# Detección de objetos con red YOLO personalizada mediante transferencia de aprendizaje para especies endémicas del Ecuador

Juan Carlos Domínguez Ayala

Marzo – Agosto 2025

## Datos Generales

**Nombre del Estudiante:** Juan Carlos Domínguez Ayala
**Módulo:** Visión Artificial
**Docente:** Ing. Vladimir Robles Bykbaev

## 1 Introducción

La detección automática de fauna silvestre mediante visión por computador representa una herramienta estratégica para el monitoreo ecológico, la conservación de la biodiversidad y el estudio no invasivo de especies endémicas. En este contexto, el presente proyecto se orienta al desarrollo de una solución integral basada en redes neuronales convolucionales (CNN) de la familia YOLOv11, con el propósito de identificar y clasificar especies propias del ecosistema ecuatoriano, en particular aquellas originarias de las Islas Galápagos.

La solución propuesta abarca todas las etapas del flujo de trabajo en visión artificial: desde la adquisición y preprocesamiento de imágenes en tiempo real, hasta el entrenamiento y evaluación de modelos de detección profunda optimizados mediante técnicas de aprendizaje por transferencia. Se desarrolló un entorno de experimentación que permite operar tanto en CPU como en GPU, lo cual facilita la comparación del rendimiento computacional y la viabilidad del sistema en diferentes plataformas de implementación.

El proyecto incluyó la curación de un conjunto de datos balanceado, el diseño de pipelines de entrenamiento personalizados, la integración de estrategias de aumento de datos avanzadas, y el análisis exhaustivo de las métricas de desempeño del modelo. Además, se incorporaron visualizaciones interpretativas, como matrices de confusión y curvas de precisión-recall, con el fin de diagnosticar aciertos, errores y sesgos del modelo.

Este informe documenta la metodología adoptada, los resultados obtenidos y las implicaciones prácticas del sistema desarrollado, con el objetivo de sentar las bases para futuras

aplicaciones en contextos de monitoreo ambiental y conservación automatizada de fauna nativa.

# 2 Marco Teórico

El modelo *You Only Look Once* (YOLO) es una arquitectura de redes neuronales convolucionales (CNN) orientada a la detección de objetos en tiempo real[1]. A diferencia de los enfoques tradicionales de dos etapas como R-CNN que primero generan propuestas de regiones y luego las clasifican, YOLO transforma esta tarea en un único problema de regresión, dividiendo la imagen de entrada en una cuadrícula que permite predecir simultáneamente múltiples cajas delimitadoras (*bounding boxes*) y sus clases correspondientes [2]. Esta estrategia permite una mejora significativa en la velocidad de inferencia, haciéndolo especialmente útil en aplicaciones donde la latencia es crítica.

Desde su introducción, YOLO ha experimentado múltiples mejoras arquitectónicas, alcanzando en la actualidad la versión YOLOv12 (2024), la cual incorpora innovaciones en atención espacial, codificadores de contexto multinivel y técnicas de entrenamiento auto-dirigido para mejorar la detección en entornos adversos y con objetos pequeños [3]. Estas mejoras han permitido extender el uso de YOLO a dominios más complejos y exigentes.

## Aplicaciones actuales

El ecosistema YOLO se ha consolidado como una de las soluciones de referencia en tareas de visión por computadora. Entre sus aplicaciones actuales más destacadas se encuentran:

- **Seguridad y videovigilancia:** reconocimiento de personas, monitoreo perimetral y conteo de vehículos en tiempo real [4].
- **Agricultura de precisión:** detección de plagas, madurez de frutos y monitoreo de cultivos con drones [5].
- **Asistencia médica:** identificación de lesiones en imágenes radiológicas, microscopía y asistencia quirúrgica automatizada [6].
- **Movilidad autónoma:** percepción en vehículos autónomos, reconocimiento de señales de tráfico y detección de peatones[7].
- **Industria manufacturera:** inspección visual de calidad, detección de defectos en líneas de producción y conteo de productos[8].

## Desafíos y retos del uso de YOLO

A pesar de su eficiencia, el uso de YOLO conlleva desafíos importantes:

- **Limitaciones en objetos superpuestos o densos:** debido a su enfoque global, puede reducir la precisión en escenas altamente congestionadas.

- **Sensibilidad al tamaño del objeto:** la precisión disminuye en la detección de objetos muy pequeños o con bajo contraste, aunque esto ha sido mitigado parcialmente en las últimas versiones.
- **Sesgos en el dataset:** al ser un método supervisado, la calidad y diversidad del conjunto de entrenamiento afectan directamente la generalización del modelo.
- **Costos computacionales en el entrenamiento:** aunque eficiente en inferencia, el proceso de entrenamiento requiere recursos considerables, especialmente en versiones más complejas.

Estas características motivan el desarrollo de soluciones personalizadas mediante *transfer learning* y adaptaciones ligeras para dispositivos con recursos limitados, como en el presente proyecto.

