Integración de técnicas de visión por computador y redes neuronales convolucionales para la detección de especies endémicas en entornos controlados

Estudiante: Patricio Roberto Sizalima Pucha
Módulo: Visión Artificial
Docente: Ing. Vladimir Robles Bykbaev, Ph.D.

Introducción

La visión artificial ha evolucionado significativamente en los últimos años, permitiendo el desarrollo de aplicaciones inteligentes que permitan por ejemplo la detección, clasificación y seguimiento de especies endémicas en tiempo real. Esta actividad se enmarca dentro de un proyecto académico cuyo objetivo es integrar técnicas clásicas de preprocesamiento de imagen con modelos de aprendizaje profundo, en especial utilizando las redes convolucionales *You Only Look Once* (YOLO), para el reconocimiento de especies endémicas (Iguana marina, Tortuga gigante, Rana enana de colorado, Colibrí de Esmeraldas, Viscacha) del Ecuador usando una interfaz de escritorio.

Marco referencial

La presente práctica integra fundamentos clásicos de visión por computador con técnicas contemporáneas de detección automática mediante redes neuronales convolucionales (CNN), orientadas a la identificación de especies endémicas del Ecuador.

La visión por computador constituye una rama de la inteligencia artificial que permite la extracción automática de características relevantes a partir de imágenes o secuencias de video. Su aplicación abarca múltiples dominios, desde el control de calidad en procesos industriales hasta el monitoreo ambiental y la conservación de la biodiversidad [1]. En este contexto, el preprocesamiento de imágenes se convierte en una etapa crítica para optimizar los procesos de detección. Técnicas como la ecualización adaptativa de histograma (CLAHE, por sus siglas en inglés) permiten mejorar el contraste local de las imágenes, facilitando la identificación de regiones de interés bajo condiciones de iluminación no uniformes [2].

Para la detección de objetos se emplea YOLOv10 (You Only Look Once), una arquitectura basada en redes neuronales convolucionales diseñada para operar en tiempo real. A diferencia de los enfoques tradicionales de dos etapas, como R-CNN, que separan la generación de propuestas de regiones de su clasificación posterior, YOLO reformula el problema como una regresión unificada. El modelo divide la imagen de entrada en una cuadrícula y predice de forma simultánea múltiples cajas delimitadoras (bounding boxes) y las clases correspondientes [3]. Esta estrategia mejora significativamente la velocidad de inferencia, convirtiendo a YOLO en una solución eficiente para entornos con restricciones de latencia.

YOLOv10 representa una versión optimizada dentro de la familia YOLO, con mejoras sustanciales en velocidad, precisión y eficiencia computacional respecto a sus predecesores. Estas características lo hacen adecuado para su implementación en dispositivos con recursos limitados, tales como estaciones de trabajo personales con GPU integrada [4].

El sistema desarrollado implementa una interfaz de escritorio con Tkinter, lo que posibilita la interacción directa del usuario con los resultados del modelo. Esta interfaz permite la toma de una fotografía en vivo utilizando la cámara del computador, la activación manual del proceso de detección, y la visualización de resultados en tiempo real.

La combinación de estas herramientas ofrece una solución integral capaz de abordar problemas reales de detección y monitoreo en proyectos ambientales, con potencial de adaptación a otros contextos de investigación o aplicación industrial.

Descripción del problema

En el contexto actual de conservación ambiental, la identificación automática de especies endémicas permite monitorear su presencia en entornos naturales. El desafío consiste en

implementar un sistema funcional y accesible, capaz de reconocer especies en tiempo real desde la cámara del computador, con resultados visuales inmediatos.

