capacidad de detección estable en escenas variadas y constituyen la referencia para comparar futuras mejoras del sistema43).

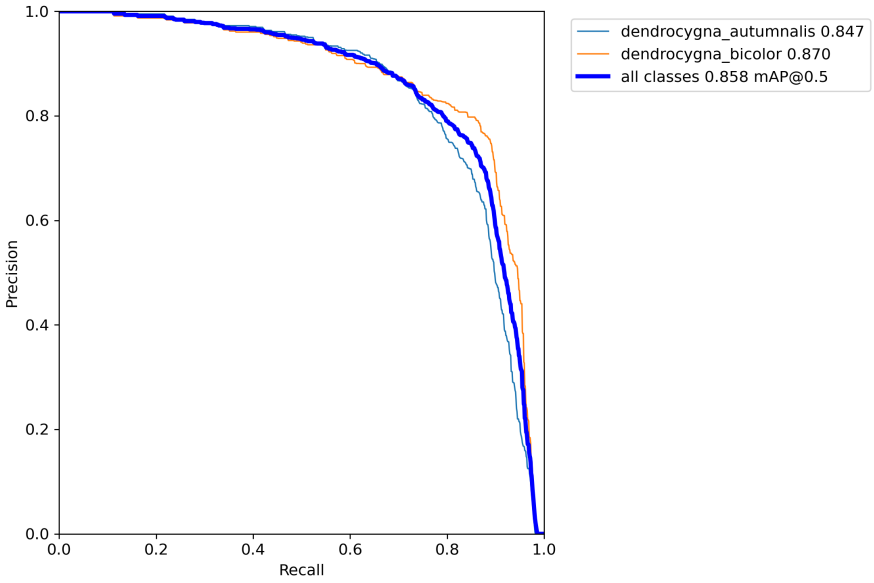


Figura 42. Curva Precisión–Recall sobre el conjunto de prueba Fuente: por A. Macero y J. Ronquillo.

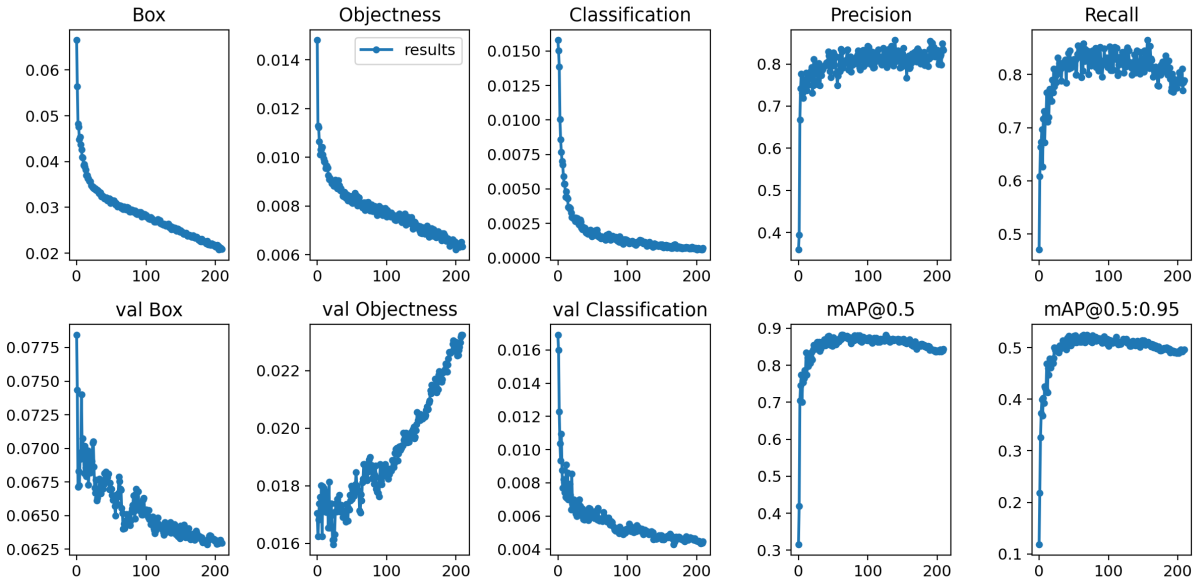


Figura 43. Evolución de métricas durante el entrenamiento: estabilización de mAP y pérdidas hacia las últimas épocas Fuente: por A. Macero y J. Ronquillo.

VII-B. Matriz de confusión y análisis de errores

La matriz de confusión muestra altas tasas de acierto en la diagonal para ambas clases, con una ligera ventaja de D. bicolor respecto de D. autumnalis. La confusión entre especies es marginal, lo que sugiere que el modelo distingue adecuadamente los rasgos característicos de cada taxón (Figura 44).

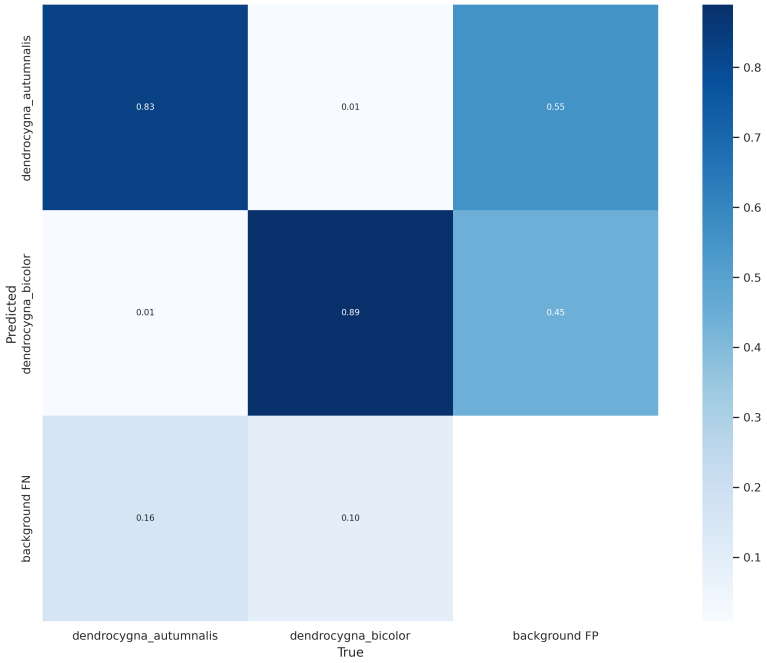


Figura 44. Matriz de confusión normalizada sobre el conjunto de prueba Fuente: por A. Macero y J. Ronquillo.

VII-C. Umbral de confianza y compromiso Precisión–Recall

Para determinar el punto de operación se analizaron las curvas F1 vs. confianza, precisión vs. confianza y recall vs. confianza (Figura 45). Tal como es esperable, la precisión aumenta al elevar el umbral, mientras que el recall disminuye; el equilibrio entre ambos se refleja en la F1.

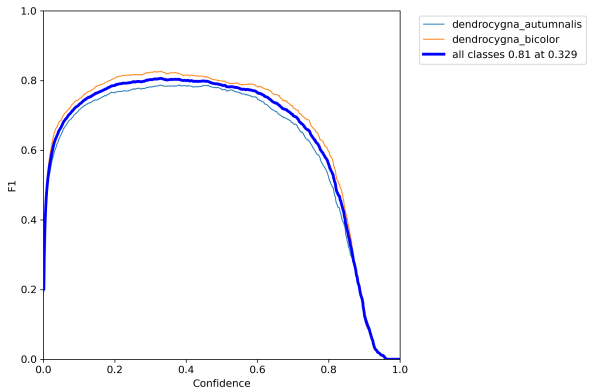


Figura 45. F1 en función del umbral de confianza; máximo $\approx$ 0,81 en 0,33 Fuente: por A. Macero y J. Ronquillo.

VII-C1. Umbral de confianza y compromiso Precisión–Recall: Como complemento a la curva F1, se analizó el comportamiento individual de Precisión y Recall frente al umbral de confianza (Figura 46). La Precisión muestra una tendencia monótona creciente: al elevar el umbral se reduce la incidencia de falsos positivos hasta valores próximos a 1.0 en los umbrales más altos.

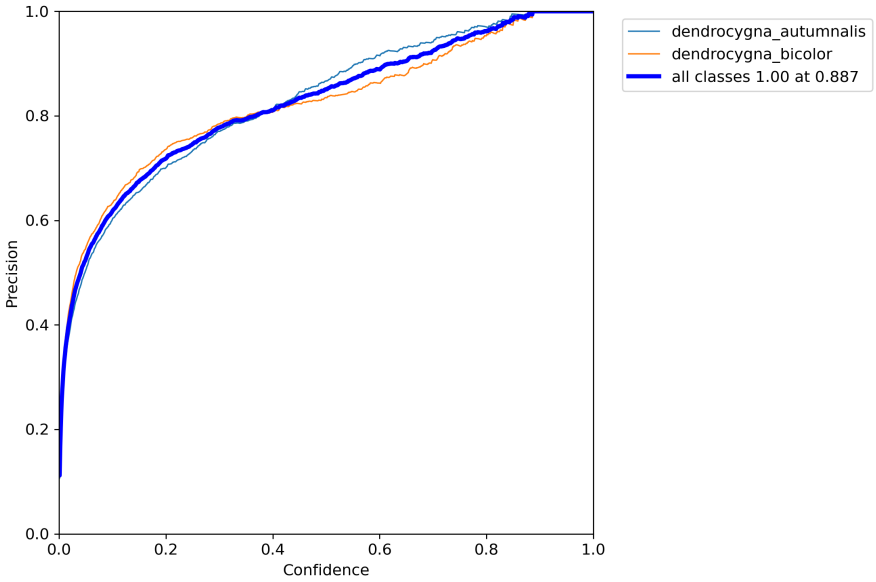


Figura 46. La precisión aumenta con el umbral y se aproxima a 1,0 en valores altos Fuente: por A. Macero y J. Ronquillo.

En sentido opuesto, el Recall decrece gradualmente al aumentar el umbral, reflejando la pérdida de verdaderos positivos cuando se exige mayor confianza para aceptar una detección. La separación entre clases es moderada (Figura 47).

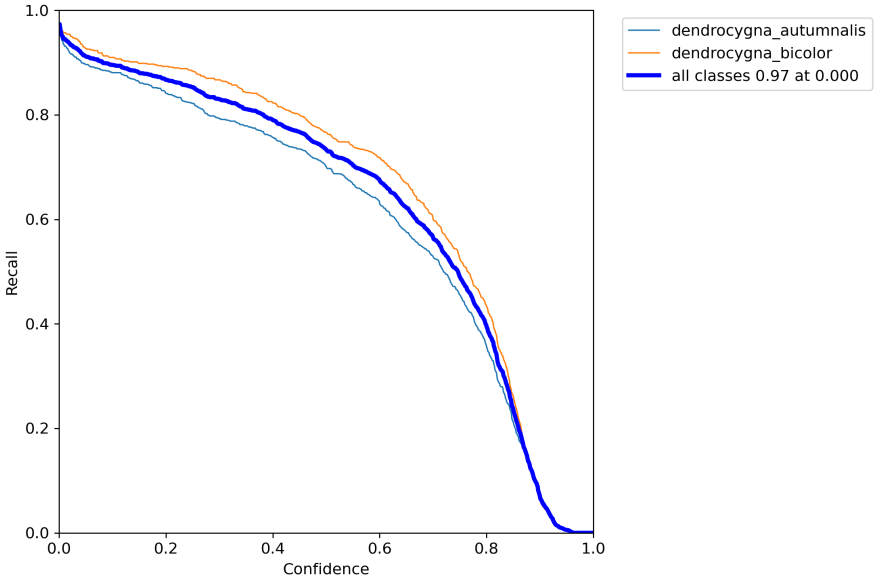


Figura 47. El recall disminuye al elevar el umbral Fuente: por A. Macero y J. Ronquillo.