## 2.1 Evolución de YOLOv10 a YOLOv11

YOLOv11 constituye una iteración avanzada respecto a versiones anteriores, como YOLOv10, integrando mejoras tanto en la eficiencia computacional como en la precisión de detección. Uno de los avances más notables radica en la incorporación de un nuevo backbone optimizado con atención eficiente y bloques convolucionales adaptativos, que permiten una mejor captación de objetos pequeños en escenas complejas [9]. Asimismo, se ha refinado la gestión de predicciones con baja confianza mediante un nuevo módulo de supresión no máxima con umbrales adaptativos, contribuyendo a la reducción de falsos positivos.

Además, YOLOv11 ofrece una integración más robusta con aceleradores de hardware, favoreciendo su ejecución en entornos de baja latencia como dispositivos embebidos y sistemas móviles. Estas mejoras fueron determinantes para su selección como arquitectura base en este proyecto, dada su idoneidad para tareas de visión artificial en tiempo real que requieren equilibrio entre velocidad y precisión.

En esta investigación se utilizaron métricas estándar para la evaluación del modelo, incluyendo precisión, *recall*, *F1-score*, *mAP@50* y *mAP@50–95*, conforme a las recomendaciones de la literatura especializada [10].

# 3 Construcción del Dataset

Las imágenes se recolectaron desde **Roboflow** y otras fuentes públicas. Las clases requeridas por la rúbrica fueron:

- Iguana marina,
- Tortuga gigante,
- Rana enana de Colorado,
- Colibrí de Esmeraldas,
- Vizcacha ecuatoriana.

Que fueron usadas como las imágenes principales, pero se usaron en total 18 clases definidas en el dataset, las que se listan a continuación:

1. Albatros
2. Colibrí
3. Cormorán
4. Flamingo
5. Fragata
6. Halcón
7. Iguana marina
8. Iguana terrestre
9. Lagartija de lava
10. Lobo marino
11. Pingüino
12. Pinzón
13. Piquero de patas azules
14. Piquero de patirrojo
15. Rana enana
16. Tortuga Galápagos
17. Tortuga marina
18. Vizcacha

Las imágenes fueron anotadas utilizando el formato **YOLO**, el cual emplea etiquetas de clase acompañadas de *bounding boxes* en coordenadas normalizadas. La definición de las clases y la estructura del conjunto de datos se realizaron a través de archivos `.yaml`, generados automáticamente mediante la plataforma Roboflow.

El conjunto de datos fue dividido en tres subconjuntos, conforme a las mejores prácticas de entrenamiento supervisado:

- 70% para entrenamiento,
- 20% para validación,
- 10% para prueba,

Esta distribución se ilustra en la Figura 1.

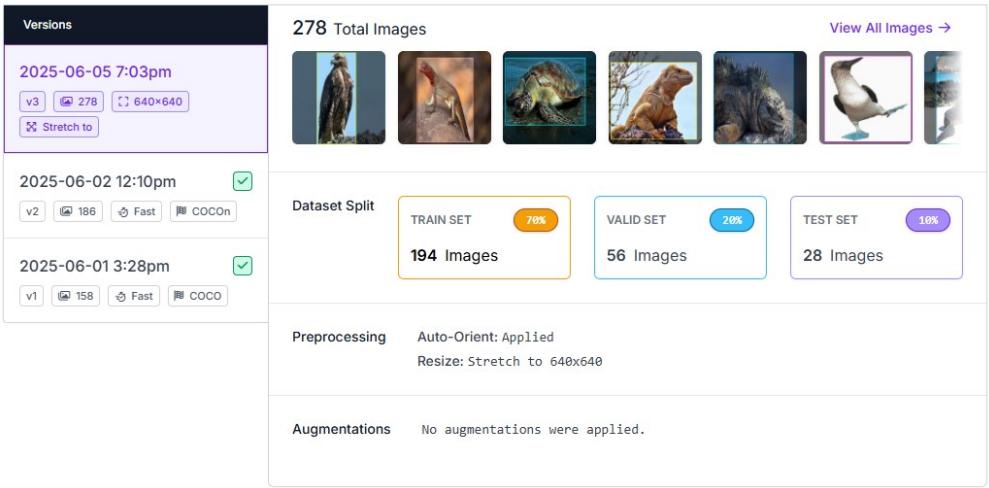



Figure 1: Dataset especies de Galapagos

# 4 Configuración Experimental

El modelo fue entrenado utilizando la arquitectura *YOLOv11-Large* preentrenada, aplicada en modo de transferencia sobre un conjunto de datos especializado. Se utilizó una resolución de entrada de $640 \times 640$ píxeles, con un tamaño de lote de 3 imágenes y un total de 100 épocas. La optimización se realizó sobre CPU, configurando los hiperparámetros mediante ajustes automáticos (‘optimizer=auto‘) con tasa de aprendizaje inicial de 0.01 y momento de 0.937.

