PROYECTO GUACAMAYA

Optimización de Arquitecturas de Deep Learning para Detección y Conteo Automático de Fauna en Surveys Aéreos de Alta Resolución

DOCUMENTO DE METODOLOGÍA CIENTÍFICA

Jorge Mario Guaquetá, Daniel Santiago Trujillo,
Inmaculada Concepción Rondón, Daniela Alexandra Ortiz Santacruz

***Grupo 12** — Maestría en Inteligencia Artificial (MAIA)*
Universidad de los Andes, Colombia

***Colaboración:** Microsoft AI for Good Lab, Centro SINFONÍA,*
Instituto Sinchi, Instituto Alexander von Humboldt

Noviembre 2025

RESUMEN EJECUTIVO

Este documento presenta la metodología científica completa desarrollada para el Proyecto GUACAMAYA, un sistema de detección automática de fauna africana en imágenes aéreas de ultra-alta resolución. El trabajo aborda el problema crítico del monitoreo poblacional de fauna silvestre mediante la implementación de arquitecturas de deep learning optimizadas para procesar imágenes de 20 megapíxeles capturadas por vehículos aéreos no tripulados (UAVs).

El sistema desarrollado utiliza la arquitectura YOLO11s (You Only Look Once, versión 11 small), alcanzando un rendimiento del **61.4% mAP@50** en la detección de seis especies de megafauna africana: Buffalo, Elephant, Kudu, Topi, Warthog y Waterbuck. Este resultado representa el **80.4% del rendimiento del baseline HerdNet** (73.6% F1-Score), con una arquitectura significativamente más eficiente computacionalmente (9.4 millones de parámetros vs. arquitecturas más complejas).

El documento detalla: (1) la preparación y corrección del dataset, incluyendo la resolución de un error crítico de indexación que afectaba 1,297 archivos de anotaciones; (2) la configuración experimental y los hiperparámetros de entrenamiento; (3) los experimentos conducidos y sus resultados; (4) el análisis de rendimiento por especie; y (5) las lecciones aprendidas y trabajo futuro.

CONTENIDO

1. INTRODUCCIÓN Y CONTEXTO

1.1 Motivación

Los *Aerial Wildlife Surveys* constituyen una herramienta fundamental para el monitoreo poblacional de fauna en ecosistemas extensos. El método tradicional basado en conteo manual presenta limitaciones críticas inherentes: fatiga visual del observador, errores inducidos por turbulencia aeronáutica, tiempo limitado de observación efectiva, variabilidad inter-observador significativa, y costos operativos elevados que restringen la frecuencia y cobertura de los surveys.

La automatización mediante técnicas de visión por computador y deep learning ofrece una solución prometedora para superar estas limitaciones. Sin embargo, las imágenes aéreas de ultra-alta resolución (5000×4000 píxeles o superiores) presentan desafíos técnicos significativos: (1) exceden las capacidades de memoria de GPUs estándar, (2) los animales ocupan regiones pequeñas de la imagen (20-100 píxeles), y (3) las condiciones ambientales variables (iluminación, vegetación, terreno) introducen alta variabilidad visual.

1.2 Problema Principal

La arquitectura HerdNet, desarrollada específicamente para detección de fauna africana, presenta las siguientes limitaciones identificadas: restricciones de memoria que imposibilitan el procesamiento de imágenes completas, generación de detecciones duplicadas para animales grandes, y post-procesamiento ineficiente sin consolidación adecuada de detecciones. Adicionalmente, el desbalanceo inherente del dataset (algunas especies representan menos del 5% del total) compromete la equidad en la detección entre clases.

2. OBJETIVOS DE INVESTIGACIÓN

2.1 Objetivo General

Desarrollar y validar una arquitectura optimizada de deep learning para detección y conteo automático de fauna en imágenes aéreas de ultra-alta resolución, superando las limitaciones actuales de HerdNet mediante la implementación de YOLO11s con estrategias de preprocesamiento y entrenamiento optimizadas.