VII-D. Evolución de métricas de validación por época

La evolución de métricas en validación por época muestra que el entrenamiento converge y alcanza una meseta estable hacia las fases finales. La precisión y el recall tienden a mejorar en las primeras épocas y a estabilizarse después, en línea con lo observado en las curvas de desempeño global (Figura 48).

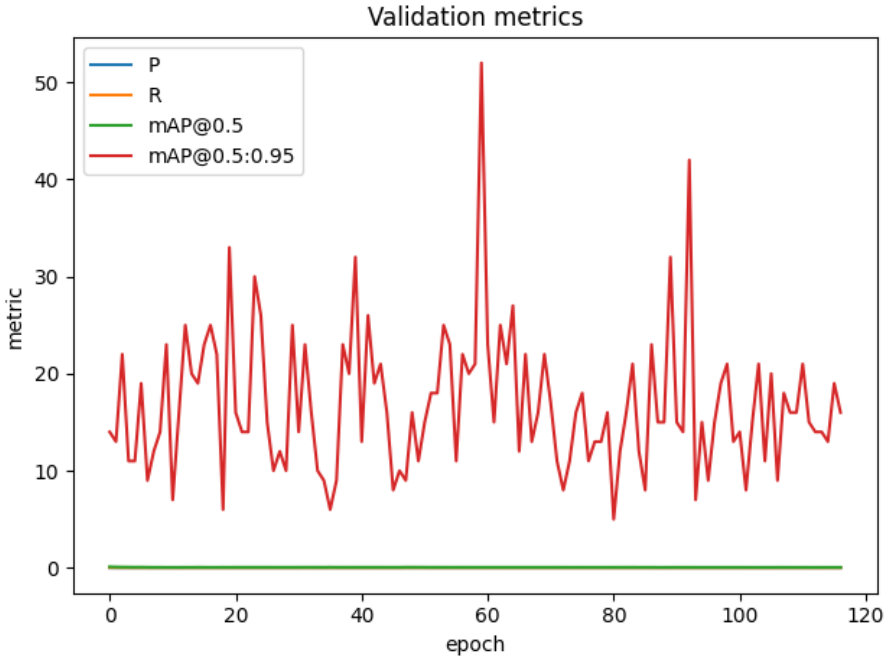


Figura 48. Validación por época Fuente: por A. Macero y J. Ronquillo.

VII-E. Pruebas de inferencia en Raspberry Pi

Con el modelo cargado, se ejecutan inferencias sobre imagen fija y flujo de vídeo (Figura 49). Se reporta la salida guardada y el rendimiento observado (FPS aproximados).

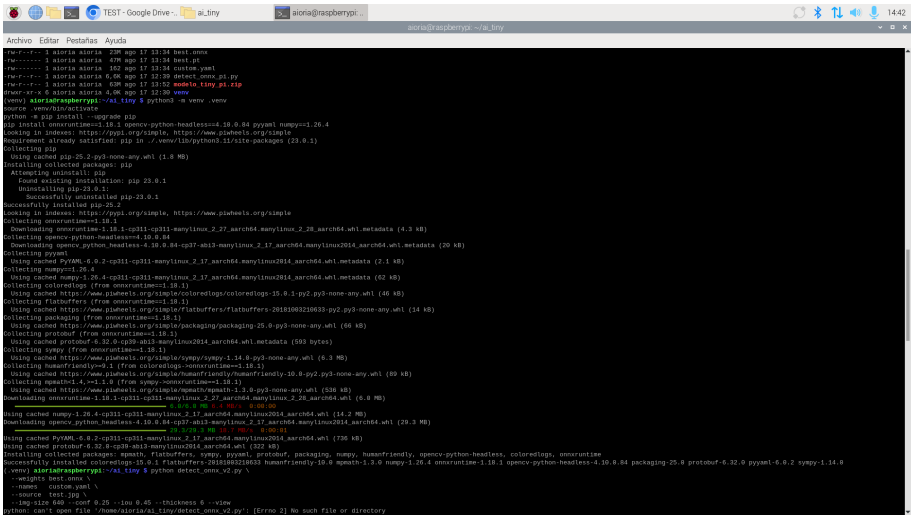


Figura 49. Inferencia sobre imagen fija en Raspberry Pi Fuente: por A. Macero y J. Ronquillo.

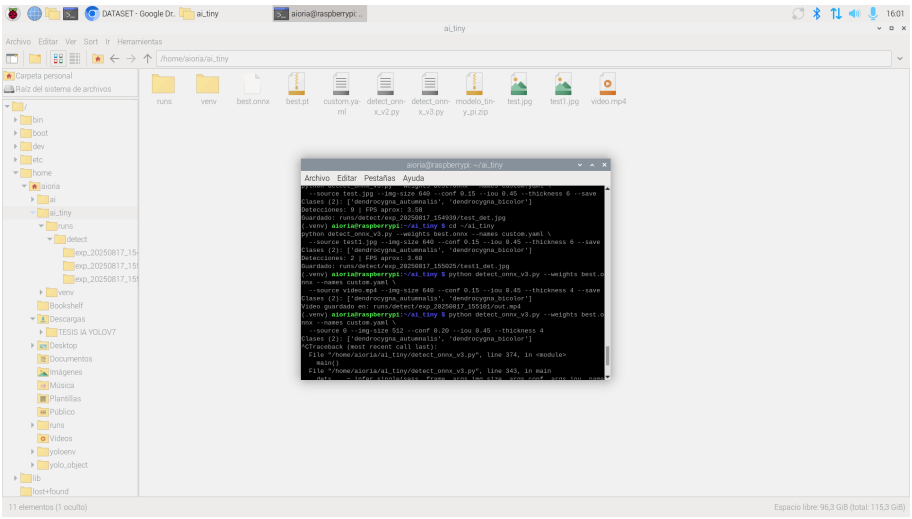


Figura 50. Resultados generados localmente (imagen y vídeo) tras la inferencia con ONNX en la Raspberry Pi Fuente: por A. Macero y J. Ronquillo.

VII-F. Resultados finales en dispositivo

Tras el entrenamiento y la exportación a ONNX, el detector se ejecutó en la Raspberry Pi con cámara y visualización local (Figura 51).

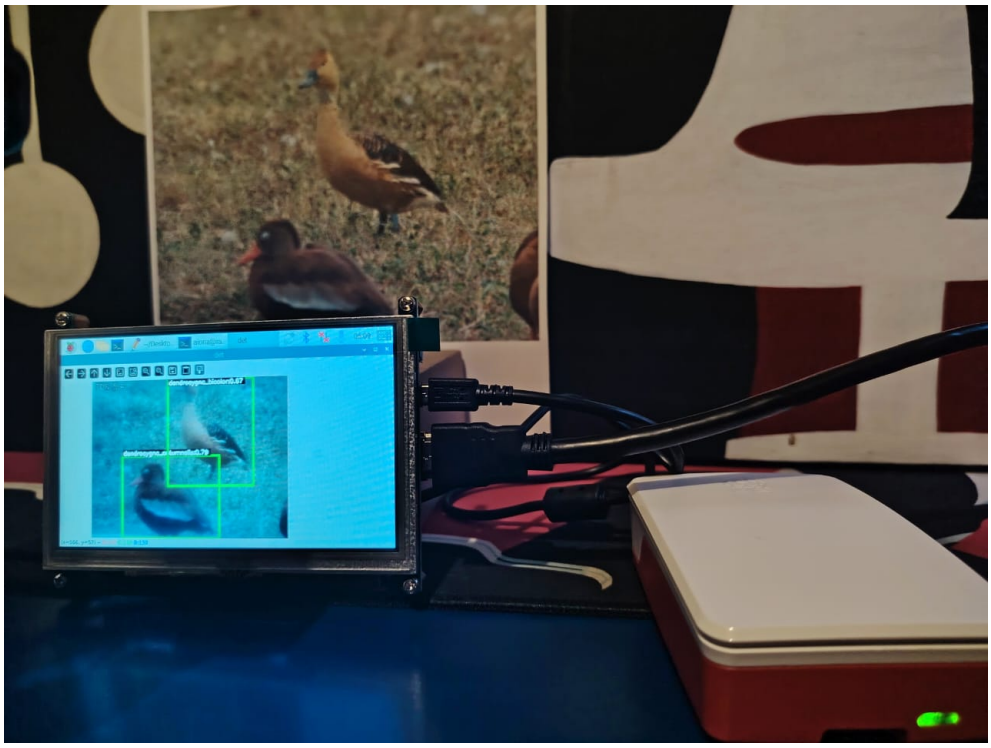


Figura 51. Montaje físico del prototipo Fuente: por A. Macero y J. Ronquillo.

Figura 52. Detección con obstáculos Fuente: por A. Macero y J. Ronquillo.

Detección en video con vegetación flotante: múltiples individuos de Dendrocygna bicolor identificados con confianza alrededor de 0.7; el sistema conserva la clasificación a pesar de oclusiones parciales y salpicaduras (Figura 52).

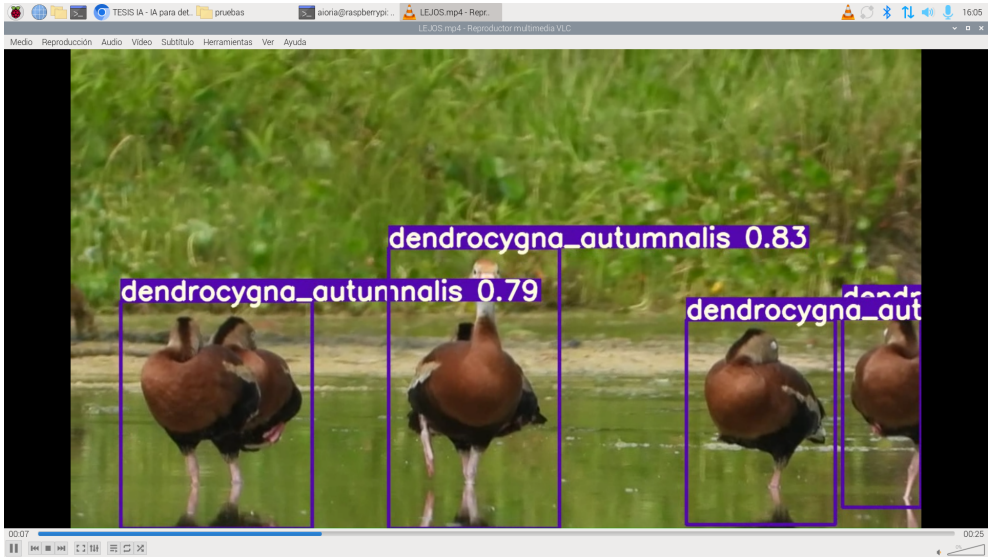


Figura 53. Detecciones simultáneas Fuente: por A. Macero y J. Ronquillo.

Bandada en ribera: detecciones simultáneas de Dendrocygna autumnalis con confianzas altas en varios sujetos; la escena confirma coherencia entre cajas y etiquetas en presencia de grupos cercanos(Figura 53).

Figura 54. Inferencia en vivo con cámara conectada Fuente: por A. Macero y J. Ronquillo.

