Implementación de Drones en la Ganadería de Precisión para el Conteo y Seguimiento de Animales

Sergio Andrés Angarita Camacho, Andrés Felipe Araque Guerrero y Jhonathan Alexander Murcia Galán

Trabajo de Grado para Optar al Título de Ingeniero Electrónico

Director

Jaime Guillermo Barrero Perez

Mg.

Codirector

Carlos Augusto Fajardo Ariza

PhD.

Universidad Industrial de Santander

Facultad de Ingenierías Fisicomecánicas

Escuela E3T

Ingeniería Electrónica

Bucaramanga

2024

Tabla de Contenido

Pág.

Lista de Tablas

Pág.

Lista de Figuras

Pág.

Resumen

Título: Implementación de Drones en la Ganadería de Precisión para el Conteo y Seguimiento de Animales[*]

Autor: Sergio Andrés Angarita Camacho, Andrés Felipe Araque Guerrero y Jhonathan Alexander Murcia Galán[**]

Palabras Clave: Vigilancia aérea, robo de ganado, dron, base de datos, ganado, YoloV8

Descripción: En Colombia, la ganadería representa una importante fuente de ingresos, contribuyendo en un 27% al Producto Interior Bruto agrícola. Sin embargo, esta actividad se enfrenta a diversos retos para garantizar su correcto desarrollo, como el robo de ganado, conocido como abigeato. Este problema es el que se pretende abordar con este proyecto, que propone un sistema de recuento y seguimiento a través de la vigilancia aérea del ganado. Para ello se utilizan imágenes captadas por un dron y procesadas posteriormente por un algoritmo de inteligencia artificial (IA). Un aspecto clave del proyecto es la creación de una base de datos con 1546 imágenes aéreas de ganado en 4 localidades de Santander y Boyacá, Colombia. Tras un riguroso etiquetado de las imágenes, se procedió al entrenamiento del modelo YoloV8m. Con la implementación de este modelo, logramos una exactitud del 95,1%, una precisión del 96,1%, una sensibilidad del 98,9% y una métrica F1 del 97,47%. Estos resultados validan la capacidad del modelo para detectar vacas.

[*] Trabajo de Grado

[**] Facultad de Ingenierías Fisicomecánicas. Escuela de E3T. Ingeniería Electrónica. Director: Jaime Guillermo Barrero Perez. Mg. Codirector: Carlos Augusto Fajardo Ariza. PhD.

Abstract

Title: Implementation of Drones in Precision Livestock Farming for AnimalCounting and Tracking*

Author(s): Sergio Andrés Angarita Camacho, Andrés Felipe Araque Guerrero y Jhonathan Alexander Murcia Galán**

Key Words: Aerial surveillance, cattle rustling, drone, database, livestock, YoloV8

Description: In Colombia, livestock farming represents an important source of income, contributing 27% to agricultural Gross Domestic Product. However, this activity faces various chal-lenges to ensure its proper development, such as cattle theft, known as rustling. This problem is what we seek to address with this project, which proposes a counting and monitoring system through aerial surveillance of livestock. Such results are achieved using images captured by a drone and further processed by an artificial intelligence (AI)algorithm. Remarkably, a key aspect of the project is the establishment of a database, col- lecting 1546 aerial images of livestock in 4 locations in Santander and Boyacá, Colom-bia. After rigorous labeling of the images, we proceeded to train the YoloV8m model. With the implementation of this model, we achieved an accuracy of 95.1%, precision of 96.1%, a sensitivity of 98.9%, and F1 metric of 97.47%. These results validate the model's ability to detect cows.

* Degree Work

**Faculty of Fisicomecanicas. School E3T. Electronic Engineering. Director: Jaime Guillermo Barrero Perez. Mg. Codirector: Carlos Augusto Fajardo Ariza. PhD.

Introducción

El robo de ganado, técnicamente conocido como "cuatrerismo", supone un importante reto para la ganadería en Colombia. Según la información proporcionada por el Ministerio de Agricultura, la ganadería prevalece en 27 de los 32 departamentos del país. En 2022, se denunció el robo de más de 1350 cabezas de ganado, lo que repercute en la economía y la seguridad de los ganaderos. (3)

En relación con este problema, se han aplicado diversas soluciones, algunas de las cuales implican el uso de inteligencia artificial. Andrew et al. (2017) presentaron un sistema que utiliza imágenes captadas por drones y visión computacional deep learning para la identificación y localización de un tipo específico de ganado (Holstein Friesian), alcanzando niveles de precisión del 99,3% (4). La principal aportación de este trabajo radica no solo en la detección de ganado, sino también en la individualización de los animales. Puetaman et al. (2014) propusieron un prototipo de dispositivo basado en tecnología XBee colocado en el cuello del ganado para determinar su estado de vida mediante la monitorización de la temperatura. Además, este dispositivo puede identificar si el animal se encuentra dentro de la zona de pastoreo permitida, emitiendo una señal de radiofrecuencia transmitida desde el dispositivo a un sistema central (5).