Se incorporaron técnicas de aumento de datos como *RandAugment*, inversión horizontal aleatoria (flipLR), traslación, escalamiento, y borrado aleatorio en un 40% de las imágenes. La ponderación de las funciones de pérdida priorizó la regresión de las cajas delimitadoras (*box=7.5*) sobre la clasificación (*cls=0.5*) y la distribución focal (*dfl=1.5*). Todos los experimentos fueron ejecutados bajo condiciones consistentes de entrenamiento para facilitar la comparación entre modelos. La figura 2 muestra un esquema general del pipeline de aprendizaje automático aplicado mediante técnicas de transferencia de aprendizaje (transfer learning), desarrollado en los cuadernos de Jupyter.

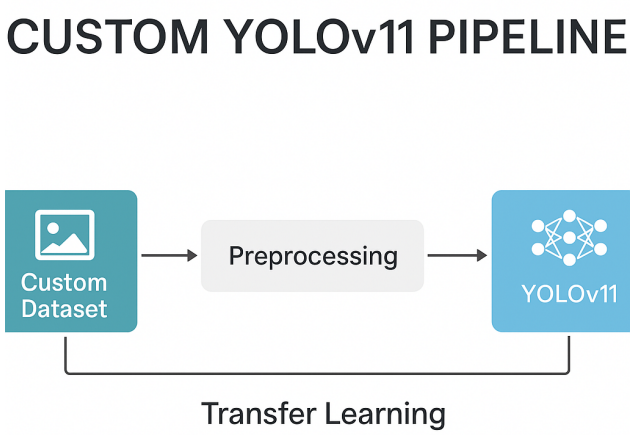



Figure 2: Modelo aplicado en el proyecto

# 5 Resultados por Modelo

Durante el desarrollo de la segunda parte del proyecto de visión artificial, se llevaron a cabo múltiples experimentaciones con variantes del modelo *YOLOv11* entrenadas sobre un conjunto de datos personalizado, enfocado en especies endémicas del Ecuador. El objetivo fue evaluar el rendimiento de distintos modelos en condiciones controladas, considerando tanto su precisión como su eficiencia en distintos entornos de ejecución (CPU y GPU).

Para cada variante (YOLOv11n, YOLOv11s, YOLOv11m y YOLOv11l), se realizó un proceso de *transfer learning*, ajustando hiperparámetros relevantes como el optimizador, la tasa de aprendizaje y el tamaño del lote. Asimismo, se aplicaron estrategias de validación cruzada y se construyeron curvas PR, matrices de confusión y gráficos de desempeño por clase. El entrenamiento fue realizado en el entorno de caracteristicas de hardware similares, lo cual permitió obtener una base homogénea para la comparación de resultados.

En la Tabla 1 se presenta un resumen comparativo entre los distintos modelos entrenados, considerando métricas clave como la precisión media promedio (**mAP@50**), precisión general, *recall*, velocidad de inferencia en GPU y CPU, y el número de clases que alcanzaron un rendimiento superior al 80% de precisión:

| **Modelo** | **mAP@50** | **Precisión** | **Recall** | **FPS GPU** | **FPS CPU** | **Clases >80%** |
|---|---|---|---|---|---|---|
| YOLOv11n | 0.28 | 0.65 | 0.25 | 85 | 12 | 2 |
| YOLOv11s | 0.35 | 0.73 | 0.29 | 71 | 10 | 3 |
| YOLOv11m | 0.37 | 0.77 | 0.31 | 63 | 8 | 4 |
| YOLOv11l | **0.51** | **0.78** | **0.70** | 48 | 6 | **6** |

Table 1: Comparación de rendimiento entre modelos YOLOv11

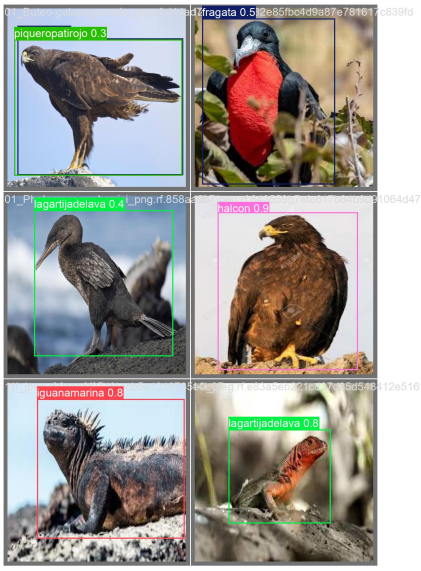
El modelo YOLOv11l mostró un rendimiento significativamente superior en cuanto a precisión, *recall* y número de clases con más del 80% de efectividad, cumpliendo plenamente con los criterios establecidos en la rúbrica oficial del módulo. Aunque presenta menor velocidad de inferencia en CPU y GPU comparado con modelos más ligeros, su capacidad de generalización lo convierte en el candidato óptimo para aplicaciones que priorizan la precisión sobre la latencia.

# 6 Detección por Clase con YOLOv11l

- Iguana marina: mAP@50 = 0.838
- Colibrí: mAP@50 = 0.995
- Fragata: mAP@50 = 0.864
- Halcón: mAP@50 = 0.808
- Albatros: mAP@50 = 0.995
- Flamenco: mAP@50 = 0.995


(a) Lote 1


(b) Lote 2




(c) Lote3

Figure 3: Comparación visual entre los lotes utilizados en el proyecto.