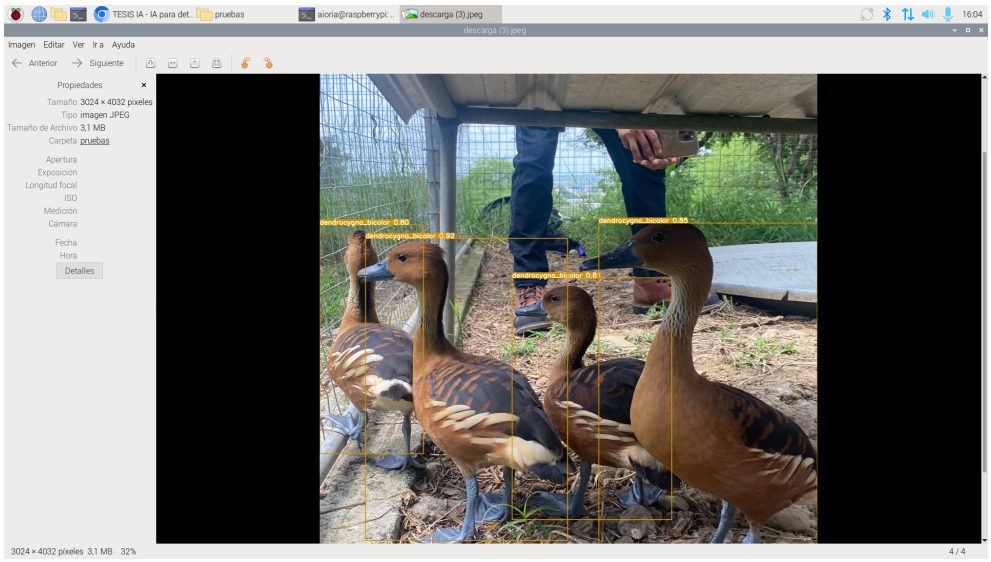


Figura 55. Imagen estática de alta resolución Fuente: por A. Macero y J. Ronquillo.

Múltiples D. bicolor identificados en un entorno controlado; las cajas cubren correctamente la silueta y mantienen confidencias consistentes entre individuos (Figura 55).

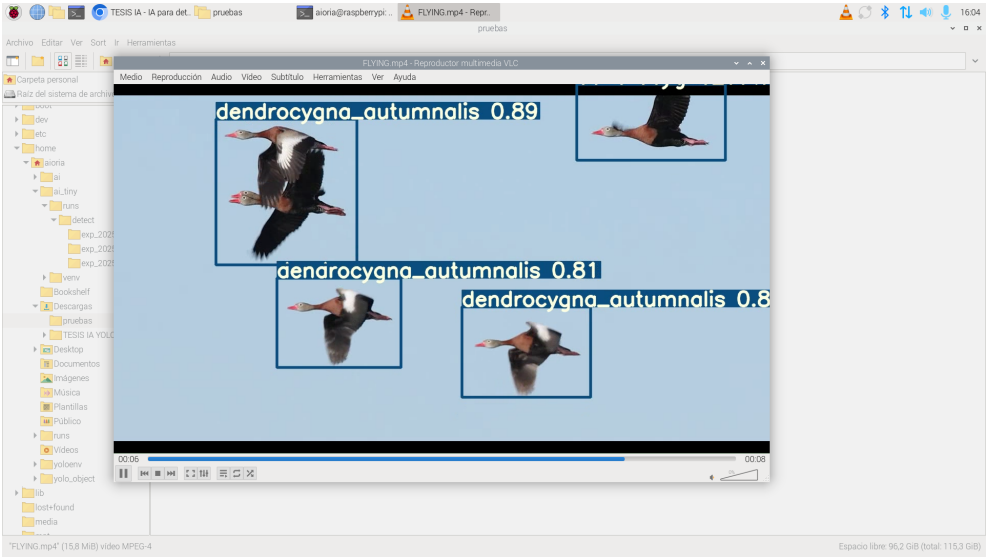


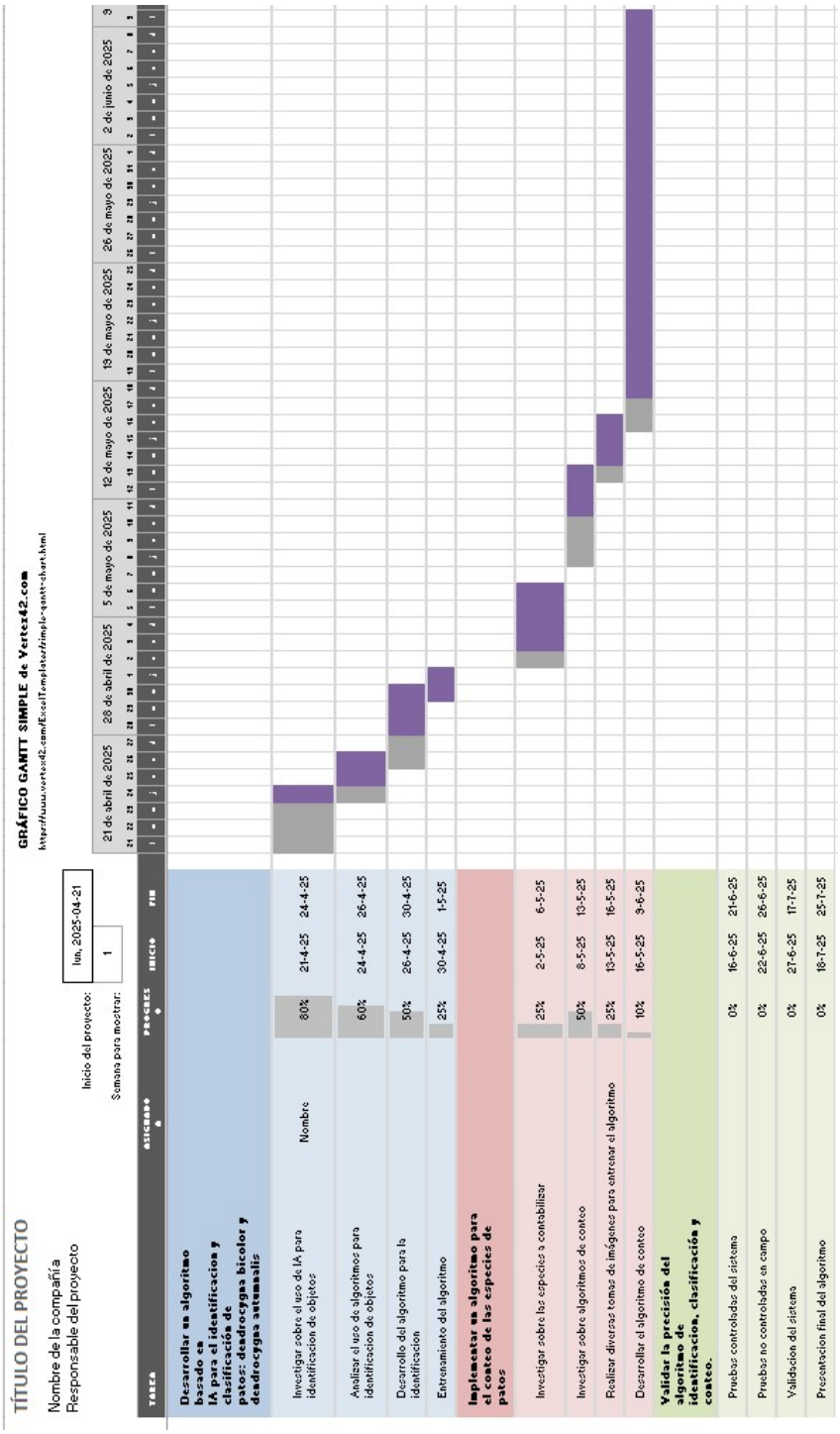
Figura 56. Secuencia de vuelo en cielo despejado Fuente: por A. Macero y J. Ronquillo.

Detecciones de D. autumnalis en movimiento con confianzas alrededor de 0.8–0.9; la uniformidad del fondo favorece la delimitación precisa de las cajas (Figura 56).

VIII. CRONOGRAMA

A continuación se muestra el cronograma de trabajo en la figura IV.

Tabla IV
CRONOGRAMA



IX. PRESUPUESTO

A continuación se muestra el presupuesto del trabajo.

Tabla V
PRESUPUESTO

Ítem	Articulo	P. Unitario USD	Cantidad	P. Total USD
1	Raspberry Pi 5 8GB	$185	1	$185
2	Camara web full HD	$22	1	$22
3	Case oficial Raspberry Pi 5	$25.65	1	$25.65
3	Raspberry display 7”	$69.99	1	$69.99
4	Kit Memoria SD 32GB	$22.50	1	$22.50
5	Cargador Tipo C 27W	$11.50	1	$11.50
6	Disipador aluminio	$2	5	$10
TOTAL				**$344.64**

X. Conclusiones

El sistema demostró que es posible detectar Dendrocygna autumnalis y Dendrocygna bicolor en un dispositivo embebido. La cadena completa, recolección de imágenes desde la web, etiquetado con LabelImg, entrenamiento en Colab, exportación a ONNX y ejecución en Raspberry Pi funcionó de forma integrada y estable.

Se construyó una base de datos de imágenes representativas de ambas especies, recolectadas desde la web con énfasis en condiciones afines a zonas aeroportuarias (agua, vegetación de ribera, grupos en movimiento y variación de distancias). Este material se organizó de forma ordenada en particiones de entrenamiento, validación y prueba, dejando trazabilidad de fuentes, estructura de carpetas y clases.

Se llevó a cabo el preprocesamiento y el etiquetado con LabelImg, generando anotaciones en formato YOLO y garantizando consistencia entre nombres de clase y configuración del proyecto. Sobre esta base se desarrolló un modelo de detección (YOLOv7-tiny) que, en la práctica, resuelve la clasificación a nivel de instancia dentro de cada imagen. La elección de YOLOv7 tiny resultó adecuada por su equilibrio entre velocidad y precisión, lo que permitió operar con fluidez en el campo. En las pruebas se observó un desempeño consistente en escenas variadas; los fallos más comunes se dieron en fondos con texturas, reflejos y en aves muy pequeñas u ocultas.

Se validó el rendimiento del algoritmo mediante pruebas controladas y la lectura de métricas estándar (curvas de aprendizaje, precisión, sensibilidad y mAP), además de evidencias cualitativas con mosaicos y matriz de confusión. Las salidas confirman un comportamiento estable y coherente con el objetivo del proyecto. Con el modelo exportado a ONNX y los scripts de inferencia, la Raspberry Pi ejecuta la detección de manera continua con la webcam Full HD, lo que comprueba la portabilidad y el carácter aplicado de la propuesta.

Más allá de los números, las salidas cualitativas, matriz de confusión, curvas PR y mosaicos, evidencian un comportamiento estable en escenarios reales. El resultado más valioso es, quizá, la portabilidad: el mismo modelo entrenado en Colab se ejecuta con fiabilidad en el hardware objetivo, genera evidencia visual trazable y deja un camino claro para la mejora iterativa.

XI. Recomendaciones

Conviene reforzar el dato antes que complicar la arquitectura: ampliar el conjunto con imágenes del entorno objetivo, incluir “negativos duros” (vegetación, reflejos, aves no objetivo) y añadir más casos difíciles de baja escala, oclusiones y condiciones de luz adversas. Un protocolo de anotación breve, caja ceñida, manejo de solapamientos y criterios para descartar ambigüedades ayuda a mantener consistencia y reduce ruido en el entrenamiento.

En la fase de ajuste, se sugiere afinar aumentos de datos, tamaño de imagen, umbral de decisión y parámetros de supresión de solapamientos, priorizando cambios simples antes de modificar la arquitectura.

Para la operación, es útil exponer un control de umbral en la interfaz y registrar métricas básicas junto con las detecciones para seguimiento y mejora continua. En el dispositivo, se recomienda evaluar optimizaciones como cuantización y revisar el desempeño por resolución de entrada.

Finalmente, mantener una evaluación periódica con un conjunto de validación fijo y retroalimentar el entrenamiento con fallos reales recogidos en campo asegura que el sistema mejore con el uso. Todo ello debería ir acompañado de una verificación básica de licencias en las imágenes de la web y de buenas prácticas de manejo de datos, para que el proyecto sea no solo eficaz y reproducible, sino también responsable.

Referencias

[1] C. McDonald-Gibson., *Planes are striking more birds, but Detective Dove is on the case.* 2024. dirección: https://www.thetimes.com/world/us-world/article/avian-strikes-on-planes-are-rising-but-detective-dove-is-on-the-case-z7xlslg96.

[2] Y. Hu, P. Xing, F. Yang, G. Feng, G. Yang y Z. Zhang, «A birdstrike risk assessment model and its application at Ordos Airport, China,» *Scientific Reports*, vol. 10, n.º 1, pág. 19 627, 2020, ISSN: 2045-2322. DOI: 10.1038/s41598-020-76275-z. dirección: https://doi.org/10.1038/s41598-020-76275-z.