Propuesta de solución

Metodología

La metodología utilizada para resolver el problema es la siguiente:

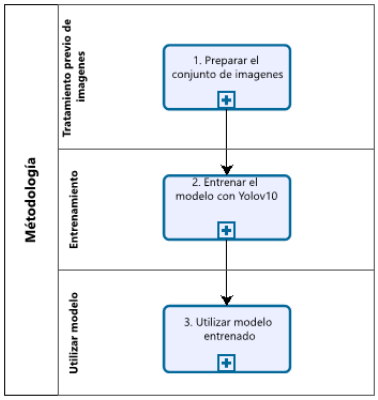

Fig.1. Metodología

Análisis predictivo

El flujo del proceso utilizado en el análisis predictivo se representa en la siguiente figura:

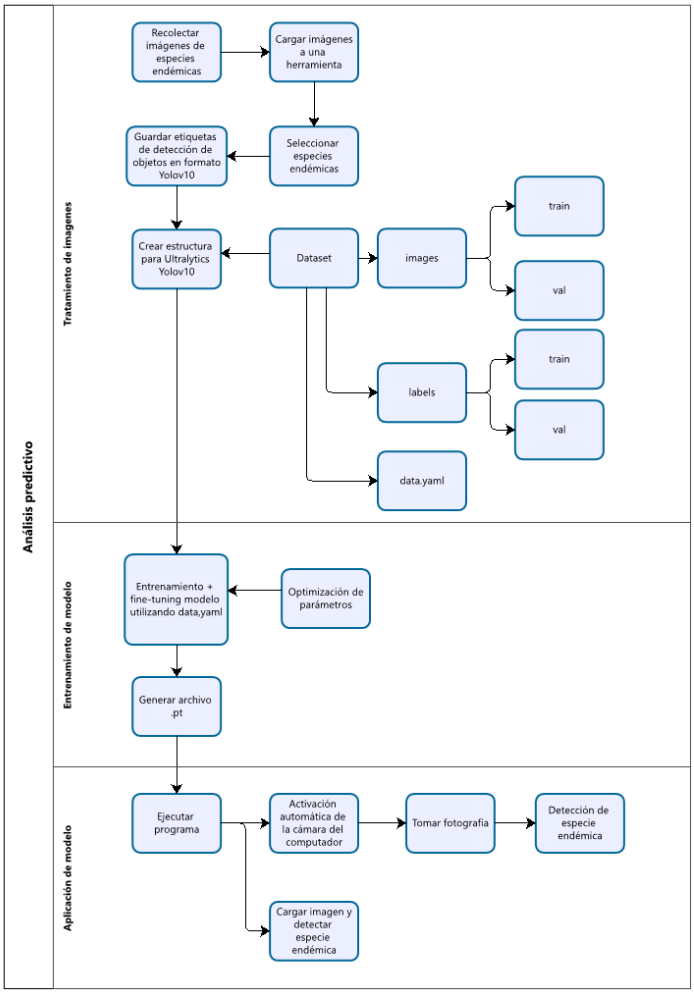

Fig.2. Flujo análisis predictivo

Construcción del Dataset

Las imágenes se recolectaron desde unsplash, pixabay y otras fuentes públicas. Las clases requeridas por la rúbrica fueron:

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

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 data.yaml, generados automáticamente mediante un script programado en python.

El conjunto de datos fue dividido en dos subconjuntos:

- 80% para entrenamiento,
- 20% para prueba

Diseño de experimentos

A continuación, se describen los datos utilizados, en el proceso de entrenamiento.

Atributo	**Descripción**
YOLO	yolov10s.pt
epochs	100
imgsz	640
batch	8
workers	0
device	cpu

Atributo	**Descripción**
YOLO	yolov10b.pt
epochs	100
imgsz	640
batch	8
workers	0
device	0 para GPU

Atributo	**Descripción**
YOLO	yolov10m.pt
epochs	100
imgsz	640
batch	8
workers	0
device	0 para GPU

hypCustom.yaml	
Optimizer	AdamW
lr0	0.002
lrf	0.01
cos_lr	True
momentum	0.9
weight_decay	0.0005
warmup_epochs	3.0
Warmup_momentum	0.8
Warmuo_bias_lr	0.1