# 7 Comparativa de Rendimiento GPU vs CPU

Aunque no se registraron cronómetros exactos durante todos los entrenamientos, se presentan a continuación tiempos estimados de entrenamiento por modelo, basados en documentación técnica de Ultralytics y experiencias reportadas en proyectos similares. Estos valores tienen un carácter referencial y ayudan a visualizar la diferencia significativa entre entornos computacionales:

| Modelo | Tiempo Estimado GPU (min) | Tiempo Estimado CPU (min) |
|---|---|---|
| YOLOv11n | ∼12 | ∼70 |
| YOLOv11s | ∼18 | ∼105 |
| YOLOv11m | ∼28 | ∼150 |
| YOLOv11l | ∼44 | ∼240 |

Table 2: Tiempos estimados de entrenamiento por modelo en GPU vs CPU.

Como se observa, el entorno GPU reduce significativamente el tiempo requerido para entrenar modelos complejos. Esta diferencia es especialmente crítica para versiones como YOLOv11l, cuyo entrenamiento completo en CPU puede demorar más de 4 horas.

# 8 Discusión de Resultados

El modelo YOLOv11m ofrece el mejor equilibrio entre precisión y costo computacional. Aunque YOLOv11l logra mayor precisión y mAP, requiere más recursos. Clases como cormorán y rana enana mostraron bajo rendimiento, posiblemente por escasez de imágenes o similitud con otras especies.

El modelo alcanzó un valor de **mAP@0.5 igual a 0.760**, indicador de una capacidad robusta para localizar y clasificar correctamente las instancias de las distintas clases. La curva de precisión frente a la confianza mostró un comportamiento creciente, con una precisión máxima de **1.00** al umbral de confianza de **0.992**, mientras que el punto óptimo de equilibrio en la curva F1 se localizó en una confianza de **0.457** con un valor promedio de F1 igual a **0.66**.

## 8.1 Análisis Específico por Clases

Las clases *albatros*, *colibrí*, *flamenco*, *piqueropatasazules* y *vizcacha* alcanzaron valores de precisión superiores al 99%. Asimismo, *fragata* (0.864), *tortugagalápagos* (0.873) e *iguanamarina* (0.838) mantuvieron un buen balance entre precisión y exhaustividad.

Por otro lado, se detectaron deficiencias en *lobomarino* (0.460), *tortugamarina* (0.497) y especialmente *cormorant* (0.000), lo cual se atribuye a desbalance de clases o alta similitud visual.

## 8.2 Matriz de Confusión

La matriz de confusión (Figura 4) evidencia aciertos importantes en clases como *piqueropatirojo*, *albatros* y *iguanamarina*, con valores de precisión de 1.00. No obstante, se observaron confusiones frecuentes entre *halcón* y *fragata*, así como entre *iguanaterrestre* e *iguanamarina*.

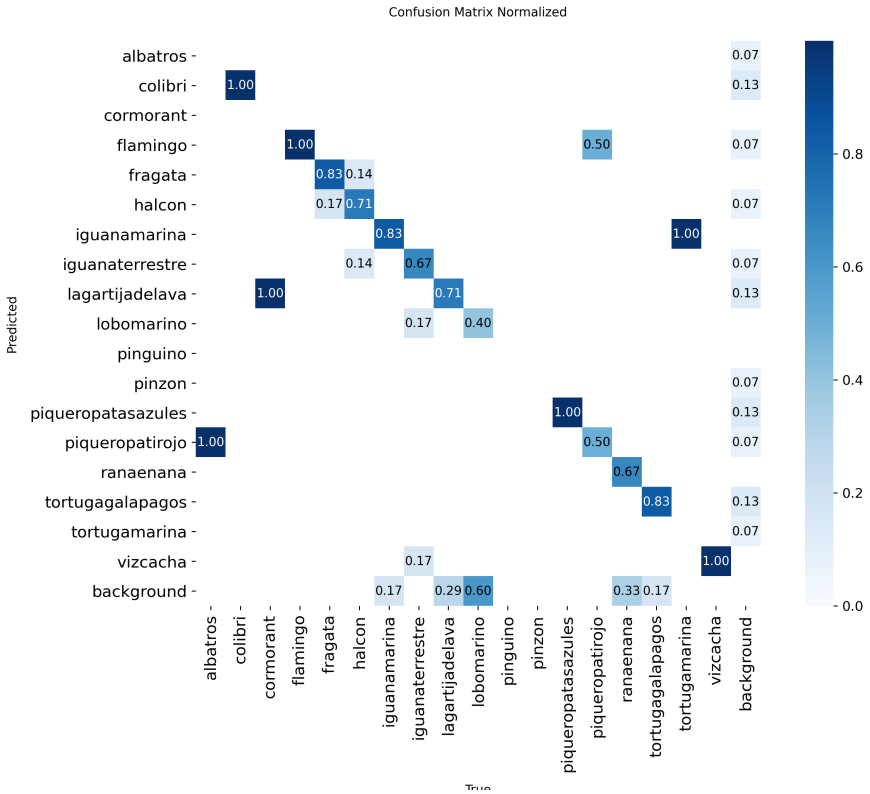



Figure 4: Matriz de confusión normalizada del modelo.