[3] K. Chen, X. Yang y J. Zhu, «A novel probabilistic risk analysis approach for engine bird strike issue applied on the airworthiness regulations,» en *14th International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE 2024)*, vol. 2024, 2024, págs. 330-337. DOI: 10.1049/icp.2024.3454.

[4] M. B. Deshpande, N. D. Patel, M. P M, N. P. Pragashri y P. Ramsamy, «Quantifying Bird Strikes due to Migration from a Machine Learning Perspective,» en *2025 13th International Conference on Intelligent Control and Information Processing (ICICIP)*, 2025, págs. 115-121. DOI: 10.1109/ICICIP64458.2025.10898117.

[5] R. Brunner., *Flight with 'Fumes in the Cabin' Makes Emergency Landing in Boston After Striking a Bird.* 2025. dirección: https://people.com/flight-with-fumes-in-cabin-makes-emergency-landing-in-boston-after-striking-bird-11722802.

[6] C. G. Calabrese, B. R. Molesworth, J. Hatfield y E. Slavich, «Effects of the Federal Aviation Administration's Compliance Program on aircraft incidents and accidents,» *Transportation Research Part A: Policy and Practice*, vol. 163, págs. 304-319, 2022, ISSN: 0965-8564. DOI: https://doi.org/10.1016/j.tra.2022.07.016. dirección: https://www.sciencedirect.com/science/article/pii/S0965856422001938.

[7] F. J. A. Andrade, *Caso Aeropuerto de Guayaquil: El problema con las aves se conoce desde hace 12 años*, 2022. dirección: https://www.expreso.ec/guayaquil/caso-aeropuerto-error-reconoce-persiste-temor-aves-141369.html.

[8] Direccion General de Aviación Civil, *Nuevo avistamiento de aves en el Islote el Palmar e Isla Celeste para dispersar aves en zonas de aproximación del Aeropuerto de Guayaquil*, 2024. dirección: https://www.aviacioncivil.gob.ec/nuevo-avistamiento-de-aves-en-el-islote-el-palmar-e-isla-celeste-para-aviar-en-zonas-de-aproximacion-del-aeropuerto-de-guayaquil/.

[9] PRIMICIAS, *Expertos extranjeros evaluarán peligro aviario en aeropuerto de Guayaquil*, 2022. dirección: https://www.primicias.ec/noticias/sociedad/guayaquil-aeropuerto-peligro-aviario-expertos-internacionales/.

[10] U. F. W. Service, *Black-bellied Whistling Duck*, 2022. dirección: https://www.fws.gov/species/black-bellied-whistling-duck-dendrocygna-autumnalis.

[11] G. el comercio", *.ªves impactan dos aviones en Guayaquil; aerolínea solicita medidas a la DGAC"*, 2022. dirección: https://www.elcomercio.com/actualidad/guayaquil/aves-impactan-aviones-guayaquil-solicita-medidas.html.

[12] U. D. O. AGRICULTURE, *NWRC Research Areas: Aviation Safety*, 2025. dirección: https://www.aphis.usda.gov/national-wildlife-programs/nwrc/research-areas/aviation-safety.

[13] QUIPORT, *Faunaetus celebrates 10 years managingresponsibly wildlife at the Quito airport*, 2023. dirección: https://www.quiport.com/faunaetus-celebrates-10-years-managingresponsibly-wildlife-at-the-quito-airport/.

[14] J. D. James y J. E. Thompson., «Black-bellied Whistling-Duck (Dendrocygna autumnalis).,» 2020.

[15] Price, K., *Protegiendo las aves mediante inteligencia artificial*, 2024. dirección: https://www.nationalgeographic.es/animales/2024/03/aves-protegidas-mediante-inteligencia-artificial.

[16] X. Zhao y C. Chen, «Multi-Scale Localization and Attention Mechanism for Fine-Grained Image Classification,» en *2024 International Conference on Image Processing, Computer Vision and Machine Learning (ICICML)*, 2024, págs. 1173-1178. DOI: 10.1109/ICICML63543.2024.10958076.

[17] R. N. Tripathi, K. Agarwal, V. Tripathi, R. Badola y S. A. Hussain, «Conservation in action: Cost-effective UAVs and real-time detection of the globally threatened swamp deer (Rucervus duvaucelii),» *Ecological Informatics*, vol. 85, pág. 102 913, 2025, ISSN: 1574-9541. DOI: https://doi.org/10.1016/j.ecoinf.2024.102913. dirección: https://www.sciencedirect.com/science/article/pii/S1574954124004552.

[18] D. F. S. T. S. y L. Rico., «Black-bellied Whistling-Duck (Dendrocygna autumnalis).,» 2006.

[19] L. Kuhlane, D. Brown y M. Marais, «Real- Time Detecting and Tracking of Squids Using YOLOv5,» en *2023 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)*, 2023, págs. 1-5. DOI: 10.1109/icABCD59051.2023.10220521.

[20] B. Eren, M. H. Demir y S. Mistikoglu, «Recent developments in computer vision and artificial intelligence aided intelligent robotic welding applications,» *The International Journal of Advanced Manufacturing Technology*, vol. 126, n.º 11, págs. 4763-4809, 2023, ISSN: 1433-3015. DOI: 10.1007/s00170-023-11456-4. dirección: https://doi.org/10.1007/s00170-023-11456-4.

[21] L. Altringer y J. Begier M.J. Washburn, «Estimating the impact of airport wildlife hazards management on realized wildlife strike risk.,» 2024.

[22] J. Juračka, J. Chlebek y V. Hodaň, «Bird strike as a threat to aviation safety,» *Transportation Research Procedia*, vol. 59, págs. 281-291, 2021, 10th International Conference on Air Transport – INAIR 2021, TOWARDS AVIATION REVIVAL, ISSN: 2352-1465. DOI: https://doi.org/10.1016/j.trpro.2021.11.120. dirección: https://www.sciencedirect.com/science/article/pii/S2352146521008826.

[23] A. U. U. K. Filiz Ekici Öner Gümüş, «An investigation of bird strike cases in the aviation sector with a novel approach within the context of the principal-agent phenomenon: Bird strikes and insurance in the USA.,» 2023.

[24] Y. Ma, J. Liu, F. Yi et al., «AI vs. Human–differentiation analysis of scientific content generation,» *arXiv preprint arXiv:2301.10416*, 2023.

[25] Bowen, Z., Huacai, L. Shengbo y Z., «Night target detection algorithm based on improved YOLOv7.,» 2024.

[26] H. Y. Chen X Pu H, «An Efficient Method for Monitoring Birds Based on Object Detection and Multi-Object Tracking Networks.,» 2023.

[27] Europair, *¿Cómo afecta el sector de la aviación en la economía y el turismo internacional?* https://latam.europair.com/blog/como-afecta-el-sector-de-la-aviacion-en-la-economia-y-el-turismo-internacional, [En línea; accedido: mayo 13, 2025], 2023.

[28] J. Wiltshire y A. Jaimurzina, *Transporte aéreo como motor del desarrollo sostenible en América Latina y el Caribe: retos y propuestas de política*, https://repositorio.cepal.org/server/api/core/bitstreams/3b1e7acc-2c7a-4968-9f5d-4c548cbf455f/content, [En línea; accedido: mayo 13, 2025], 2017.

[29] I. A. T. A. (IATA), *El transporte aéreo genera 65,5 millones de empleos y aporta 2,7 billones de dólares a la economía mundial*, https://www.iata.org/contentassets/d3dee4898f8649cc876dd8d4f3a92231/2018-10-02-01-sp.pdf, [En línea; accedido: 13-may-2025], 2018.

[30] Ferrovial, *Aviones: qué son, cómo vuelan, qué tipos hay*, https://www.ferrovial.com/es/recursos/aviones/, [En línea; accedido: 13-may-2025], mayo de 2025.

[31] S. Prodel, *Aerodinámica*, https://www.prodel.es/subareas/aerodinamica/, [En línea; accedido: 13-may-2025].

[32] Euronews, *La policía surcoreana registra las oficinas de Jeju Air mientras prosigue la investigación*, https://es.euronews.com/2025/01/02/la-policia-surcoreana-registra-las-oficinas-de-jeju-air-mientras-prosigue-la-investigacion, [En línea; accedido: 13-may-2025], ene. de 2025.

[33] U. Fish y W. Service, *Threats to Birds: Collisions-Aircraft*, https://www.fws.gov/story/threats-birds-collisions-aircraft, [En línea; accedido: 13-may-2025], mayo de 2025.

[34] J. McCarthy, M. L. Minsky, N. Rochester y C. E. Shannon, *A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence*, https://www.cs.utexas.edu/users/novak/ai/dartmouth.html, Unpublished manuscript, ago. de 1955.

[35] R. C. Gonzalez y R. E. Woods, *Digital Image Processing*, 4th. New York, NY: Pearson, 2018, [En línea; accedido: 13-may-2025], ISBN: 9780133356724.

[36] Y. LeCun, Y. Bengio y G. Hinton, «Deep learning,» *nature*, vol. 521, n.º 7553, págs. 436-444, 2015.

[37] R. C. Gonzalez y R. E. Woods, *Digital Image Processing*, 3rd. Prentice Hall, 2008, ISBN: 978-0131687288.

[38] Y. LeCun, Y. Bengio y G. Hinton, «Deep Learning,» *Nature*, vol. 521, n.º 7553, 436–444, 2015. DOI: 10.1038/nature14539.

[39] I. Goodfellow, Y. Bengio y A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016.

[40] C. O'Neil, *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. New York, NY: Crown Publishing Group, 2016.

[41] J. Redmon, S. Divvala, R. Girshick y A. Farhadi, «You Only Look Once: Unified, Real-Time Object Detection,» en *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. dirección: https://arxiv.org/abs/1506.02640.

[42] Python Software Foundation, *Python Language Reference, version 3.11*, Python Software Foundation, 2024. dirección: https://www.python.org.

[43] Google Research, *Google Colaboratory*, https://colab.research.google.com/, Accedido el 16 de agosto de 2025, 2025.

[44] E. Upton y G. Halfacree, *Raspberry Pi User Guide*, 4th. Wiley, 2016, ISBN: 978-1119264369.

[45] Debian Project, *Debian GNU/Linux – The Universal Operating System*, https://www.debian.org/, Accedido el 16 de agosto de 2025, 2025.

[46] *Official Raspberry Pi 7 Touch Display*, https://www.raspberrypi.com/products/raspberry-pi-touch-display/, Accessed: 2025-07-16, 2024.

[47] USB Implementers Forum, *USB Video Class (UVC) Specification*, https://www.usb.org/document-library/video-class-v15-document-set, Consulta: 18-08-2025.