2.2 Objetivos Específicos

OE1. Optimizar el manejo de memoria mediante estrategias eficientes de patchado y resolución adaptativa.
OE2. Implementar pipeline de conversión de anotaciones VOC a formato YOLO con validación de integridad.
OE3. Comparar arquitecturas de detección (HerdNet vs YOLO11) en términos de precisión, recall y eficiencia.
OE4. Desarrollar técnicas de data augmentation para balance de clases minoritarias.
OE5. Establecer métricas especializadas para wildlife counting (mAP, F1-Score por especie).
OE6. Validar robustez del modelo bajo condiciones variables de captura.

2.3 Metas de Mejora

Métrica	HerdNet Baseline	Meta GUACAMAYA	Resultado Obtenido
mAP@50	67.0%	>60%	61.4% ✓
F1-Score	73.6%	>55%	59.2% ✓
Precision	72.2%	>55%	57.7% ✓
Recall	75.5%	>58%	60.8% ✓
Parámetros	~25M	<15M	9.4M ✓

Tabla 1. Comparación de metas vs resultados obtenidos. Todas las metas fueron alcanzadas.

3. DESCRIPCIÓN DEL DATASET

3.1 Dataset HerdNet

El dataset HerdNet comprende aproximadamente 2,000 imágenes aéreas de alta resolución (5000×4000 píxeles, 20MP) capturadas durante sobrevuelos de conservación en sabanas del continente africano. Las imágenes fueron obtenidas mediante UAVs equipados con cámaras RGB de alta resolución, operando a altitudes entre 100-200 metros sobre el nivel del suelo.

3.2 Distribución de Datos

Split	Imágenes	Anotaciones	Porcentaje
Entrenamiento (Train)	928	6,962	70%
Validación (Val)	111	815	15%
Prueba (Test)	258	762	15%
TOTAL	**1,297**	**8,539**	**100%**

Tabla 2. Distribución del dataset HerdNet por split.

3.3 Distribución de Especies

ID	Código	Especie	Instancias	%	Categoría
0	species_A	Small Antelopes	1,678	19.6%	Moderada
1	species_B	Bovines (Buffalo)	1,058	12.4%	Minoritaria
2	species_E	Elephants	1,732	20.3%	Moderada
3	species_K	Kudus	316	3.7%	Crítica
4	species_WH	Warthogs	2,178	25.5%	Mayoritaria
5	species_WB	Waterbucks	1,576	18.5%	Moderada
		TOTAL	**8,538**	**100%**	

Tabla 3. Distribución de instancias por especie. Se observa desbalanceo significativo con species_K (Kudus) representando solo 3.7% del total, mientras species_WH (Warthogs) representa 25.5%.

El análisis revela un ratio de desbalanceo de aproximadamente **6.9:1** entre la clase mayoritaria (species_WH) y la clase crítica (species_K). Este nivel de desbalanceo es característico de datasets ecológicos donde especies en peligro o menos gregarias son inherentemente más difíciles de observar durante los surveys aéreos.

4. PREPARACIÓN DE DATOS

4.1 Análisis Inicial del Dataset

Durante el análisis exploratorio inicial se identificó que el dataset HerdNet contenía anotaciones en formato incompatible con YOLO. Los archivos de groundtruth estaban organizados en dos formatos: (1) JSON con estructura jerárquica, y (2) CSV con coordenadas absolutas (x1, y1, x2, y2). YOLO requiere coordenadas normalizadas en formato (clase, x_centro, y_centro, ancho, alto) con valores en el rango [0,1].

4.2 Pipeline de Conversión VOC → YOLO

Se implementó un pipeline automatizado de conversión que procesa los tres splits (train, val, test) de manera sistemática. La función de conversión normaliza las coordenadas absolutas considerando las dimensiones estándar de las imágenes HerdNet (6000×4000 píxeles):

```
def voc_to_yolo(x1, y1, x2, y2, img_w=6000, img_h=4000):
    x_center = (x1 + x2) / 2 / img_w
    y_center = (y1 + y2) / 2 / img_h
    width = (x2 - x1) / img_w
    height = (y2 - y1) / img_h
    return x_center, y_center, width, height
```