Por otra parte, en Colombia, el sector agrícola se caracteriza por un nivel relativamente bajo de adopción tecnológica (6). A falta de tecnología, en muchas partes del país se emplean soluciones más tradicionales, como patrullas, guardias rurales, protestas y, en casos extremos, el uso de armas de fuego en defensa propia contra los robos (7). Aunque habituales, estas medidas suelen ser menos eficaces que las tecnologías más avanzadas desarrolladas para el control y la supervisión del ganado.

Chamoso et al. (2014) utilizaron imágenes aéreas captadas por drones y redes neuronales convolucionales para el recuento de ganado en granjas, logrando precisiones de recuento superiores

al 97% (8). El uso de la inteligencia artificial se ha revelado como una medida tecnológica muy eficaz para el recuento preciso del ganado, contribuyendo significativamente a la lucha contra el abigeato. Las técnicas actuales de recuento de ganado en la industria de la carne de vacuno se consideran anticuadas y costosas, y se basan principalmente en tres enfoques: recuento total, recuento de parcelas y marcado y recuento (9).

Para hacer frente a esta necesidad identificada, se propone la utilización de un algoritmo que permita el recuento de cabezas de ganado mediante el empleo de drones, animando a los ganaderos a adoptar esta nueva tecnología con fines de censo automatizado.

Ante la dificultad de encontrar una base de datos adecuada, optamos por crear una propia, adaptada a los requisitos del proyecto.

1. Métodos

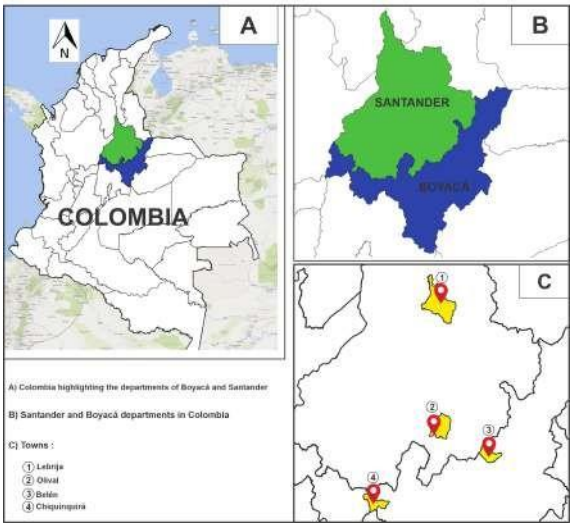
A lo largo del desarrollo del proyecto, pasamos por siete fases clave. Comenzamos con la adquisición de las imágenes para la base de datos, seguida de la tarea de etiquetado de imágenes. Posteriormente, realizamos una búsqueda de modelos capaces de detectar objetos y seleccionamos el adecuado. Evaluamos su rendimiento mediante la técnica de validación cruzada k-Fold y concluimos con el desarrollo de la aplicación móvil, que permite presentar los resultados de forma sencilla.

1.1 Adquisición

Colombia cuenta con varias regiones ganaderas. Sin embargo, los pueblos que nos acogieron se encuentran en Santander y Boyacá. Específicamente, Olival, Lebrija, Chiquinquirá y Belén (Fig. 1) fueron los lugares donde obtuvimos permisos para volar alrededor de aproximadamente 10 granjas en varios lugares.

Figura 1

Ciudades colombianas donde están situadas las explotaciones, que nos permitieron capturar las fotos para el conjunto de datos



Nota. Adaptado de Google Maps

Acceso al mapa En el siguiente enlace puede acceder al mapa interactivo con las localizaciones: Ubicaciones de los proyectos de Ganadería de Precisión

Dron El (2) Mini 2 destaca por ser un dron ligero y compacto, diseñado pensando en la portabilidad y la facilidad de uso. Fabricado por DJI, pionera en tecnología de drones, este modelo representa una iteración mejorada de su precursor, el DJI Mavic Mini. Con un peso de apenas 249 gramos, el Mini 2(Figura 2) está exento de los requisitos de registro en determinadas jurisdicciones, lo que lo convierte en una opción atractiva tanto para principiantes como para entusiastas.