ANEXO A

CÓDIGO PARA EL ENTRENAMIENTO DEL MODELO EN GOOGLE COLAB

```
# -*- coding: utf-8 -*-
"""YOLOv7_tiny_on_CustomData.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1wfWIT-rz1mf8yZAq1cQNMgovWaKSgwwm

# YOLOv7tiny en tu propio dataset (Colab)
"""

# Limpio estado y actualizo herramientas b sicas
!nvidia-smi -L || true
!python -V
!pip -q install --upgrade pip wheel setuptools

# Commented out IPython magic to ensure Python compatibility.
# %cd /content
!rm -rf yolov7
!git clone https://github.com/WongKinYiu/yolov7.git
# %cd /content/yolov7

# Instala deps del repo (sin forzar torch/cuda: Colab ya trae)
!pip -q install -r requirements.txt

# Commented out IPython magic to ensure Python compatibility.
# %cd /content
# %pip install -q gdown
!gdown --fuzzy "https://drive.google.com/file/d/18PJ2KzfQMP4h4V7-ckJA4TKyytXs6cMJ/view?usp=
    sharing" -O data_yolov7.zip
!unzip -q data_yolov7.zip -d data_yolov7
!rm data_yolov7.zip

# Commented out IPython magic to ensure Python compatibility.
# %cd /content
!find data_yolov7 -maxdepth 3 -type d | sed 's#^/content/##' | sort
!find data_yolov7 -maxdepth 2 -type f | head -n 20
!find /content/data_yolov7 -type f \( -iname "*.jpg" -o -iname "*.jpeg" -o -iname "*.png" \) |
    wc -l

import os, re, glob, yaml

ROOT = "/content/data_yolov7"

IMG_EXTS = (".jpg", ".jpeg", ".png", ".bmp", ".tif", ".tiff")
LBL_EXT  = ".txt"
train_tokens = ["train","training","trn"]
val_tokens   = ["val","valid","validation","validacion","validadcion","valid-set"]