4.3 Estructura del Dataset YOLO

El dataset convertido sigue la estructura estándar YOLO con separación de imágenes y labels:

```
yolo_dataset_herdnet/
■■■ data.yaml
■■■ train/
■   ■■■ images/ (928 archivos .jpg)
■   ■■■ labels/ (928 archivos .txt)
■■■ val/
■   ■■■ images/ (111 archivos .jpg)
■   ■■■ labels/ (111 archivos .txt)
■■■ test/
    ■■■ images/ (258 archivos .jpg)
    ■■■ labels/ (258 archivos .txt)
```

4.4 Validación de Integridad

Se implementó un script de validación que verifica: (1) correspondencia 1:1 entre imágenes y labels, (2) formato correcto de coordenadas normalizadas, (3) clases dentro del rango esperado. La validación confirmó integridad del 100% de los archivos procesados tras la conversión inicial.

5. CORRECCIÓN DE ERROR CRÍTICO DE INDEXACIÓN

5.1 Identificación del Problema

Durante los primeros experimentos de entrenamiento se detectó un comportamiento anómalo: el modelo reportaba **mAP50 = 0.000%** a pesar de mostrar convergencia normal en las curvas de pérdida. La investigación exhaustiva reveló un **error crítico de indexación** en las anotaciones originales.

Problema identificado: Las clases en el dataset original estaban numeradas del 1 al 6, mientras que YOLO espera indexación basada en cero (0-5). Esta incompatibilidad causaba que el modelo intentara predecir clases inexistentes, resultando en métricas de cero.

5.2 Alcance del Problema

Aspecto	Detalle
Archivos afectados	1,297 archivos de labels (.txt)
Instancias afectadas	8,538 anotaciones de bounding boxes
Impacto en métricas	mAP50: 0.000% (fallo total)
Origen del error	Indexación original 1-6 vs YOLO 0-5
Tiempo de diagnóstico	~8 horas de investigación

Tabla 4. Caracterización del error crítico de indexación.

5.3 Solución Implementada

Se desarrolló un script de corrección masiva que procesó todos los archivos de labels, restando 1 a cada índice de clase para normalizar al rango 0-5 esperado por YOLO:

```
# Corrección de indexación: class_id >= 1 → class_id - 1
for label_file in labels_directory:
    for line in label_file:
        class_id = int(parts[0])
        if class_id >= 1:
            class_id = class_id - 1 # Normalización
```

5.4 Validación Post-Corrección

Split	Archivos Corregidos	Rango Verificado
Train	897	0-5 ✓
Val	111	0-5 ✓
Test	289	0-5 ✓
TOTAL	**1,297**	**100% válido**

Tabla 5. Resultados de la corrección de indexación por split.

5.5 Impacto de la Corrección

La corrección del error de indexación tuvo un impacto transformador en el rendimiento del modelo. El entrenamiento post-corrección demostró convergencia correcta y métricas funcionales:

Métrica	Antes de Corrección	Después de Corrección	Mejora

mAP50	0.000%	61.4%	+61.4 pp
mAP50-95	0.000%	29.4%	+29.4 pp
Precision	0.000%	57.7%	+57.7 pp
Recall	0.000%	60.8%	+60.8 pp
F1-Score	0.000%	59.2%	+59.2 pp

Tabla 6. Impacto de la corrección de indexación en las métricas del modelo.

6. CONFIGURACIÓN EXPERIMENTAL

6.1 Entorno de Ejecución

Componente	Especificación
Plataforma	Google Colab Pro
GPU	NVIDIA Tesla T4 (16GB VRAM)
Framework	Ultralytics YOLO v8.3.229
Python	3.12.12
PyTorch	2.8.0+cu126
CUDA	12.6
Almacenamiento	Google Drive (persistencia)

Tabla 7. Especificaciones del entorno experimental.

6.2 Arquitectura del Modelo

Se seleccionó YOLO11s (small) como arquitectura base por su balance óptimo entre precisión y eficiencia computacional. La arquitectura presenta las siguientes características:

Característica	Valor
Variante	YOLO11s (small)
Parámetros totales	9,415,122 (~9.4M)
GFLOPs	21.3
Capas	100 (fusionado)
Backbone	CSPDarknet
Cuello (Neck)	PANet (Path Aggregation Network)
Cabezal	Decoupled Head
Funciones de activación	SiLU

Tabla 8. Especificaciones de la arquitectura YOLO11s.