Figura 2

El DJI Mini 2, un dron ligero y compacto diseñado para ser portátil y fácil de usar

Nota. Adaptado de DJI Mini 2

A pesar de su diminuto tamaño, el Mini 2 cuenta con una serie de funciones avanzadas de vuelo e imagen. Equipado con un estabilizador de tres ejes, garantiza tomas suaves y estables, incluso en condiciones de viento moderado. La cámara integrada impresiona, capaz de grabar vídeos en una impresionante resolución 4K a 30 fotogramas por segundo y de capturar fotos nítidas de 12 megapíxeles. Para una visión general de las especificaciones de la DJI Mini 2, consulta la Tabla I. La Tabla 1 proporciona las especificaciones clave del dron DJI Mini 2 utilizado en la adquisición del conjunto de datos.

Tabla 1

Detalles del (2) Mini 2, Dron utilizado en la adquisición del conjunto de datos

Característica	Especificación
Alcance Máximo	10 kilómetros
Altura Máxima de Vuelo	100 metros
Tiempo máximo de vuelo	30 minutos
Velocidad máxima de vuelo	50 km/h
Resolución de imagen	4000 x 2250 pixeles
Tamaño de imagen	Aproximadamente 4 MB
Cámara	Estabilizada
Altura de vuelo	30 a 50 metros

Nuestro conjunto de datos consta de 1546 imágenes, tomadas a una altura que oscila entre 30 y 50 metros. Dentro de estas imágenes, hay entre 35 y 45 vacas, como se ilustra en la figura 3, que muestran una gran variedad de ganado y entornos. Las imágenes captan escenas que van desde exuberantes praderas verdes a áridos prados, mostrando vacas en diversas etapas de crecimiento y, a veces, sin animales presentes. Esta rica diversidad permite al modelo desarrollar una sólida comprensión tanto de la presencia como de la ausencia de vacas, mejorando su capacidad de discernimiento y reforzando su precisión.

Figura 3

Imagen ilustrativa del conjunto de datos, captada a 30 metros de altura

1.2 Etiquetado de imágenes

El etiquetado de datos es fundamental en un proyecto de aprendizaje automático, ya que proporciona la base sobre la que el modelo aprenderá y hará predicciones. El etiquetado de datos consiste en asignar etiquetas o categorías a cada instancia de los datos, lo que permite al modelo comprender y aprender patrones y relaciones en los datos.

Theos (1) es una plataforma en línea que ofrece una herramienta de anotación que agiliza el proceso de anotación de imágenes para el entrenamiento de modelos de aprendizaje automático, como se muestra en la Figura 4. Mediante el uso de Theos, los usuarios pueden etiquetar manualmente las imágenes para que el modelo pueda aprender y hacer predicciones. Mediante

Theos, los usuarios pueden anotar manualmente sus conjuntos de datos, marcando y categorizando de forma precisa y eficiente los objetos de interés dentro de las imágenes.

Además de la anotación manual, Theos AI ofrece una función de "autoetiquetado" que acelera considerablemente el proceso de anotación. Esta función utiliza modelos preentrenados para identificar automáticamente objetos comunes en las imágenes y aplicar las etiquetas correspondientes. Esto ahorra tiempo y esfuerzo al usuario, especialmente cuando se trata de grandes conjuntos de datos.

En este proyecto, se utilizó la herramienta de anotación de Theos AI para anotar manualmente 800 imágenes. Estas imágenes se utilizaron para entrenar un modelo personalizado. Posteriormente, se utilizó la función de etiquetado automático de Theos AI, aprovechando el modelo entrenado, para anotar automáticamente las 746 imágenes restantes. Esta combinación de anotación manual y etiquetado automático permitió obtener un conjunto de datos exhaustivo y preciso, crucial para el desarrollo y la evaluación del modelo de detección de objetos utilizado en el proyecto.

Figura 4

Interfaz de la plataforma Theos

Nota. Adaptado de Theos platform

1.3 Selección de modelos de detección de objetos

A lo largo de los años se han desarrollado importantes modelos en el campo de la detección de objetos. Cada modelo ha contribuido significativamente al avance de la inteligencia artificial y la visión por ordenador.

A continuación, presentamos una breve descripción de algunos modelos destacados:

1.3.1 R-CNN (Region-based Convolutional Neural Networks)

Introdujo el enfoque de detección basado en regiones en 2014.