```

```
def has_images(d):
    for f in os.scandir(d):
        if f.is_file() and f.name.lower().endswith(IMG_EXTS):
            return True
    return False

def count_images(d):
    return sum(1 for p in glob.glob(os.path.join(d, "*")) if p.lower().endswith(IMG_EXTS))

def count_txt(d):
    return sum(1 for p in glob.glob(os.path.join(d, "*.txt")))

#  listar todos los dirs que contienen im genes
img_dirs = []
for cur, dirs, files in os.walk(ROOT):
    if has_images(cur):
        img_dirs.append(cur)

# clasificar candidatos de train/val por tokens
def pick_best(cands, tokens):
    # prioriza dirs que incluyan token y 'image' en la ruta
    ranked = []
    for d in cands:
        path_low = d.lower()
        hits = sum(t in path_low for t in tokens)
        is_images = ("image" in path_low or "/img" in path_low or "\\img" in path_low)
        score = (hits>0, is_images, count_images(d))
        ranked.append((score, d))
    ranked.sort(reverse=True)
    return ranked[0][1] if ranked else None

train_img_dir = pick_best(img_dirs, train_tokens)
val_img_dir   = pick_best(img_dirs, val_tokens)

# Si no los encontr  por tokens, intenta patrones comunes
if not train_img_dir or not val_img_dir:
    # busca dirs llamados exactamente 'images' que tengan subcarpeta 'train'/'val' arriba
    for d in img_dirs:
        low = d.lower()
        if any(t in low for t in train_tokens) and ("image" in low or "/img" in low):
            train_img_dir = train_img_dir or d
        if any(t in low for t in val_tokens) and ("image" in low or "/img" in low):
            val_img_dir = val_img_dir or d

#  localizar labels correspondientes
def guess_labels_dir(img_dir):
    if not img_dir: return None
    # caso t pico .../images/... -> .../labels/...
    cand = img_dir.lower().replace("/images", "/labels").replace("\\images","\\labels")
    # si cambi  algo y existe
    if cand != img_dir.lower() and os.path.isdir(cand):
        # devuelve con las may sculas reales (no solo lower)
```

```
        real = re.sub(r"/labels", "/labels", img_dir).replace("/images","/labels")
        return real if os.path.isdir(real) else cand
    # buscar cualquier 'labels' con los mismos tokens de train/val
    parent = os.path.dirname(img_dir)
    candidates = []
    for cur, dirs, files in os.walk(os.path.commonpath([ROOT, parent])):
        if os.path.basename(cur).lower() == "labels":
            # escoger subcarpeta train/val si existe
            # ejemplo: .../labels/train     .../labels/val
            for t in train_tokens+val_tokens:
                p = os.path.join(cur, t)
                if os.path.isdir(p):
                    candidates.append(p)
            candidates.append(cur)
    # elige el que tenga m s .txt
    best = None
    best_cnt = -1
    for c in candidates:
        cnt = count_txt(c)
        if cnt > best_cnt:
            best_cnt, best = cnt, c
    return best

train_lbl_dir = None
val_lbl_dir   = None

# Suponemos estructura "yolo" cl sica: labels replican estructura de images
if train_img_dir:
    # intenta el exacto espejo
    train_lbl_dir = guess_labels_dir(train_img_dir)
if val_img_dir:
    val_lbl_dir = guess_labels_dir(val_img_dir)

print(">> Candidatos detectados:")
print("  train_img_dir:", train_img_dir)
print("  val_img_dir  :", val_img_dir)
print("  train_lbl_dir:", train_lbl_dir)
print("  val_lbl_dir  :", val_lbl_dir)

# Comprobaciones m nimas
def ok_dir(d): return d and os.path.isdir(d)
def has_some(d, exts):
    return any(f.lower().endswith(exts) for f in os.listdir(d)) if ok_dir(d) else False

assert ok_dir(train_img_dir) and ok_dir(val_img_dir), "No pude localizar carpetas de IM GENES
    train/val."
assert has_some(train_img_dir, IMG_EXTS) and has_some(val_img_dir, IMG_EXTS), "No veo im genes
     dentro de esas carpetas."

# Aviso si no hay labels    pareadas    (YOLO form)
if not (ok_dir(train_lbl_dir) and ok_dir(val_lbl_dir)):
```

```
    print("       No localic  carpetas de labels claras. Si tus anotaciones est n en otro
        formato, YOLOv7 no podr  entrenar hasta convertirlas.")
else:
    # muestreo r pido: cu ntos .txt hay
    print("labels train:", count_txt(train_lbl_dir), " | labels val:", count_txt(val_lbl_dir))

# 3) Crear el YAML para YOLO
NAMES = ["dendrocygna_autumnalis","dendrocygna_bicolor"]  # <-- ajusta aqu  si cambian tus
    clases

yaml_dict = {
    "train": train_img_dir,
    "val":   val_img_dir,
    "nc":    len(NAMES),
    "names": NAMES
}

os.makedirs("/content/yolov7/data", exist_ok=True)
out_yaml = "/content/yolov7/data/custom_tiny.yaml"
with open(out_yaml, "w") as f:
    yaml.safe_dump(yaml_dict, f, sort_keys=False)

print("\ n   Escrib ", out_yaml)
print(yaml_dict)
print("\nIm genes train:", len(glob.glob(os.path.join(train_img_dir, "*"))))
print("Im genes val  :", len(glob.glob(os.path.join(val_img_dir, "*"))))

# Commented out IPython magic to ensure Python compatibility.
# %cd /content/yolov7
!ls -1 cfg/training | grep -i tiny || true
!ls -l data | sed -n '1,200p'
!sed -n '1,200p' data/custom_tiny.yaml

"""## ENTRENAMIENTO YOLO TINY
"""

# Commented out IPython magic to ensure Python compatibility.
# %cd /content/yolov7
!python train.py --weights yolov7-tiny.pt
--cfg cfg/training/yolov7-tiny.yaml
--data /content/yolov7/data/custom_tiny.yaml
--hyp data/hyp.scratch.tiny.yaml
--epochs 300
--batch-size 16
--img-size 640 640
--device 0
--workers 8
--name exp_tiny

!pip uninstall -y wandb

# Commented out IPython magic to ensure Python compatibility.
```

```
# %cd /content/yolov7

# Parchea train.py para forzar weights_only=False
import io, re, os, sys, textwrap

p = "train.py"
s = open(p, "r", encoding="utf-8").read()
# hay dos usos t picos en train.py: el run_id y el ckpt
s = s.replace("torch.load(weights, map_location=device).get('wandb_id')",
              "torch.load(weights, map_location=device, weights_only=False).get('wandb_id')")
s = s.replace("torch.load(weights, map_location=device)",
              "torch.load(weights, map_location=device, weights_only=False)")
open(p, "w", encoding="utf-8").write(s)
print("    parcheado", p)

# Parchea models/experimental.py (lo usa detect/test al cargar pesos)
p = "models/experimental.py"
s = open(p, "r", encoding="utf-8").read()
s = s.replace("torch.load(w, map_location=map_location)",
              "torch.load(w, map_location=map_location, weights_only=False)")
open(p, "w", encoding="utf-8").write(s)
print("    parcheado", p)

# (opcional) Desactiva W&B y entrena
import os
os.environ["WANDB_DISABLED"] = "true"
os.environ["WANDB_SILENT"] = "true"

!python train.py \
  --weights yolov7-tiny.pt \
  --cfg cfg/training/yolov7-tiny.yaml \
  --data /content/yolov7/data/custom_tiny.yaml \
  --hyp data/hyp.scratch.tiny.yaml \
  --epochs 300 --batch-size 16 --img-size 640 640 \
  --device 0 --workers 8 --name exp_tiny2

"""## AFTER TRAINING"""

# Commented out IPython magic to ensure Python compatibility.
# %cd /content/yolov7
!cp -v runs/train/exp_tiny22/weights/best.pt runs/train/exp_tiny22/weights/best_snapshot.pt
!cp -v runs/train/exp_tiny22/weights/last.pt runs/train/exp_tiny22/weights/last_snapshot.pt

!tail -n 20 runs/train/exp_tiny22/results.txt
from utils.plots import plot_results
plot_results(save_dir='runs/train/exp_tiny22')  # genera results.png

# Commented out IPython magic to ensure Python compatibility.
# %cd /content/yolov7
import os, glob, pprint, pathlib
exps = sorted(glob.glob('runs/train/exp*'), key=os.path.getmtime)
EXP = exps[-1] if exps else 'runs/train/exp_tiny22'
```

```
print("Usando EXP:", EXP)
!ls -lh "$EXP" | sed -n '1,200p'
!tail -n 20 "$EXP/results.txt"

# Commented out IPython magic to ensure Python compatibility.
# %cd /content/yolov7
import os, glob
exps = sorted(glob.glob('runs/train/exp*'), key=os.path.getmtime)
EXP = exps[-1] if exps else 'runs/train/exp_tiny22'
print("Usando EXP:", EXP)
!ls -lh "$EXP" | sed -n '1,200p'
!tail -n 5 "$EXP/results.txt"
# %cd /content/yolov7
import os, glob
exps = sorted(glob.glob('runs/train/exp*'), key=os.path.getmtime)
EXP = exps[-1] if exps else 'runs/train/exp_tiny22'
print("Usando EXP:", EXP)
!ls -lh "$EXP" | sed -n '1,200p'
!tail -n 5 "$EXP/results.txt"

import numpy as np, matplotlib.pyplot as plt, os, re, pathlib, csv

results_txt = pathlib.Path(EXP) / "results.txt"
assert results_txt.exists(), f"No encuentro {results_txt}"

rows=[]
for ln in open(results_txt):
    # extrae todos los n meros de cada l nea (ignora texto)
    nums = [float(x) for x in re.findall(r"[-+]?\d*\.\d+|\d+", ln)]
    if len(nums)>=8:   # suele haber >8 columnas por  poca
        rows.append(nums)
data = np.array(rows)

# Heur stica:  ltimas  4 columnas = [P, R, mAP50, mAP50-95]
P,R,m50,m5095 = data[:,-4], data[:,-3], data[:,-2], data[:,-1]

# Primeras 4     p rdidas de train [box, obj, cls, total] (puede variar por repo)
box,obj,cls,total = data[:,1], data[:,2], data[:,3], data[:,4]

fig,axs=plt.subplots(2,2,figsize=(12,8))
axs = axs.ravel()
axs[0].plot(box,label='box'); axs[0].plot(obj,label='obj'); axs[0].plot(cls,label='cls'); axs
    [0].plot(total,label='total'); axs[0].set_title("P rdidas (train)"); axs[0].legend(); axs
    [0].set_xlabel(" poca ")
axs[1].plot(P);     axs[1].set_title("Precision");    axs[1].set_xlabel(" poca ")
axs[2].plot(R);     axs[2].set_title("Recall");       axs[2].set_xlabel(" poca ")
axs[3].plot(m50,label="mAP@0.5"); axs[3].plot(m5095,label="mAP@0.5:0.95"); axs[3].legend(); axs
    [3].set_title("mAP"); axs[3].set_xlabel(" poca ")
plt.tight_layout()
out_png = f"{EXP}/results_report.png"
plt.savefig(out_png, dpi=200, bbox_inches="tight")
print("Guardado:", out_png)
```

```
300
301 # Commented out IPython magic to ensure Python compatibility.
302 # Borra caches
303 !rm -f /content/data_yolov7/data_yolov7/labels/train.cache
304 !rm -f /content/data_yolov7/data_yolov7/labels/val.cache
305
306 # Corre la evaluaci n (val) y genera curvas PR/F1/P/R (y a veces conf. matrix)
307 # %cd /content/yolov7
308 !python test.py \
309   --weights runs/train/exp_tiny22/weights/best.pt \
310   --data /content/yolov7/data/custom_tiny.yaml \
311   --img-size 640 \
312   --conf-thres 0.001 --iou-thres 0.65 \
313   --task val \
314   --project runs/test --name exp_tiny_eval --exist-ok
315
316 # Ver qu  se gener
317 !ls -lh runs/test/exp_tiny_eval | sed -n '1,200p'
318
319 """# PRUEBAS"""
320
321 # Commented out IPython magic to ensure Python compatibility.
322 # %cd /content/yolov7
323 EXP = "runs/train/exp_tiny22"    # c mbialo si tu experimento se llama distinto
324 WEIGHTS = f"/content/yolov7/{EXP}/weights/best.pt"
325 print(WEIGHTS)
326
327 from google.colab import drive; drive.mount('/content/drive')
328 IMG = "/content/drive/MyDrive/TESIS IA YOLOV7/DATASET/TEST/Imagen de WhatsApp 2025-08-15 a las
        12.01.21_500a4934.jpg"   # ajusta la ruta
329
330 # Commented out IPython magic to ensure Python compatibility.
331 # Sube el grosor de las cajas que detect.py pasa a plot_one_box
332 # %cd /content/yolov7
333
334 import re, pathlib
335 p = pathlib.Path("detect.py")
336 s = p.read_text(encoding="utf-8")
337
338 # Reemplaza TODOS los "line_thickness=NUM" por 12 (aj stalo si quieres m s)
339 s2 = re.sub(r"line_thickness\s*=\s*\d+", "line_thickness=4", s)
340
341 # Si el c digo no ten a ese argumento expl cito, no habr  cambios (te avisamos)
342 if s2 == s:
343     print("No encontr  'line_thickness=NUM' en detect.py; nada que cambiar.")
344 else:
345     p.write_text(s2, encoding="utf-8")
346     print("       Cambi  line_thickness a 12 en detect.py")
347
348 WEIGHTS="/content/yolov7/runs/train/exp_tiny22/weights/best.pt"
349 IMG="/content/drive/MyDrive/TESIS IA YOLOV7/DATASET/TEST/PATO3.jpg"  #     pon aqu  tu imagen
350
```

```
!python detect.py \
  --weights "$WEIGHTS" \
  --source  "$IMG" \
  --img-size 640 \
  --conf-thres 0.25 --iou-thres 0.45 \
  --project runs/detect --name demo_img --exist-ok

# Ver el resultado
from IPython.display import Image, display
display(Image(filename="runs/detect/demo_img/" + IMG.split("/")[-1]))

WEIGHTS="/content/yolov7/best.pt"
VID="/content/drive/MyDrive/TESIS IA YOLOV7/DATASET/TEST/OTRO.mp4"  #     tu video

!python detect.py \
  --weights "$WEIGHTS" \
  --source  "$VID" \
  --img-size 640 \
  --conf-thres 0.25 --iou-thres 0.45 \
  --project runs/detect --name demo_vid --exist-ok

# Reproducir el .mp4 resultante
import glob
from IPython.display import Video, display
out = glob.glob("runs/detect/demo_vid/*.mp4")
display(Video(out[0], embed=True)) if out else print("No se encontr  el MP4 de salida")

"""# SAVE"""

EXP = "runs/train/exp_tiny22"  # <-- AJUSTA AQU  SI ES OTRA
BEST = f"{EXP}/weights/best.pt"
YAML = "/content/yolov7/data/custom_tiny.yaml"

import os, glob, textwrap, datetime, pathlib
print("EXP:", EXP, "\nExiste best.pt:", os.path.isfile(BEST))

import glob, os, pathlib, pprint

# Ajustar si usaste otro --name; esto busca el  m  s  reciente automticamente
cands = sorted(glob.glob("runs/detect/demo_vid*"))
RUN_DIR = cands[-1] if cands else "runs/detect/exp"
print("Usando carpeta:", RUN_DIR)
!ls -lah "$RUN_DIR"

# Commented out IPython magic to ensure Python compatibility.
from google.colab import drive, files
drive.mount('/content/drive')
STAMP = datetime.datetime.now().strftime("%Y%m%d_%H%M")
DEST = f"/content/drive/MyDrive/yolov7_backups/{pathlib.Path(EXP).name}_{STAMP}"
os.makedirs(DEST, exist_ok=True)

# copia la carpeta completa del experimento
```

```
!rsync -ah --progress "{EXP}/" "{DEST}/"

# adem s, exporta a ONNX y empaqueta lo importante
# %cd /content/yolov7
!python export.py --weights "{BEST}" --img-size 640 640 --device cpu --simplify
PKG = f"/content/modelo_tiny_{STAMP}.zip"
!zip -qr "{PKG}" "{BEST}" "{YAML}" "{EXP}/results.txt" "{EXP}/results_report.png" "runs/train/
    exp_tiny22/weights/best.onnx"
!cp "{PKG}" "{DEST}/"
print("Backup en:", DEST)

import datetime, os
STAMP = datetime.datetime.now().strftime("%Y%m%d_%H%M")
PKG = f"/content/modelo_tiny_{STAMP}.zip"
!zip -qr "{PKG}" \
  runs/train/exp_tiny22/weights/best.pt \
  runs/train/exp_tiny22/results.txt \
  runs/train/exp_tiny22/results_report.png \
  /content/yolov7/data/custom_tiny.yaml \
  runs/test/exp_tiny_eval

print("ZIP listo:", PKG)

from google.colab import drive
drive.mount('/content/drive')
!mkdir -p /content/drive/MyDrive/yolov7_backups/
!cp -v "{PKG}" /content/drive/MyDrive/yolov7_backups/

from google.colab import drive
drive.mount('/content/drive')



"""# ERRORES"""

# Quita cosas que est n meti ndose en medio
!pip uninstall -y tensorflow tensorflow-cpu tensorflow-intel jax jaxlib tb-nightly tensorboard
    tensorboard-data-server tensorboard-plugin-wit

# Instala una versi n de TensorBoard estable que funciona con PyTorch SummaryWriter sin TF
!pip install -U tensorboard==2.14.1

# Verificaci n r pida
try:
    from torch.utils.tensorboard import SummaryWriter
    print("    torch.utils.tensorboard OK")
except Exception as e:
    print("    torch.utils.tensorboard fall :", e)

# Commented out IPython magic to ensure Python compatibility.
# %cd /content/yolov7
!ls -l data/hyp.scratch.tiny.yaml || echo "       NO existe data/hyp.scratch.tiny.yaml"
```

```
453
454 !HYP="data/hyp.scratch.tiny.yaml"
455
456 !ls -l /content/yolov7/data/custom_tiny.yaml
457
458 # Commented out IPython magic to ensure Python compatibility.
459 # %%writefile /content/yolov7/utils/tb_fallback.py
460 # # Fallback m nimo para evitar dependencia de tensorboard/tensorflow
461 # class SummaryWriter:
462 #     def __init__(self, *args, **kwargs):
463 #         pass
464 #     def add_scalar(self, *args, **kwargs):
465 #         pass
466 #     def add_scalars(self, *args, **kwargs):
467 #         pass
468 #     def add_image(self, *args, **kwargs):
469 #         pass
470 #     def add_histogram(self, *args, **kwargs):
471 #         pass
472 #     def add_graph(self, *args, **kwargs):
473 #         pass
474 #     def close(self):
475 #         pass
476 #
477
478 !grep -n "tensorboard" /content/yolov7/train.py || echo "OK: train.py no usa torch.utils.
    tensorboard"
479 !sed -i 's/from torch.utils.tensorboard import SummaryWriter/from utils.tb_fallback import
    SummaryWriter/' /content/yolov7/train.py
480
481 """##      Validar (mAP)"""
482
483 !python test.py \
484   --weights runs/train/{EXPNAME}/weights/best.pt \
485   --data {DATA_YAML} \
486   --img-size {TRAIN_IMG} \
487   --batch-size 16 \
488   --task val \
489   --device 0 \
490   --name {EXPNAME} \
491   --project runs/test \
492   --v5-metric
493
494 """## Graficar `results.png` del entrenamiento"""
495
496 # Commented out IPython magic to ensure Python compatibility.
497 from pathlib import Path
498 from utils.plots import plot_results
499 from IPython.display import Image, display
500
501 # %cd /content/yolov7
502 save_dir = Path(f"runs/train/{EXPNAME}")
```

```
plot_results(save_dir=save_dir)
display(Image(filename=save_dir/"results.png"))

"""## Prueba r pida con 'detect.py' sobre '/content/val'"""

!python detect.py \
  --weights runs/train/{EXPNAME}/weights/best.pt \
  --source /content/val \
  --img-size {TRAIN_IMG} \
  --conf 0.25 --iou 0.45 \
  --device 0 \
  --name {EXPNAME} \
  --project runs/detect \
  --exist-ok

"""## Exportar a ONNX + crear ZIP para Raspberry Pi"""

# Exportar a ONNX y empaquetar
!python export.py \
  --weights runs/train/{EXPNAME}/weights/best.pt \
  --img-size {TRAIN_IMG} {TRAIN_IMG} \
  --device cpu \
  --simplify

# Empaquetar modelo + yaml
!zip -j /content/modelo_pi_tiny.zip \
  runs/train/{EXPNAME}/weights/best.pt \
  runs/train/{EXPNAME}/weights/best.onnx \
  {DATA_YAML} \
  || true

!ls -lh /content/modelo_pi_tiny.zip || true

"""##
# RETOMAR
"""

# Commented out IPython magic to ensure Python compatibility.
# %cd /content/yolov7
!python train.py --resume

# Commented out IPython magic to ensure Python compatibility.

# %cd /content/yolov7
!python test.py --weights "{BEST}" --data "{YAML}" --img-size 640 --task val \
               --project runs/test --name exp_tiny_eval --exist-ok

"""# NUEVA RUN"""

# Commented out IPython magic to ensure Python compatibility.
# GPU en Configuraci n de ejecuci n: activada (si no, igual sirve en CPU pero m s lento)
# %cd /content
```

```
!git clone https://github.com/WongKinYiu/yolov7.git
# %cd /content/yolov7

# Paquetes m nimos para inferencia (evita conflictos)
!pip install -q "numpy<2" matplotlib==3.8.4 opencv-python-headless==4.8.0.76

from google.colab import drive
drive.mount('/content/drive')

# Commented out IPython magic to ensure Python compatibility.
!cp "/content/drive/MyDrive/yolov7_backups/modelo_tiny_pi.zip" /content/
# %cd /content
!unzip -o modelo_tiny_pi.zip -d /content/yolov7  # dejar  best.pt y custom_tiny.yaml en la
    repo
# %cd /content/yolov7

# Ajusta las rutas a como los guardaste en tu Drive
!cp "/content/drive/MyDrive/yolov7_backups/modelo_tiny_pi/best.pt" /content/yolov7/
!cp "/content/drive/MyDrive/yolov7_backups/modelo_tiny_pi/custom.yaml" /content/yolov7/data/

import re, pathlib
p = pathlib.Path("utils/plots.py")
s = p.read_text()
# pone un grosor fijo 6 cuando no se pasa line_thickness
s2 = re.sub(
    r'tl\s*=\s*line_thickness\s*or\s*round\(0\.002\s*\*\s*\(im\.shape\[0\]\s*\+\s*im\.shape
        \[1\]\)\s*/\s*2\)\s*\+\s*1',
    'tl = line_thickness or 6',
    s
)
p.write_text(s2)
print("Grosor por defecto: 6")

WEIGHTS="/content/yolov7/best.pt"
IMG="/content/yolov7/DOSMAS.jpeg"  #     pon aqu  tu imagen

!python detect.py \
  --weights "$WEIGHTS" \
  --source  "$IMG" \
  --img-size 640 \
  --conf-thres 0.25 --iou-thres 0.45 \
  --project runs/detect --name demo_img --exist-ok

# Ver el resultado
from IPython.display import Image, display
display(Image(filename="runs/detect/demo_img/" + IMG.split("/")[-1]))

!python /content/yolov7/detect.py \
  --weights best.pt \
  --source /content/drive/MyDrive/TESIS IA YOLOV7/DATASET/TEST/OTRO.mp4 \
  --img-size 640 \
  --conf-thres 0.25 --iou-thres 0.45 \
```

```
  --project runs/detect --name exp_tiny_vid --exist-ok

# Listar y mostrar el MP4 generado
!ls -lh runs/detect/exp_tiny_vid/

from IPython.display import HTML
from base64 import b64encode
mp4 = 'runs/detect/exp_tiny_vid/video.mp4'  # ajusta al nombre real
mp4_data = open(mp4,'rb').read()
data_url = "data:video/mp4;base64," + b64encode(mp4_data).decode()
HTML(f'<video width=800 controls><source src="{data_url}" type="video/mp4"></video>')

"""# EXPORTAR"""

# Commented out IPython magic to ensure Python compatibility.
# %cd /content/yolov7
WEIGHT = "runs/train/exp_tiny22/weights/best.pt"   #     ajusta si tu exp cambi
OUTDIR = "/content/export_pi_tiny"
!mkdir -p $OUTDIR

# Exportar a ONNX (din mico + simplificado). No metas NMS por ahora (m s estable en Pi).
!python export.py --weights {WEIGHT} --img-size 640 640 --device cpu --dynamic --simplify

# Copiar a carpeta de export
!cp {WEIGHT}                        {OUTDIR}/best.pt
!cp {WEIGHT.replace('.pt','.onnx')} {OUTDIR}/best.onnx
!cp data/custom_tiny.yaml           {OUTDIR}/custom.yaml

# (opcional) una imagen o video de prueba
!cp /content/drive/MyDrive/TESIS IA YOLOV7/DATASET/TEST/PATO3.jpg        {OUTDIR}/ejemplo.jpg
!cp /content/drive/MyDrive/TESIS IA YOLOV7/DATASET/TEST/FLYING.mp4         {OUTDIR}/video.mp4

# Empaquetar para llevar a la Pi
# %cd /content
!zip -jr modelo_tiny_pi.zip {OUTDIR}

from google.colab import drive; drive.mount('/content/drive')
!mkdir -p "/content/drive/MyDrive/yolov7_backups"
!cp /content/modelo_tiny_pi.zip "/content/drive/MyDrive/yolov7_backups/"

"""# EXPO ONNX"""

from google.colab import drive
drive.mount('/content/drive')

!mkdir -p /content/work && cd /content/work

!git clone https://github.com/WongKinYiu/yolov7
!cd yolov7 && ls

# AJUSTA estas rutas a donde guardaste tu backup
```

```
!cp "/content/drive/MyDrive/yolov7_backups/modelo_tiny_pi/best.pt" /content/work/weights_best.pt
!cp "/content/drive/MyDrive/yolov7_backups/modelo_tiny_pi/custom.yaml" /content/work/custom.yaml

!pip install -q onnx onnxsim

# Commented out IPython magic to ensure Python compatibility.
!pip install -q onnx onnxsim
# %cd /content/work/yolov7
!python export.py --weights /content/work/weights_best.pt --img-size 640 640 --device cpu --dynamic --simplify

# Commented out IPython magic to ensure Python compatibility.
# %cd /content/work/yolov7
# Parchear el sitio donde se cargan los pesos
!sed -i "s/torch.load(w, map_location=map_location)/torch.load(w, map_location=map_location, weights_only=False)/" models/experimental.py

# (si el sed no encuentra el patr n, haz el parche en Python)
import io, sys
p='models/experimental.py'
t=open(p,'r').read()
t=t.replace("torch.load(w, map_location=map_location)",
            "torch.load(w, map_location=map_location, weights_only=False)")
open(p,'w').write(t)
print("Patcheado:", p)

# Commented out IPython magic to ensure Python compatibility.
!mkdir -p /content/export_pi_tiny
!cp /content/work/weights_best.pt /content/export_pi_tiny/best.pt
!cp /content/work/custom.yaml   /content/export_pi_tiny/custom.yaml

# ONNX puede quedar en runs/train/*/weights/ o junto al .pt; cubrimos ambas rutas:
!cp -n /content/work/yolov7/runs/train/*/weights/*.onnx /content/export_pi_tiny/ 2>/dev/null || true
!cp -n /content/work/*.onnx /content/export_pi_tiny/ 2>/dev/null || true
!ls -lh /content/export_pi_tiny

# %cd /content
!zip -jr modelo_tiny_pi.zip /content/export_pi_tiny
```