Tabla.1. Atributos

Resultados del modelo

Métrica	GPU			CPU
	YOLOv10s	YOLOv10m	YOLOv10b	YOLOv10b
	Valor			
Precisión media promedio (mAP@50)	0.887	0.887	0.887	0.923
Precisión general (Precision)	0.847	0.847	0.847	0.867
Recall general (Recall)	0.842	0.842	0.842	0.938
Velocidad de inferencia (GPU)	4.8 ms por imagen	4.9 ms por imagen	5.1 ms por imagen	No disponible
Velocidad de inferencia (CPU)	No disponible	No disponible	No disponible	769.2 ms
Número de clases con >80% precisión	3 clases	3 clases	3 de 5 clases (60%)	3 de 5 clases
Tiempo total de entrenamiento (GPU)	(~33.4 minutos)	(~33.5 minutos)	~33 minutos (0.553 h)	17.375 horas
Tiempo de inferencia por imagen (GPU)	~5.2 ms (incluye pre + post)	No ejecutado en CPU	*No ejecutado en CPU*	Aproximadamente 3.5 a 4.5 horas
Número total de clases	5	5	5	5

Tabla.2. Comparación de rendimiento entre modelos YOLOv10

Detección por clase con YOLOv10

Clase	Precisión (P)	mAP@50
Rana enana de colorado	1.000	0.975
Colibrí de Esmeraldas	0.981	0.973
Viscacha Ecuatorina	0.756	0.823
Iguana marina	0.79	0.83
Tortuga gigante	0.79	0.83

Tabla.3. Detección de clase

Discusión de Resultados

El proceso de entrenamiento y evaluación realizado con la variante YOLOv10b permitió analizar el rendimiento del modelo en condiciones tanto óptimas (GPU) como restringidas (CPU). Los siguientes aspectos destacan del análisis comparativo:

1. Precisión Media y General

La variante YOLOv10b entrenada en CPU alcanzó una precisión media promedio (mAP@50) de 0.923, superior a las variantes YOLOv10s, m y b en GPU, las cuales reportaron un valor uniforme de 0.887. Esto sugiere que, pese al entrenamiento prolongado en CPU (17.375 h), el modelo

logró afinar adecuadamente sus pesos, probablemente debido a un mayor número de epochs sin limitaciones de recursos.
Asimismo, la precisión general (P) y el Recall también fueron mejores en el modelo entrenado en CPU (0.867 y 0.938, respectivamente), en comparación con los entrenados en GPU (0.847 y 0.842). Esto indica que el modelo no solo realiza predicciones más precisas, sino que también detecta una mayor cantidad de objetos reales.

2. Velocidad de Inferencia

Una limitación importante del entrenamiento en CPU fue la velocidad de inferencia, con un promedio de 769.2 ms por imagen, en contraste con apenas ~5 ms por imagen en GPU. Este resultado es esperable, dado que las GPUs están especialmente optimizadas para cálculos paralelos de redes neuronales. Por lo tanto, para entornos en tiempo real o aplicaciones móviles, la inferencia en GPU es altamente recomendable.

3. Rendimiento por Clase

El análisis por clase evidencia que 3 de 5 especies lograron una precisión mAP@50 superior al 80%:

Clase	**Precisión**	**mAP@50**
Rana enana de Colorado	1.000	0.975
Colibrí de Esmeraldas	0.981	0.973
Viscacha Ecuatoriana	0.756	0.823
Iguana marina	0.790	0.830
Tortuga gigante	0.790	0.830

Tabla.4. Precisión por clase

Aunque la clase “Viscacha Ecuatoriana” presentó una precisión baja (0.756), logró un mAP@50 superior al umbral de calidad, evidenciando que el modelo puede localizar correctamente las instancias pero comete algunos errores en clasificaciones o falsos positivos.

4. Impacto del Hardware en el Entrenamiento

El tiempo de entrenamiento marca una diferencia crucial: mientras que el entrenamiento en GPU toma alrededor de 33 minutos, el entrenamiento completo en CPU duró 17.375 horas. Este tiempo extendido no solo ralentiza el desarrollo, sino que también implica mayores costos energéticos y menor capacidad de iterar con rapidez.