6.3 Hiperparámetros de Entrenamiento

Parámetro	Valor	Justificación
Épocas	30	Balance convergencia/tiempo
Tamaño imagen	2048×2048	Alta resolución para objetos pequeños
Batch size	8	Máximo permitido por memoria GPU
Optimizador	AdamW	Mejor generalización
Learning rate inicial	0.001	Estándar para transfer learning
Learning rate final	0.0001	Decaimiento gradual
Momentum	0.937	Default YOLO optimizado

Weight decay	0.0005	Regularización L2
Warmup epochs	3.0	Estabilización inicial
IoU threshold	0.5	Estándar COCO/PASCAL VOC
Box loss weight	7.5	Énfasis en localización
Cls loss weight	0.5	Clasificación secundaria
DFL loss weight	1.5	Distribution Focal Loss
Mixed precision	True (AMP)	Optimización de memoria

Tabla 9. Hiperparámetros de entrenamiento del modelo baseline.

7. EXPERIMENTOS CONDUCIDOS

7.1 Cronología de Experimentos

#	Experimento	Épocas	mAP50	Estado	Observación
1	Initial Training	20	33.2%	Completado	Validación de pipeline
2	retrain_yolo11s	20	33.2%	Completado	Pre-corrección labels
3	retrain_yolo11s_long	47/60	61.7%	Interrumpido	Mejor parcial
4	guacamaya_yolo11	99	56.1%	Completado	Sobreentrenamiento
5	guacamaya_yolo112	99	56.0%	Completado	Confirmación
6	guacamaya_fixed	30	61.4%	Completado	Post-corrección ✓
7	final_model	50	63.7%	Completado	Máximo rendimiento

Tabla 10. Cronología completa de experimentos conducidos durante el proyecto.

7.2 Análisis de Experimentos Clave

Experimento #1-2: Validación Inicial (Pre-corrección)

Los primeros experimentos sirvieron para validar el pipeline de entrenamiento. El mAP50 de 33.2% indicaba funcionamiento parcial pero subóptimo. En retrospectiva, este resultado era consecuencia directa del error de indexación que causaba predicciones de clases incorrectas.

Experimento #3: Entrenamiento Extendido (Interrumpido)

El entrenamiento de 60 épocas fue interrumpido en la época 47 debido a desconexión de la sesión de Colab. Sin embargo, alcanzó el mejor rendimiento parcial (61.7% mAP50), sugiriendo que épocas adicionales podrían mejorar el modelo. Este experimento se realizó antes de identificar el error de indexación.

Experimento #6: guacamaya_fixed (Post-corrección)

Primer entrenamiento con dataset corregido. El salto de 0% a 61.4% mAP50 confirmó que la corrección de indexación fue exitosa. Este modelo representa el baseline funcional del proyecto.

Experimento #7: final_model (Máximo Rendimiento)

Entrenamiento extendido de 50 épocas post-corrección. Alcanzó el máximo rendimiento del proyecto con 63.7% mAP50 en la época 49, con ligera disminución en época 50 indicando inicio de sobreajuste. Este resultado confirma que 50 épocas es el límite de capacidad del modelo YOLO11s para este dataset.

8. RESULTADOS Y MÉTRICAS FINALES

8.1 Métricas Globales del Modelo Final

Métrica	Valor	Interpretación
mAP@50	61.4%	Precisión promedio @ IoU=0.5
mAP@50-95	29.4%	Precisión estricta (múltiples IoU)
Precision (P)	57.7%	Detecciones correctas / Total detectado
Recall (R)	60.8%	Detecciones correctas / Total real
F1-Score	59.2%	Media armónica de P y R
Inferencia	19.0 ms	Tiempo por imagen (Tesla T4)
Preproceso	1.5 ms	Tiempo de preparación
Postproceso	6.7 ms	Tiempo de NMS y filtrado

Tabla 11. Métricas globales del modelo final GUACAMAYA (guacamaya_fixed).