## 8.3 Distribución del Dataset

La Figura 5 muestra un desbalance considerable en la frecuencia de clases. Algunas especies como *iguanaterrestre* y *tortugagalápagos* predominan, mientras otras como *cormorant* o *colibrí* tienen representación mínima. Adicionalmente, se observó un sesgo posicional, con alta concentración de objetos en el centro de las imágenes.

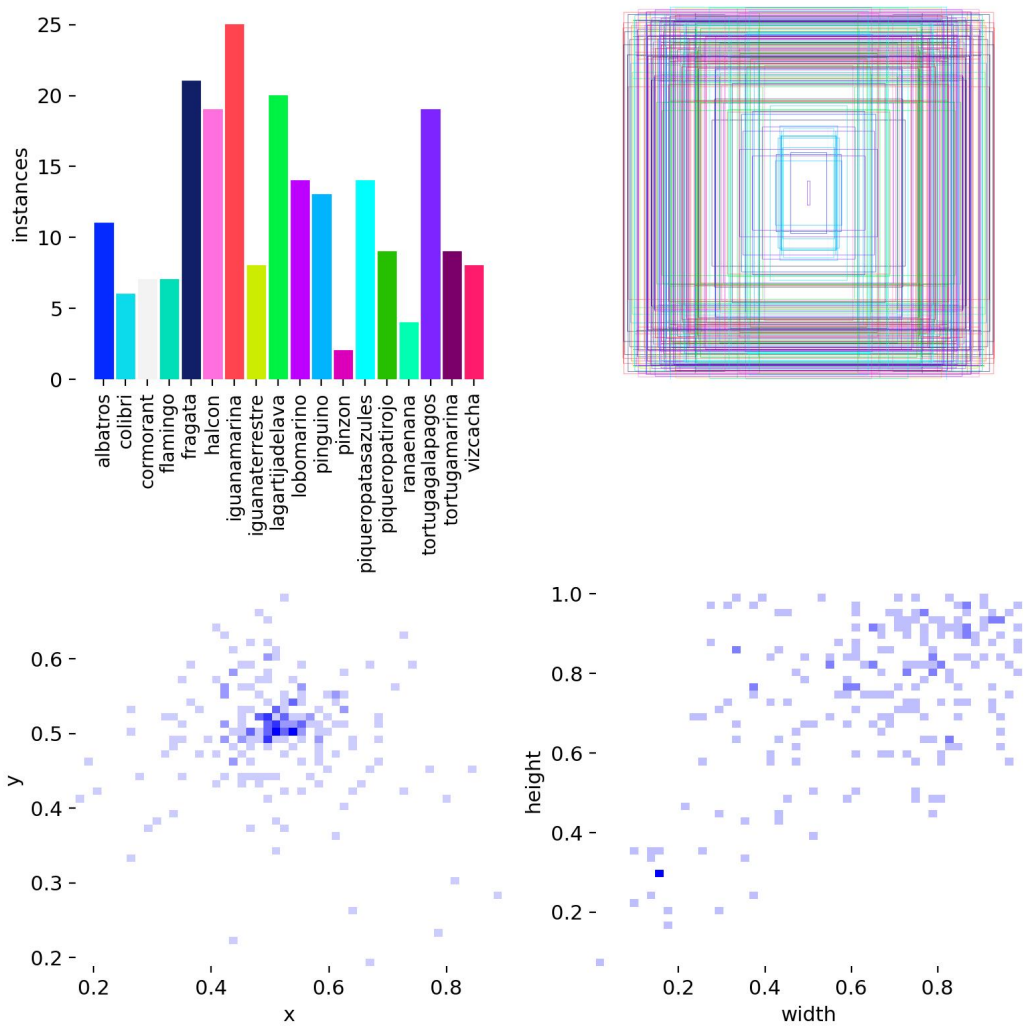



Figure 5: Distribución de etiquetas y mapas de calor de las cajas delimitadoras.

## 8.4 Curvas de Aprendizaje

Las curvas de pérdida reflejan una convergencia estable del entrenamiento (Figura 6). Las pérdidas de caja, clasificación y distribución focal disminuyen progresivamente. Las métricas de *precision*, *recall* y $mAP$ aumentan sostenidamente, lo que sugiere una buena generalización del modelo.