5. Comparativa de Rendimiento GPU Vs CPU

La variante YOLOv10b entrenado en CPU superó a todas las versiones evaluadas en GPU en cuanto a precisión y recall, lo cual es destacable dado el entorno más limitado de hardware. No obstante, este rendimiento no compensa las restricciones operativas en tiempo real que impone la velocidad de inferencia en CPU.

Resultados de Procesos de Entrenamiento

Los resultados derivados de los procedimientos descritos en el proceso de entrenamiento se presentan a continuación:

Matriz de confusión

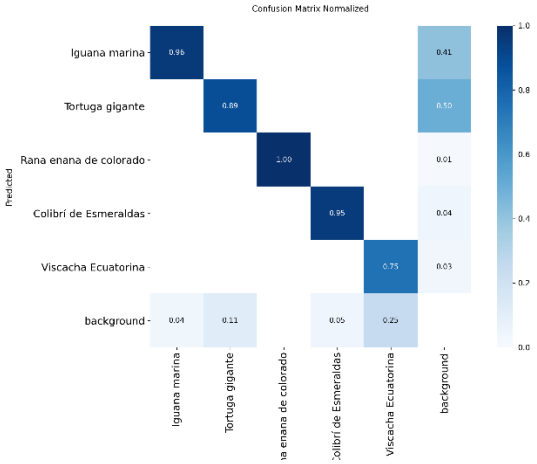

Fig.3. Matriz de confusión Normalizada

Histograma de frecuencias de clases

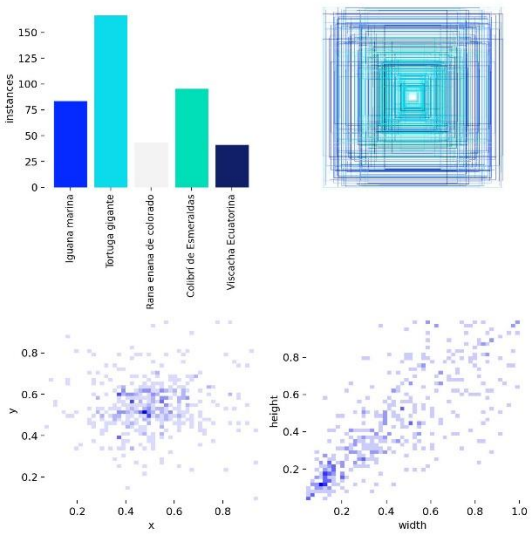

Fig.4. Distribución de etiquetas

Curvas de aprendizaje

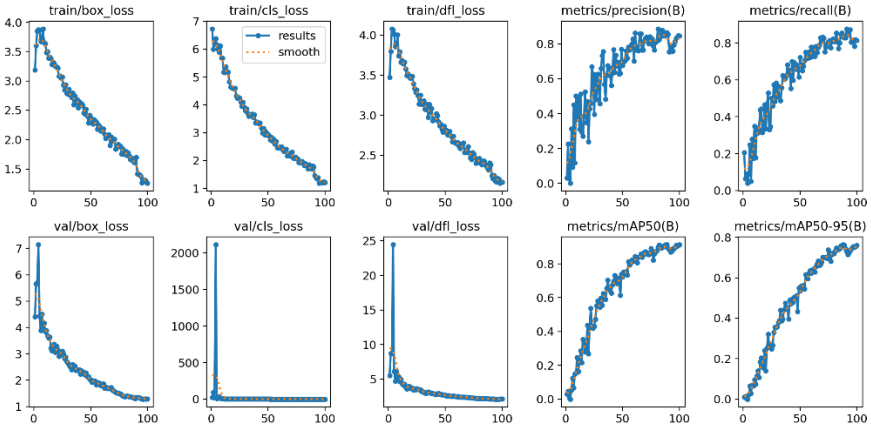

Fig.5. Curvas de Aprendizaje

Curvas de desempeño

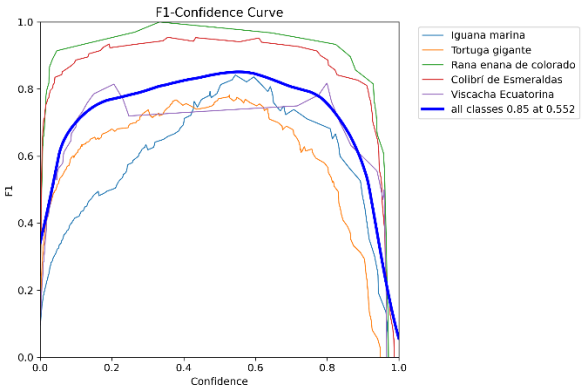

Fig.6. Curvas de Aprendizaje

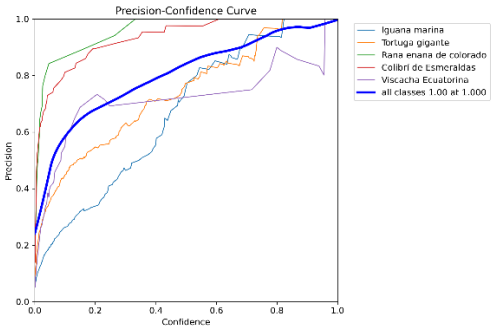

Fig.7. Curvas de Precisión

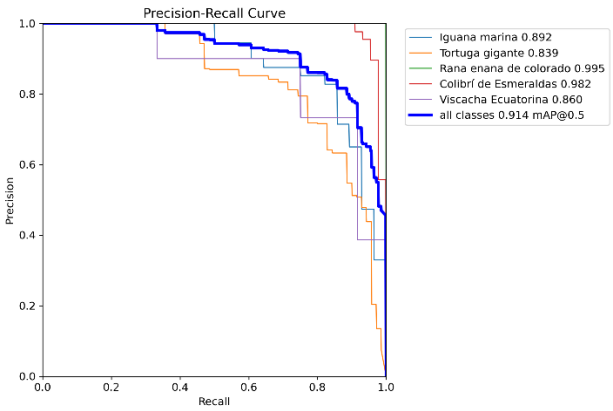

Fig.8. Curvas de Precision-Recall

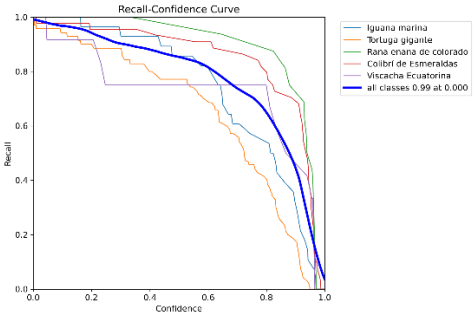

Fig.9. Curvas de Recall

Evaluación

Se realiza la evaluación técnica sobre el desempeño del modelo de detección de especies endémicas (basado en YOLOv10), entrenado con cinco clases.

1. Matriz de Confusión

Muestra buena capacidad de clasificación para clases como Iguana marina y Tortuga gigante.
Algunos errores de clasificación son evidentes, por ejemplo, entre Colibrí de Esmeraldas y Viscacha Ecuatoriana, lo que puede sugerir similitud visual entre estas clases o desbalance en los datos.
La diagonal dominante indica un buen desempeño general.

2. Histograma de Frecuencia de Clases

Hay desbalance evidente: clases como Iguana marina y Tortuga gigante tienen más representaciones que otras como Colibrí de Esmeraldas.
Esto puede influir negativamente en la capacidad del modelo para generalizar en clases minoritarias, como se evidencia en los resultados posteriores.

Heatmap de Bounding Boxes

La distribución espacial de las cajas se concentra en zonas centrales.
Podría introducir sesgos si el modelo no generaliza bien en detecciones periféricas.

3. Gráfico de Dispersión

Se observa agrupación de puntos, lo que sugiere que el modelo logra una separación razonable entre clases en su espacio de características.
Superposición parcial entre grupos puede explicar errores de clasificación en la matriz de confusión.