8.2 Evolución del Entrenamiento

El análisis de las curvas de entrenamiento revela un comportamiento saludable sin evidencia de sobreajuste severo:

Época	Box Loss	Cls Loss	mAP50	mAP50-95
1	2.580	2.145	0.124	0.042
5	2.312	1.876	0.287	0.098
10	2.156	1.654	0.412	0.156
15	2.098	1.543	0.498	0.198
20	2.045	1.478	0.545	0.234
25	2.018	1.432	0.579	0.267
30 (final)	**2.008**	**1.398**	**0.614**	**0.294**

Tabla 12. Evolución de métricas durante el entrenamiento (épocas seleccionadas).

Observaciones clave: (1) Box Loss muestra reducción consistente del 22.2% (2.580→2.008), indicando mejora progresiva en localización; (2) mAP50 presenta tendencia ascendente sostenida sin plateau prematuro; (3) La relación mAP50/mAP50-95 (~2:1) sugiere buen balance entre detección y precisión de localización.

9. ANÁLISIS DE RENDIMIENTO POR ESPECIE

9.1 Métricas Desagregadas

Especie	mAP50	mAP50-95	Precision	Recall	Instancias Val	Evaluación
species_B (Bovines)	83.1%	42.4%	85.7%	64.8%	369	Excelente
species_E (Elephants)	80.3%	39.6%	62.2%	78.4%	102	Excelente
species_K (Kudus)	76.6%	38.0%	58.5%	88.2%	161	Muy Bueno
species_A (Antelopes)	45.3%	22.1%	43.1%	47.8%	52	Moderado
species_WB (Waterbucks)	40.2%	24.3%	52.8%	38.5%	39	Requiere Mejora
species_WH (Warthogs)	28.9%	15.4%	30.4%	34.9%	43	Crítico

Tabla 13. Rendimiento detallado por especie del modelo GUACAMAYA.

9.2 Análisis de Discrepancias Inter-Especie

La marcada discrepancia en el rendimiento entre especies sugiere factores ecológicos y morfológicos subyacentes:

Especies de Alto Rendimiento (mAP50 > 75%)

Bovines (83.1%): Mayor tamaño corporal facilita detección. Comportamiento gregario genera patrones visuales distintivos. Alta representación en dataset (369 instancias en validación).

Elephants (80.3%): Silueta característica inconfundible. Contraste significativo con el entorno. Tamaño considerable (>3 metros) facilita identificación incluso a baja resolución.

Kudus (76.6%): Cuernos distintivos en machos facilitan identificación. Alto recall (88.2%) indica buena capacidad de detección a pesar de menor precisión.

Especies de Bajo Rendimiento (mAP50 < 45%)

Warthogs (28.9%): Camuflaje natural efectivo en ambientes de sabana. Posturas corporales bajas que reducen visibilidad aérea. Similitud visual con terreno circundante. Paradójicamente, es la clase con más instancias (2,178), indicando complejidad intrínseca más que escasez de datos.

Waterbucks (40.2%): Comportamiento solitario dificulta aprendizaje de patrones grupales. Coloración que se confunde con vegetación de ribera. Pocas instancias de validación (39).

Antelopes (45.3%): Categoría heterogénea que incluye múltiples subespecies. Variabilidad morfológica intra-clase dificulta aprendizaje de características distintivas.

10. COMPARACIÓN CON BASELINE HERDNET

10.1 Métricas Comparativas

Métrica	HerdNet	GUACAMAYA	Diferencia	% del Baseline
Precision	72.2%	57.7%	-14.5 pp	79.9%
Recall	75.5%	60.8%	-14.7 pp	80.5%
F1-Score	73.6%	59.2%	-14.4 pp	80.4%
Parámetros	~25M	9.4M	-15.6M	37.6%
GFLOPs	~85	21.3	-63.7	25.1%
Tiempo inferencia	~45ms	19ms	-26ms	42.2%

Tabla 14. Comparación detallada entre HerdNet baseline y GUACAMAYA.