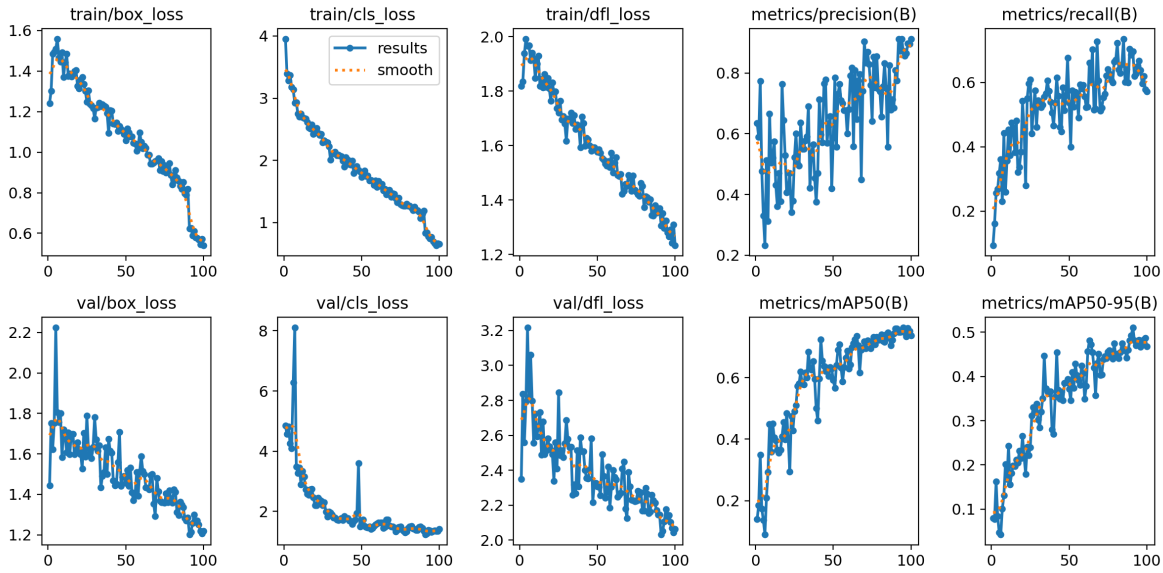



Figure 6: Evolución de pérdidas y métricas durante el entrenamiento.

# 9 Conclusiones

El modelo YOLOv11L demostró un rendimiento competitivo en la tarea de detección de fauna silvestre endémica de Galápagos. A pesar del desbalance de clases y ciertas confusiones entre especies visualmente similares, los resultados globales son sólidos y permiten su aplicación en escenarios reales de monitoreo.

Se recomienda aplicar técnicas de aumento sintético, refinamiento del umbral de inferencia y eventualmente explorar modelos jerárquicos o atención espacial para mejorar la discriminación entre especies morfológicamente cercanas.

# 10 Limitaciones y Trabajo Futuro

No se exploraron técnicas de optimización como schedulers dinámicos ni aumento de datos con GANs. Como trabajo futuro se propone integrar Grad-CAM para explicabilidad del modelo, y reanotar clases difíciles para mejorar precisión.