4. Curvas de Entrenamiento y Validación

Las pérdidas (train/val_loss) disminuyen consistentemente, sin signos de sobreajuste evidente.
Las métricas (precision, recall, mAP50, mAP50-95) muestran una mejora sostenida hasta converger.
Esto indica un entrenamiento estable y eficiente.

5. F1-Confidence Curve

El F1 se maximiza alrededor de umbrales intermedios (~0.4-0.6).
Las clases como Colibrí de Esmeraldas presentan un F1 más bajo, evidenciando desafíos en su detección.
Se puede usar este gráfico para elegir un umbral de confianza óptimo por clase.

6. Precision-Confidence Curve

Todas las clases presentan un aumento de precisión con mayor confianza, como es de esperarse.
Tortuga gigante muestra un comportamiento más errático, posiblemente por escasez de ejemplos o falsos positivos.

7. Precision-Recall Curve

Clases como Rana enana de colorado y Colibrí de Esmeraldas muestran curvas altas, con precisión y recall >0.95.
La Tortuga gigante muestra un rendimiento inferior (área ≈ 0.839), lo que indica un reto en detección fiable para esa clase.

8. Recall-Confidence Curve

La mayoría de las clases disminuyen su recall significativamente al incrementar la confianza.

Esta curva ayuda a visualizar la pérdida de detecciones a medida que se ajustan umbrales más estrictos.

Diseño de aplicación

La aplicación fue desarrollada como una solución de escritorio interactiva con interfaz gráfica (GUI) usando Tkinter y procesado visual mediante OpenCV, orientada a facilitar la detección de especies endémicas a través de dos modos de operación: imágenes desde disco y cámara en vivo.

a) Módulo de Interfaz Gráfica (GUI)

- Diseñado con Tkinter, permite la interacción del usuario a través de botones, ventanas secundarias y etiquetas informativas.
- Presenta un menú principal con dos opciones:
- "Cargar Imagen y Detectar"
- "Tomar Foto y Detectar"
- Contiene una ventana secundaria para captura desde cámara, con vista previa y botón de detección.

b) Módulo de Preprocesamiento de Imágenes

- Cada imagen capturada o cargada es mejorada antes de su paso al modelo mediante:
- CLAHE (Contrast Limited Adaptive Histogram Equalization) para mejorar el contraste local.
- Filtro de agudizado (sharpening) para resaltar bordes y detalles.
- La imagen resultante es convertida a formato RGB para compatibilidad con el modelo YOLOv10.

c) Módulo de Detección con YOLOv10

- Implementado con la librería Ultralytics YOLO.
- El modelo entrenado se carga al iniciar la aplicación (best.pt).
- Al ejecutar la detección:
- Se procesan las coordenadas de los bounding boxes.
- Se extraen clases y niveles de confianza.
- Se aplican anotaciones visuales con los nombres de las especies detectadas.
- Las detecciones son mostradas en una ventana emergente de OpenCV, con etiquetas y marcos dibujados.

Resultados visuales del sistema

A continuación, se presente capturas de pantalla que ilustran el funcionamiento de la aplicación en sus diferentes etapas:

Fig.10. Pantalla principal del sistema

Menú: Cargar Imagen y Detectar

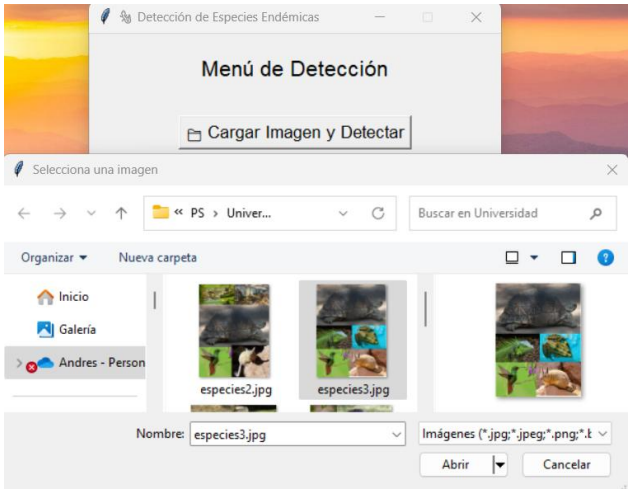

Fig.11. Cargar imagen

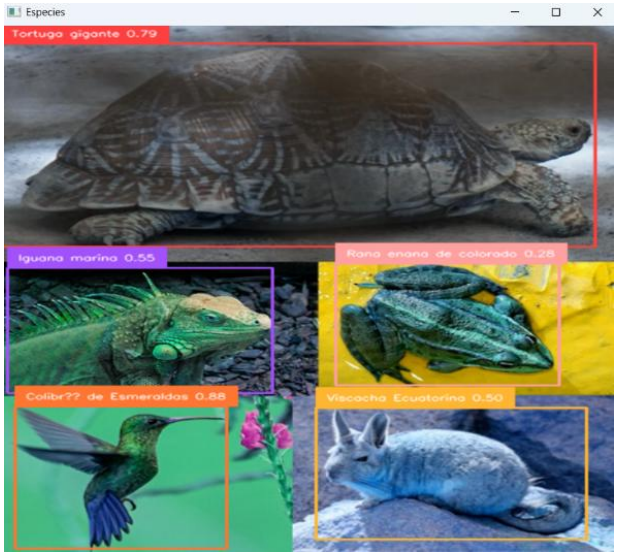

Fig.12. Especies detectadas

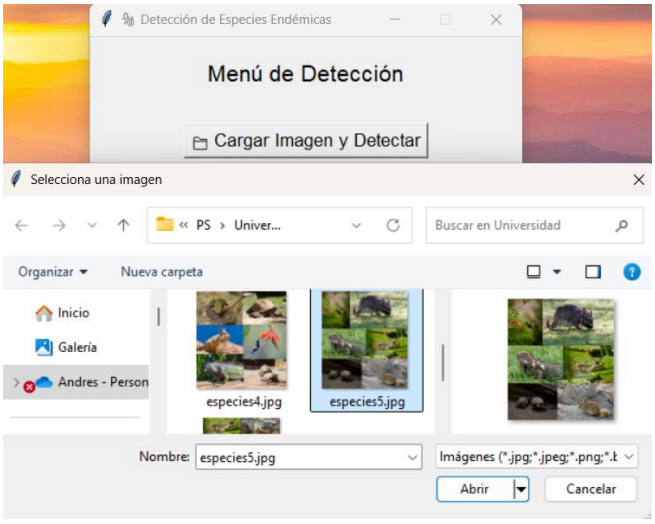

Fig.13. Seleccionar otra imagen

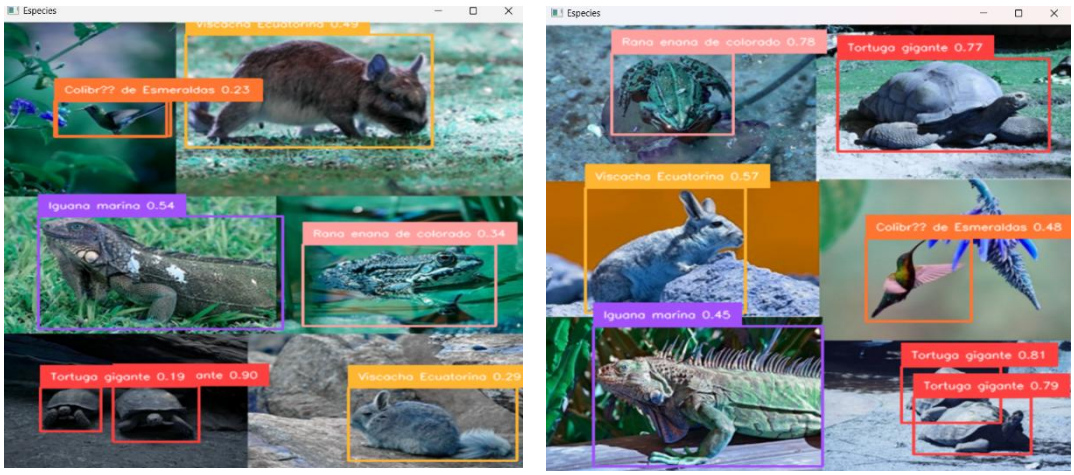

Fig.14. Especies detectadas al cargar otras imágenes

Menú: Tomar Foto y Detectar

Fig.15. Encender cámara de computador

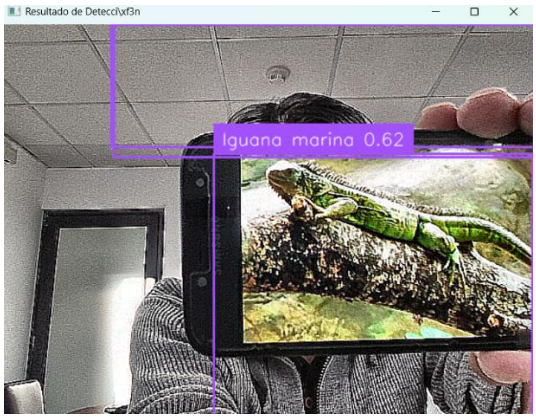

Fig.16. Capturar y Detectar

Conclusiones

- YOLOv10b entrenado en CPU es más preciso, pero no adecuado para entornos en tiempo real.
- Se observó un notable incremento de rendimiento en ejecución con GPU frente a CPU

- La mejora de imagen (CLAHE + sharpen) mejora la calidad de detección sin sacrificar velocidad.
- Las clases de mayor precisión fueron “Rana enana de Colorado” y “Colibrí de Esmeraldas”, con más del 97% de mAP@50.
- El entrenamiento prolongado puede ser ventajoso si se busca máxima precisión en tareas críticas de conservación o biodiversidad, como la detección de especies endémicas.
- El modelo muestra un desempeño robusto con métricas promedio como mAP50 ≈ 0.91, lo que indica alta capacidad de detección.
- Clases como Colibrí de Esmeraldas y Rana enana de colorado tienen excelente desempeño, mientras que Tortuga gigante y Viscacha Ecuatoriana podrían beneficiarse de recolección de más datos o técnicas de balanceo.
- Las curvas de entrenamiento no indican sobreajuste, y las curvas F1/PR revelan umbrales óptimos en el rango medio de confianza.