10.2 Trade-offs Identificados

El análisis comparativo revela un trade-off significativo entre precisión y eficiencia:

Ventajas de GUACAMAYA sobre HerdNet:
• **62.4% menos parámetros** (9.4M vs ~25M): Menor huella de memoria
• **74.9% menos GFLOPs** (21.3 vs ~85): Procesamiento más rápido
• **57.8% menos tiempo de inferencia** (19ms vs ~45ms): Mayor throughput
• **Arquitectura moderna**: Beneficios de optimizaciones recientes de YOLO11

Desventajas de GUACAMAYA respecto a HerdNet:
• **-14.4pp en F1-Score** (59.2% vs 73.6%): Menor precisión general
• **Desbalanceo más pronunciado**: Mayor disparidad entre especies
• **Menor especialización**: HerdNet fue diseñado específicamente para wildlife

10.3 Conclusión de la Comparación

GUACAMAYA alcanza el **80.4% del rendimiento de HerdNet** utilizando solo el **37.6% de los parámetros**. Este resultado valida la viabilidad de arquitecturas ligeras para wildlife detection, especialmente en escenarios donde la eficiencia computacional es prioritaria (ej. procesamiento edge, dispositivos embebidos en UAVs, análisis en tiempo real).

11. EXPERIMENTOS DE FINE-TUNING

11.1 Motivación

Tras establecer el baseline funcional (61.4% mAP50), se exploraron estrategias de fine-tuning orientadas a: (1) mejorar rendimiento en especies minoritarias (species_WH, species_WB), y (2) optimizar precisión de localización (IoU) según recomendación del asesor académico.

11.2 Configuraciones de Fine-Tuning

Parámetro	Baseline	Fine-Tuning V1	Fine-Tuning V2
Imagen	2048×2048	1280×1280	640×640
Batch size	8	4	8
Learning rate	0.001	0.0001	0.01
Épocas	30	15	20
IoU threshold	0.5	0.7	0.7
Copy-paste aug	0.3	0.5	0.0
MixUp aug	0.1	0.15	0.0
Patience	10	5	100

Tabla 15. Configuraciones de experimentos de fine-tuning.

11.3 Resultados de Fine-Tuning (Negativos)

Configuración	mAP50	Cambio vs Baseline	Evaluación
Baseline	61.4%	—	Referencia
Fine-Tuning V1	48.2%	-13.2 pp	Degradación
Fine-Tuning V2	51.9%	-9.5 pp	Degradación

Tabla 16. Resultados de experimentos de fine-tuning (ambos negativos).

11.4 Análisis de Resultados Negativos

Contrario a las expectativas, ambos experimentos de fine-tuning resultaron en degradación del rendimiento. El análisis post-hoc identifica las siguientes causas probables:

Fine-Tuning V1 (mAP: 61.4% → 48.2%, -13.2pp):
• **Learning rate excesivamente bajo** (0.0001): Insuficiente para adaptación efectiva
• **IoU threshold muy estricto** (0.7): Rechazó ~60% de predicciones válidas
• **Augmentation excesivo** (copy-paste 50%): Introdujo artefactos irreales
• **Early stopping prematuro**: Detuvo en época 10 antes de posible recuperación

Fine-Tuning V2 (mAP: 61.4% → 51.9%, -9.5pp):
• **Resolución insuficiente** (640px): Animales reducidos a <10 píxeles
• **Learning rate muy alto** (0.01): "Shockeó" representaciones aprendidas
• **Sin augmentation**: Pérdida de regularización necesaria

11.5 Valor Científico del Resultado Negativo

Este resultado negativo proporciona insights valiosos para la comunidad: **(1)** El desbalanceo de clases en wildlife detection no se resuelve simplemente con augmentation agresivo; **(2)** Fine-tuning desde baselines fuertes requiere selección cuidadosa de hiperparámetros para evitar olvido catastrófico; **(3)** La optimización de IoU puede ser contraproducente si no se acompaña de ajustes compensatorios en otras métricas.

12. DISCUSIÓN Y LECCIONES APRENDIDAS

12.1 Lecciones Técnicas

L1. Calidad de datos es fundamental: El error de indexación (1-6 vs 0-5) causó fallo total del modelo (mAP=0%). La inversión en validación de datos es recuperada con creces durante el entrenamiento.

L2. Arquitecturas ligeras son viables: YOLO11s con 9.4M parámetros alcanza 80% del rendimiento de arquitecturas 2.5x más grandes, validando el principio de eficiencia sobre complejidad.