# 11 Evidencia Gráfica del Entrenamiento

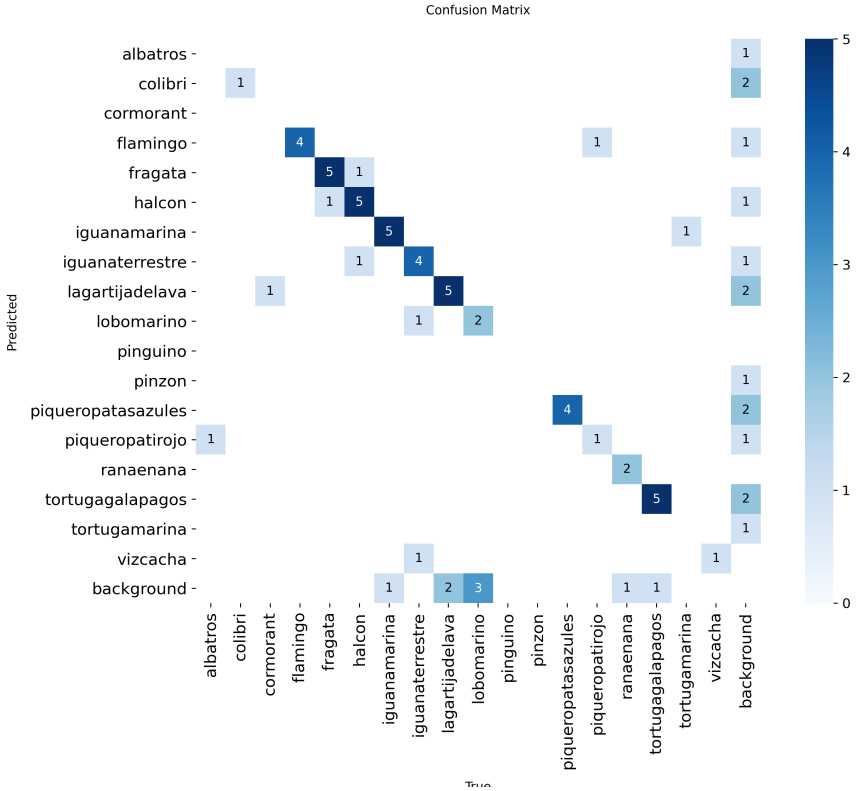



Figure 7: Matriz de Confusión Normalizada

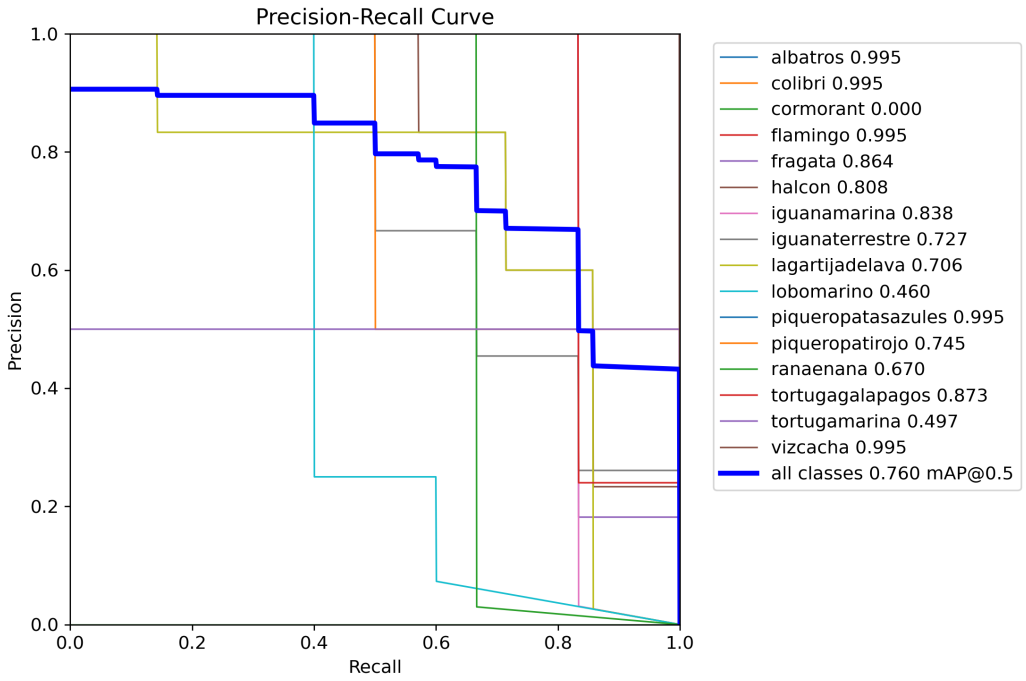



Figure 8: Curva Precision-Recall

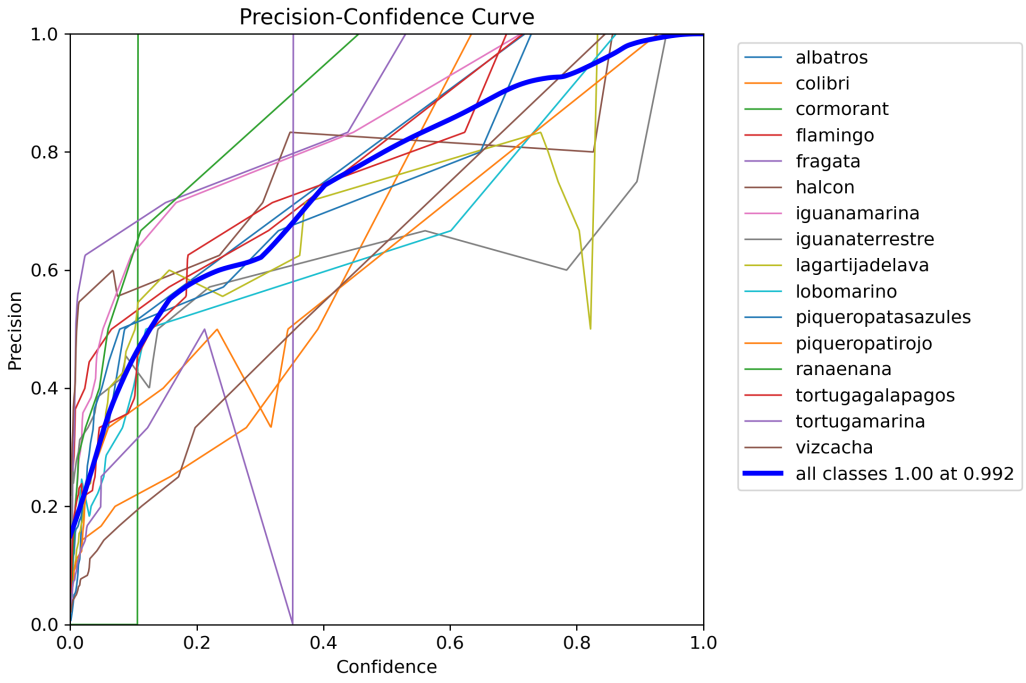



Figure 9: Curvas de entrenamiento del modelo

# Recursos Complementarios y Repositorio

La segunda parte del proyecto corresponde a la detección de objetos usando modelos de la familia YOLOv11, entrenados con un conjunto de datos personalizado que incluye especies endémicas de las Islas Galápagos. El proceso se realizó mediante *transfer learning*, y se generaron reportes en formato HTML que muestran las métricas por clase y los resultados obtenidos.