Listing 1. Entrenamiento YOLOv7-tiny en Colab

ANEXO B
SCRIPT DE INFERENCIA ONNX (YOLOV7-TINY) PARA IMAGEN, VIDEO Y CÁMARA

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import os, time, argparse, pathlib
from datetime import datetime
```

```
import cv2, yaml, numpy as np

try:
    import onnxruntime as ort
except ImportError:
    print("ERROR: onnxruntime no est  instalado. Inst lalo con:\n  pip install onnxruntime")
    raise

# ---------- Utilidades ----------
def load_names(path_or_list):
    if isinstance(path_or_list, list):
        return path_or_list
    p = str(path_or_list)
    if p.endswith((".yaml", ".yml")):
        with open(p, "r") as f:
            y = yaml.safe_load(f)
        if isinstance(y.get("names"), list):
            return y["names"]
        elif isinstance(y.get("names"), dict):
            return [y["names"][k] for k in sorted(y["names"].keys(), key=lambda x: int(x))]
        raise ValueError("El YAML no contiene 'names'.")
    elif p.endswith(".txt"):
        with open(p, "r") as f:
            return [line.strip() for line in f if line.strip()]
    raise ValueError("Formato de nombres no soportado (usa .yaml o .txt).")

def letterbox(img, new_size=640, color=(114,114,114), auto=False, scaleup=True, stride=32):
    h0, w0 = img.shape[:2]
    if isinstance(new_size, int):
        new_size = (new_size, new_size)
    new_w, new_h = new_size
    r = min(new_w / w0, new_h / h0)
    if not scaleup:
        r = min(r, 1.0)
    new_unpad = (int(round(w0 * r)), int(round(h0 * r)))
    dw, dh = new_w - new_unpad[0], new_h - new_unpad[1]
    if auto:
        dw %= stride
        dh %= stride
    dw /= 2; dh /= 2
    if (w0, h0) != new_unpad:
        img = cv2.resize(img, new_unpad, interpolation=cv2.INTER_LINEAR)
    top, bottom = int(round(dh-0.1)), int(round(dh+0.1))
    left, right = int(round(dw-0.1)), int(round(dw+0.1))
    img = cv2.copyMakeBorder(img, top, bottom, left, right, cv2.BORDER_CONSTANT, value=color)
    return img, r, (dw, dh)

def xywh2xyxy(x):
    y = np.zeros_like(x)
    y[:,0] = x[:,0] - x[:,2] / 2
    y[:,1] = x[:,1] - x[:,3] / 2
    y[:,2] = x[:,0] + x[:,2] / 2
```

```
    y[:,3] = x[:,1] + x[:,3] / 2
    return y

def clip_coords(boxes, shape_hw):
    h, w = shape_hw
    boxes[:, 0] = np.clip(boxes[:, 0], 0, w - 1)
    boxes[:, 1] = np.clip(boxes[:, 1], 0, h - 1)
    boxes[:, 2] = np.clip(boxes[:, 2], 0, w - 1)
    boxes[:, 3] = np.clip(boxes[:, 3], 0, h - 1)

def scale_coords(img_shape_hw, coords, orig_shape_hw, ratio_pad):
    r, (dw, dh) = ratio_pad
    coords[:, [0, 2]] -= dw
    coords[:, [1, 3]] -= dh
    coords[:, :4] /= r
    clip_coords(coords, orig_shape_hw)
    return coords

def box_iou(box1, box2):
    N, M = box1.shape[0], box2.shape[0]
    if N == 0 or M == 0:
        return np.zeros((N, M), dtype=np.float32)
    lt = np.maximum(box1[:, None, :2], box2[None, :, :2])
    rb = np.minimum(box1[:, None, 2:4], box2[None, :, 2:4])
    wh = np.clip(rb - lt, a_min=0, a_max=None)
    inter = wh[:, :, 0] * wh[:, :, 1]
    area1 = (box1[:, 2] - box1[:, 0]) * (box1[:, 3] - box1[:, 1])
    area2 = (box2[:, 2] - box2[:, 0]) * (box2[:, 3] - box2[:, 1])
    union = area1[:, None] + area2[None, :] - inter
    return inter / (union + 1e-9)

def nms_np(boxes, scores, iou_thres=0.45):
    if len(boxes) == 0:
        return []
    idxs = scores.argsort()[::-1]
    keep = []
    while idxs.size > 0:
        i = idxs[0]; keep.append(i)
        if idxs.size == 1: break
        ious = box_iou(boxes[i:i+1], boxes[idxs[1:]]).reshape(-1)
        idxs = idxs[1:][ious < iou_thres]
    return keep

def have_gui():
    try:
        cv2.namedWindow("__test__"); cv2.destroyWindow("__test__")
        return True
    except cv2.error:
        return False

# ---------- Decodificaci n YOLO (para salidas crudas) ----------
# Anclas por defecto de yolov7-tiny (tres escalas, strides ~8/16/32):
```

```python
ANCHORS_DEFAULT = [
    np.array([[10,13],[16,30],[33,23]], dtype=np.float32),   # grid grande (80x80 si 640)
    np.array([[30,61],[62,45],[59,119]], dtype=np.float32),  # 40x40
    np.array([[116,90],[156,198],[373,326]], dtype=np.float32) # 20x20
]

def sigmoid(x): return 1. / (1. + np.exp(-x))

def decode_head(raw, anchors, stride):
    # raw: (1, na, gh, gw, no) con x,y,w,h,obj,cls...
    y = raw.copy()
    gh, gw = y.shape[2], y.shape[3]

    # Sigmoides
    y[..., 0:2] = sigmoid(y[..., 0:2])
    y[..., 2:4] = sigmoid(y[..., 2:4])
    y[..., 4:]  = sigmoid(y[..., 4:])

    # grid
    gy, gx = np.meshgrid(np.arange(gh), np.arange(gw), indexing='ij')  # (gh,gw)
    grid = np.stack((gx, gy), axis=-1)[None, None, ...]  # (1,1,gh,gw,2)

    # anchors
    anc = anchors.reshape((1, anchors.shape[0], 1, 1, 2))  # (1,na,1,1,2)

    # xy en p xeles
    y[..., 0] = (y[..., 0] * 2 - 0.5 + grid[..., 0]) * stride
    y[..., 1] = (y[..., 1] * 2 - 0.5 + grid[..., 1]) * stride
    # wh en p xeles
    y[..., 2] = (y[..., 2] * 2) ** 2 * anc[..., 0]
    y[..., 3] = (y[..., 3] * 2) ** 2 * anc[..., 1]
    return y  # (1,na,gh,gw,no) con xywh en p xeles y sigmoids aplicados