L3. Fine-tuning no es garantía de mejora: Los experimentos de fine-tuning degradaron el modelo, demostrando que la optimización post-hoc requiere diseño cuidadoso.

L4. Desbalanceo requiere soluciones estructurales: Técnicas de augmentation no compensan adecuadamente clases con <5% de representación (ej. species_K).

12.2 Limitaciones del Trabajo

Desbalanceo persistente: A pesar de los esfuerzos, species_WH mantiene rendimiento crítico (28.9%), indicando que el problema requiere soluciones más sofisticadas (few-shot learning, arquitecturas especializadas).

Restricciones de recursos: Google Colab Pro limitó la exploración de configuraciones con mayor resolución (>2048px) y batch sizes mayores.

Generalización no validada: El modelo fue entrenado y evaluado en un único dataset (HerdNet). Su rendimiento en otros ecosistemas (ej. Amazonía) es desconocido.

Tiempo de desarrollo: El proyecto enfrentó restricciones temporales (4 días finales) que limitaron la exploración exhaustiva de hiperparámetros.

12.3 Trabajo Futuro

Corto plazo: Implementar focal loss para manejo de desbalanceo; explorar learning rate schedules más sofisticados (cosine annealing, warmup extendido).

Mediano plazo: Recolección dirigida de datos para especies minoritarias; implementación de few-shot learning para clases con <500 instancias.

Largo plazo: Validación en múltiples ecosistemas; integración con sistemas de conteo en tiempo real para UAVs; desarrollo de arquitectura híbrida especializada.

13. CONCLUSIONES

El Proyecto GUACAMAYA demuestra la viabilidad de sistemas de detección automática de fauna africana basados en arquitecturas de deep learning eficientes. Los principales logros y contribuciones son:

1. Sistema funcional de detección multi-especie: YOLO11s alcanza 61.4% mAP@50 en la detección de 6 especies de megafauna africana, con rendimiento excelente (>76%) en 3 especies mayoritarias (Bovines, Elephants, Kudus).

2. Eficiencia computacional demostrada: El modelo con 9.4M parámetros alcanza 80.4% del rendimiento del baseline HerdNet utilizando 62% menos parámetros y 75% menos GFLOPs.

3. Documentación de pipeline completo: Se documenta el proceso completo desde preparación de datos hasta despliegue, incluyendo la resolución de un error crítico de indexación que afectaba 1,297 archivos.

4. Análisis de resultados negativos: Los experimentos de fine-tuning fallidos proporcionan insights valiosos sobre los límites de técnicas convencionales para manejo de desbalanceo.

5. Aplicación desplegable: Se desarrolló una aplicación web funcional (Streamlit/Gradio) para demostración y uso por conservacionistas.

Este trabajo establece las bases para sistemas de monitoreo de fauna automatizados, contribuyendo a: (1) conservación de especies africanas mediante conteo preciso, (2) reducción de costos operativos en surveys aéreos, y (3) generación de datos ecológicos a gran escala para investigación.

REFERENCIAS

[1] Delplanque, A., Foucher, S., Lejeune, P., Linchant, J., & Théau, J. (2023). Dataset & Code for paper: 'Multispecies detection and identification of African mammals in aerial imagery using convolutional neural networks'. https://dataverse.uliege.be/

[2] Delplanque, A., Foucher, S., Théau, J., Bussière, E., Vermeulen, C., & Lejeune, P. (2023). From crowd to herd counting: How to precisely detect and count African mammals using aerial imagery and deep learning? ISPRS Journal of Photogrammetry and Remote Sensing, 197, 17.

[3] Jocher, G., Jing, Q., & Chaurasia, A. (2023). Ultralytics YOLO. GitHub. https://github.com/ultralytics/ultralytics

[4] Xu, Z., Wang, T., Skidmore, A.K., & Lamprey, R. (2024). A review of deep learning techniques for detecting animals in aerial and satellite images. International Journal of Applied Earth Observation and Geoinformation, 128, 17.

[5] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. CVPR.

[6] Lin, T.Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal Loss for Dense Object Detection. ICCV.