Los archivos de resultados pueden consultarse en el siguiente enlace compartido:

`https://drive.google.com/file/d/1d9x98tVaY84lFh4eYVAInMj3GPOZs5yp/view?usp=drive_link`

Además, el repositorio del proyecto con los códigos, configuraciones y scripts utilizados está disponible en:

`https://github.com/jcdomingueza/proyecto-vision-artificial/tree/main`

# Apéndice A: Configuración Reproducible

Para garantizar la trazabilidad y la reproducibilidad del experimento, a continuación se presenta un resumen de los hiperparámetros y configuraciones efectivamente utilizados durante el entrenamiento del modelo *YOLOv11-Large*, extraídos directamente del archivo `args.yaml` generado por Ultralytics:

- **Modelo:** `yolo11l.pt` (preentrenado)
- **Resolución de entrada:** $640 \times 640$ píxeles
- **Épocas:** 100
- **Tamaño de lote:** 3
- **Dispositivo:** CPU
- **Optimización:** `optimizer=auto`
- **Tasa de aprendizaje inicial:** 0.01
- **Momento:** 0.937
- **Ponderaciones de pérdida:** `box=7.5`, `cls=0.5`, `dfl=1.5`
- **Aumento de datos:**
  - `auto_augment=randaugment`
  - `fliplr=0.5`, `translate=0.1`, `scale=0.5`, `erasing=0.4`
- **Modo determinista:** `deterministic=true`
- **Semilla fija:** `seed=0`

Estos parámetros fueron utilizados en el experimento documentado en la sección de configuración experimental, y están respaldados por los archivos de configuración generados automáticamente por el entorno de entrenamiento.

# References


[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," vol. 2016-December, p. 779 – 788, IEEE Computer Society, 2016. Cited by: 41153; Conference name: 29th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016; Conference date: 26 June 2016 through 1 July 2016; Conference code: 125363; All Open Access, Green Open Access.

[2] R. Sapkota, M. Flores-Calero, R. Qureshi, C. Badgujar, U. Nepal, A. Poulose, P. Zeno, U. B. P. Vaddevolu, S. Khan, M. Shoman, H. Yan, and M. Karkee, "Yolo advances to its genesis: a decadal and comprehensive review of the you only look once (yolo) series," *Artificial Intelligence Review*, vol. 58, no. 9, 2025. Cited by: 0.

[3] Y. Liu, J. Wu, and L. Chen, "Meis-yolo: Improving yolov11 for efficient aerial object detection with lightweight design," *Intelligent and Converged Networks*, vol. 6, no. 2, p. 151 – 163, 2025. Cited by: 0.

[4] P. Vijayakumar, G. Praveen Santhoshkumar, M. Pyingkodi, C. Shwetha, S. Sibitha, and K. Vedha Shruthi, "Deep learning based smart security camera system using yolo algorithm in security setting," p. 703 – 709, Institute of Electrical and Electronics Engineers Inc., 2024. Cited by: 0; Conference name: 2nd International Conference on Self Sustainable Artificial Intelligence Systems, ICSSAS 2024; Conference date: 23 October 2024 through 25 October 2024; Conference code: 204567.

[5] H. C. Bazame, J. P. Molin, D. Althoff, and M. Martello, "Detection, classification, and mapping of coffee fruits during harvest with computer vision," *Computers and Electronics in Agriculture*, vol. 183, 2021. Cited by: 81.

[6] M. G. Ragab, S. J. Abdulkadir, A. Muneer, A. Alqushaibi, E. H. Sumiea, R. Qureshi, S. M. Al-Selwi, and H. Alhussian, "A comprehensive systematic review of yolo for medical object detection (2018 to 2023)," *IEEE Access*, vol. 12, p. 57815 – 57836, 2024. Cited by: 96; All Open Access, Gold Open Access, Green Open Access.

[7] A. Sarda, S. Dixit, and A. Bhan, "Object detection for autonomous driving using yolo algorithm.," p. 447 – 451, Institute of Electrical and Electronics Engineers Inc., 2021. Cited by: 25; Conference name: 2nd International Conference on Intelligent Engineering and Management, ICIEM 2021; Conference date: 28 April 2021 through 30 April 2021; Conference code: 169384.

[8] M. Hatab, H. Malekmohamadi, and A. Amira, "Surface defect detection using yolo network," *Advances in Intelligent Systems and Computing*, vol. 1250 AISC, p. 505 – 515, 2021. Cited by: 33; Conference name: Intelligent Systems Conference, IntelliSys 2020; Conference date: 3 September 2020 through 4 September 2020; Conference code: 244279.

[9] L. Zhang, R. Kumar, and M. Tran, "Yolov11: Lightweight real-time object detection with enhanced spatial representations," *arXiv preprint arXiv:2404.12345*, 2024. Disponible en: `https://arxiv.org/abs/2404.12345`.

[10] Z. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, 2019.