# ---------- Inferencia ----------
def infer_single(sess, img_bgr, img_size, conf_thres, iou_thres, names):
    h0, w0 = img_bgr.shape[:2]
    img_in, r, dwdh = letterbox(img_bgr, new_size=img_size, stride=32)
    img_rgb = cv2.cvtColor(img_in, cv2.COLOR_BGR2RGB)
    img = img_rgb.transpose(2, 0, 1).astype(np.float32) / 255.0
    img = np.expand_dims(img, 0)

    input_name = sess.get_inputs()[0].name
    outputs = sess.run(None, {input_name: img})

    # Recolecta cabezas y decide si hay que decodificar
    heads = []       # lista de (gh, gw, na, no, array_decodificado_1x(gh*gw*na)xno)
    flat_feats = []  # si ya viene decodificado (1,num,no)
    twoD_done = False

    for o in outputs:
        a = o
        if a.ndim == 2:
```

```
            # (num,6) -> ya viene con NMS/decodificado
            twoD_done = True
            det = a
            if det.shape[1] < 6:
                continue
            boxes = det[:, :4].copy()
            scores = det[:, 4].copy()
            cls = det[:, 5].astype(int).copy()
            boxes = scale_coords((img_in.shape[0], img_in.shape[1]), boxes, (h0, w0), (r, dwdh)
                )
            return [(boxes[i], float(scores[i]), int(cls[i])) for i in range(len(scores))], (r,
                 dwdh)

        if a.ndim == 5:  # (1, na, gh, gw, no)
            _, na, gh, gw, no = a.shape
            # Heur stica: si valores de xywh son muy peque os, es crudo -> decodificar
            mx = float(np.max(np.abs(a[..., :4]))) if a.size else 0.0
            stride = img_size / gh  # asumiendo entrada cuadrada
            if mx < 4.0:  # crudo
                # asigna anclas por escala seg n gh (orden: gh grande-> anchors peque os)
                # Ordena las anclas por gh descendente: 80->40->20
                # Map: gh m s grande -> ANCHORS_DEFAULT[0], etc.
                # Para tres cabezas t picas funciona bien:
                # (si tu export fuera 2 o 1 cabezas, se adapta por  ndice )
                anchors_list = ANCHORS_DEFAULT
                # El  ndice  de ancla seg n gh: cuanto mayor gh, menor stride ->  ndice  0
                # Usamos un mapeo simple por t ama o:
                heads.append(("decode", gh, gw, na, no, decode_head(a, anchors_list[0 if gh >=
                    60 else (1 if gh >= 30 else 2)], stride)))
            else:
                # parece ya decodificado en grilla (xy ~ p xeles) -> solo a (1, -1, no)
                flat_feats.append(a.reshape(1, -1, a.shape[-1]))
        elif a.ndim == 4:  # (1, gh, gw, na*no) -> probable crudo
            n, gh, gw, c = a.shape
            na = 3
            no = c // na
            a5 = a.reshape(n, na, gh, gw, no)
            stride = img_size / gh
            heads.append(("decode", gh, gw, na, no, decode_head(a5, ANCHORS_DEFAULT[0 if gh >=
                60 else (1 if gh >= 30 else 2)], stride)))
        elif a.ndim == 3:  # (1, num, no) -> ya listo
            flat_feats.append(a)
        else:
            raise RuntimeError(f"Salida ONNX no soportada: {a.shape}")

    # Si hubo cabezas crudas, comb nalas (ya en p xeles)
    if heads:
        decoded = [h[5].reshape(1, -1, h[4]) for h in heads]  # (1, num, no)
        pred = np.concatenate(decoded, axis=1)[0]             # (N, no)
    elif flat_feats:
        pred = np.concatenate(flat_feats, axis=1)[0]
    else:
```

```
        return [], (r, dwdh)

    if pred.shape[1] < 6:
        return [], (r, dwdh)

    xywh = pred[:, 0:4].copy()
    obj  = pred[:, 4:5]
    cls_logits = pred[:, 5:]
    if cls_logits.size == 0:
        return [], (r, dwdh)

    # Si todav a parecen normalizados 0..1, escala a p xeles de la imagen de entrada
    H_in, W_in = img_in.shape[0], img_in.shape[1]
    mx_xywh = float(np.max(xywh[:, :4])) if xywh.size else 0.0
    if mx_xywh <= 1.5:
        xywh[:, [0, 2]] *= W_in
        xywh[:, [1, 3]] *= H_in

    scores_per_class = obj * cls_logits
    cls_ids = scores_per_class.argmax(axis=1)
    scores  = scores_per_class.max(axis=1)

    mask = scores > conf_thres
    if not np.any(mask):
        return [], (r, dwdh)

    xywh = xywh[mask]; scores = scores[mask]; cls_ids = cls_ids[mask]
    boxes_xyxy = xywh2xyxy(xywh.copy())
    boxes_xyxy = scale_coords((img_in.shape[0], img_in.shape[1]), boxes_xyxy, (h0, w0), (r,
        dwdh))

    final_det = []
    for c in np.unique(cls_ids):
        idxs = np.where(cls_ids == c)[0]
        b = boxes_xyxy[idxs]; s = scores[idxs]
        keep = nms_np(b, s, iou_thres=iou_thres)
        for i in keep:
            final_det.append((b[i], float(s[i]), int(c)))
    return final_det, (r, dwdh)

def draw_detections(img, detections, names, thickness=2):
    for (box, score, cid) in detections:
        x1, y1, x2, y2 = box.astype(int)

        # caja
        cv2.rectangle(img, (x1, y1), (x2, y2), (0,255,0), thickness)

        # etiqueta
        label = f"{names[cid] if 0 <= cid < len(names) else cid}:{score:.2f}"
        font_scale = 0.6
        text_th = max(1, thickness - 1)
        (tw, th), _ = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, font_scale, text_th)
```

```
261
262         # fondo negro detr s del texto
263         y_text = max(0, y1 - th - 6)
264         cv2.rectangle(img, (x1, y_text), (x1 + tw + 6, y_text + th + 6), (0, 0, 0), -1)
265
266         # contorno negro + texto blanco
267         cv2.putText(img, label, (x1 + 3, y_text + th + 2),
268                     cv2.FONT_HERSHEY_SIMPLEX, font_scale, (0, 0, 0), text_th + 2, cv2.LINE_AA)
269         cv2.putText(img, label, (x1 + 3, y_text + th + 2),
270                     cv2.FONT_HERSHEY_SIMPLEX, font_scale, (255, 255, 255), text_th, cv2.LINE_AA
                        )
271     return img
272
273
274 def is_video_file(p):
275     return pathlib.Path(p).suffix.lower() in {".mp4", ".avi", ".mov", ".mkv", ".webm", ".m4v"}
276
277 def is_image_file(p):
278     return pathlib.Path(p).suffix.lower() in {".jpg", ".jpeg", ".png", ".bmp", ".tif", ".tiff",
            ".webp"}
279
280 def ensure_dir(p): os.makedirs(p, exist_ok=True)
281
282 # ---------- Main ----------
283 def main():
284     ap = argparse.ArgumentParser()
285     ap.add_argument("--weights", type=str, required=True, help="best.onnx")
286     ap.add_argument("--names",   type=str, required=True, help="custom.yaml o .txt")
287     ap.add_argument("--img-size", type=int, default=640)
288     ap.add_argument("--conf", type=float, default=0.25)
289     ap.add_argument("--iou",  type=float, default=0.45)
290     ap.add_argument("--thickness", type=int, default=3)
291     ap.add_argument("--source", type=str, default="", help="imagen, video,  ndice  de c mara,
            o pipeline GStreamer")
292     ap.add_argument("--save", action="store_true", help="guardar salida")
293     ap.add_argument("--view", action="store_true", help="mostrar ventana (si hay GUI)")
294     ap.add_argument("--save_dir", type=str, default="runs/detect", help="carpeta base de
            guardado")
295     args = ap.parse_args()
296
297     names = load_names(args.names)
298     print(f"Clases ({len(names)}): {names}")
299
300     so = ort.SessionOptions()
301     so.intra_op_num_threads = max(1, os.cpu_count() // 2)
302     sess = ort.InferenceSession(args.weights, sess_options=so, providers=["CPUExecutionProvider
            "])
303
304     ts = datetime.now().strftime("%Y%m%d_%H%M%S")
305     exp_dir = os.path.join(args.save_dir, f"exp_{ts}")
306     if args.save: ensure_dir(exp_dir)
307     gui_ok = args.view and have_gui()
```

```
308
309     # Imagen
310     if is_image_file(args.source) and os.path.isfile(args.source):
311         img0 = cv2.imread(args.source)
312         if img0 is None:
313             print(f"No pude leer {args.source}"); return
314         t0 = time.time()
315         dets, _ = infer_single(sess, img0, args.img_size, args.conf, args.iou, names)
316         fps = 1.0 / max(1e-9, (time.time() - t0))
317         out = draw_detections(img0.copy(), dets, names, thickness=args.thickness)
318         print(f"Detecciones: {len(dets)} | FPS aprox: {fps:.2f}")
319         if args.save:
320             out_path = os.path.join(exp_dir, pathlib.Path(args.source).stem + "_det.jpg")
321             cv2.imwrite(out_path, out); print(f"Guardado: {out_path}")
322         if gui_ok:
323             try: cv2.imshow("det", out); cv2.waitKey(0)
324             except cv2.error: pass
325         try: cv2.destroyAllWindows()
326         except cv2.error: pass
327         return
328
329     # Video / GStreamer / C mara
330     cap = None
331     is_pipeline = ("!" in args.source) and not os.path.isfile(args.source)
332     if is_pipeline:
333         cap = cv2.VideoCapture(args.source, cv2.CAP_GSTREAMER)
334     else:
335         if is_video_file(args.source) and os.path.isfile(args.source):
336             cap = cv2.VideoCapture(args.source)
337         else:
338             try:
339                 idx = int(args.source); cap = cv2.VideoCapture(idx)
340             except:
341                 if args.source == "" and os.path.exists("/dev/video0"):
342                     cap = cv2.VideoCapture(0)
343                 else:
344                     print("Fuente no reconocida. Usa imagen, video,  ndice  de c mara , o
                          pipeline GStreamer.")
345                     return
346
347     if not cap or not cap.isOpened():
348         print("No pude abrir la fuente de video.")
349         return
350
351     writer, out_path = None, None
352     try:
353         while True:
354             ok, frame = cap.read()
355             if not ok: break
356             t0 = time.time()
357             dets, _ = infer_single(sess, frame, args.img_size, args.conf, args.iou, names)
358             fps = 1.0 / max(1e-9, time.time() - t0)
```

```
            vis = draw_detections(frame.copy(), dets, names, thickness=args.thickness)
            if args.save and writer is None:
                ensure_dir(exp_dir)
                h, w = vis.shape[:2]
                out_path = os.path.join(exp_dir, "out.mp4")
                fourcc = cv2.VideoWriter_fourcc(*"mp4v")
                writer = cv2.VideoWriter(out_path, fourcc, 25.0, (w, h))
                if not writer.isOpened():
                    out_path = os.path.join(exp_dir, "out.avi")
                    fourcc = cv2.VideoWriter_fourcc(*"XVID")
                    writer = cv2.VideoWriter(out_path, fourcc, 25.0, (w, h))
            if writer is not None: writer.write(vis)
            if gui_ok:
                try:
                    txt = f"FPS {fps:.1f}"
                    cv2.putText(vis, txt, (10, 25), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,0,0), 2,
                        cv2.LINE_AA)
                    cv2.putText(vis, txt, (10, 25), cv2.FONT_HERSHEY_SIMPLEX, 0.8,
                        (255,255,255), 1, cv2.LINE_AA)
                    cv2.imshow("det", vis)
                    if cv2.waitKey(1) & 0xFF == 27: break  # ESC
                except cv2.error:
                    gui_ok = False
    finally:
        cap.release()
        if writer is not None: writer.release()
        try: cv2.destroyAllWindows()
        except cv2.error: pass
    if out_path: print(f"Video guardado en: {out_path}")

if __name__ == "__main__":
    main()
```

Listing 2. Entrenamiento YOLOv7-tiny en Colab