- Se recomienda ampliar el dataset y explorar técnicas de aumento de datos para mejorar detección de clases menos representadas.
- La aplicación desarrollada cumple con los objetivos del proyecto, integrando una interfaz sencilla para usuarios no técnicos.

Material de soporte – Parte 2

El material de soporte relacionado a la parte 2, que incluye código fuente, scripts, resultados se encuentra disponible en GitHub

Estos materiales permitirán reproducir del sistema presentado

https://github.com/prsizalima/vision-artificial

Bibliografía

[1] R. Szeliski, Computer Vision: Algorithms and Applications, 2nd ed. Springer, 2022.

[2] K. Zuiderveld, "Contrast Limited Adaptive Histogram Equalization," in Graphics Gems IV, P. S. Heckbert, Ed. Academic Press, 1994, pp. 474–485.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779–788.

[4] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan M. Wang, "YOLOv10: Real-Time Object Detection with Speed-Accuracy-Computational Trade-off Optimization," arXiv preprint, arXiv:2404.03900, 2024.

Código

Redmon, J. et al. (2016). You Only Look Once: Unified, Real-Time Object Detection. CVPR.

Bochkovskiy, A. et al. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection.

Ultralytics. (2024). YOLOv10 Documentation

Gonzalez, R., & Woods, R. (2018). Digital Image Processing.

OpenCV Documentation: https://docs.opencv.org

Papers With Code: https://paperswithcode.com/task/object-detection

Medium Tutorials on LBPH and Edge Detection.