1.3.2 Fast R-CNN

Mejora del tiempo de procesamiento al incorporar una única red convolucional en 2015.

1.3.3 Faster R-CNN

Introdujo el uso de una red neuronal convolucional para generar propuestas de regiones en 2015.

1.3.4 YOLO (You Only Look Once)

Propuso un enfoque de detección de objetos en tiempo real en 2015.

1.3.5 SSD (Single Shot MultiBox Detector)

Similar a YOLO, pero con mejoras en precisión y eficiencia en 2016.

1.3.6 RetinaNet

Introdujo la estructura de detección de objetos conocida como Feature Pyramid Network en 2017.

1.3.7 Mask R-CNN

Una extensión de Faster R-CNN que añade segmentación semántica en 2017.

You Only Look Once (YOLO)(10) es un popular algoritmo de detección de objetos conocido por su rapidez y precisión. Funciona dividiendo la imagen en una cuadrícula y prediciendo cuadros delimitadores y probabilidades de clase para cada celda de la cuadrícula. El modelo YOLO original,

introducido en 2015, revolucionó la detección de objetos al proporcionar capacidades de inferencia en tiempo real.

En los últimos años, YOLO ha experimentado varios cambios, y cada versión se ha basado en los éxitos y ha abordado las limitaciones de su predecesora.

La selección de YOLOv8 se basa en el hecho de que representa la versión más actualizada de YOLO, con una combinación de precisión y velocidad que la hace ideal para nuestro propósito. Además, su solidez y amplia validación en diversos escenarios, respaldada por una comunidad activa y recursos de aprendizaje fácilmente disponibles, consolidan a YOLOv8 como la opción más sólida y fiable para nuestro estudio.

Tabla 2

Comparación de los modelos YOLOv8

Modelo	mAPval	Velocidad CPU ONNX (ms)	Velocidad A100 Ten-sorRT (ms)	Parámetros (M)
YOLOv8n	37.3	80.4	0.99	3.2
YOLOv8s	44.9	128.4	1.20	11.2
YOLOv8m	50.2	234.7	1.83	25.9
YOLOv8l	52.9	375.2	2.39	43.7
YOLOv8x	53.9	479.1	3.53	68.2

La Tabla 2 presenta una comparación de varios modelos YOLOv8 basada en su tamaño, precisión media (mAPval), velocidad de inferencia en CPU utilizando ONNX (ms), velocidad de inferencia en A100 utilizando TensorRT (ms) y número de parámetros (M). Estos modelos, YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l y YOLOv8x, ofrecen diferentes compensaciones entre el tamaño del modelo, la velocidad y el rendimiento.

YOLOv8n, a pesar de su mAPval ligeramente inferior en comparación con otros modelos como YOLOv8s y YOLOv8m, ofrece un rendimiento respetable con un mAPval de 37,3. Destaca por ser el modelo más ligero y rápido de la lista, con sólo 3,2 millones de parámetros y un tiempo de procesamiento de 0,99 ms en A100 TensorRT. Esta característica lo convierte en una opción ideal para aplicaciones en las que la eficiencia y la velocidad de cálculo son cruciales.

Por otro lado, YOLOv8m destaca con un mAPval significativamente mejor que YOLOv8n, alcanzando un valor de 50,2, lo que indica una mayor precisión en la detección de objetos. Aunque tiene más parámetros que YOLOv8n, con 25,9 millones, sigue siendo un modelo de tamaño moderado en comparación con YOLOv8l y YOLOv8x. Además, con tiempos de procesamiento de 1,83 ms en A100 TensorRT, YOLOv8m consigue mantener una eficiencia considerable al tiempo que mejora la precisión en la detección de objetos.

1.4 Formación de los modelos

La selección de los modelos YOLOv8n y YOLOv8m se basó en la consideración de las necesidades específicas del caso. Optamos por estos modelos más ligeros y eficientes debido a su capacidad para proporcionar un equilibrio óptimo entre rendimiento y eficiencia computacional.

El proceso de entrenamiento fue el siguiente:

- Preparación:
  - **Descarga del conjunto de datos:** El conjunto de datos de imágenes de ganado se descargó de la nube y se descomprimió dentro del espacio de trabajo de Colab.
  - **Instalación de la biblioteca:** Se instaló la librería Ultralytics, imprescindible para el funcionamiento de YOLO.
  - **Importación del modelo:** Tras evaluar las limitaciones de hardware y los tiempos de procesamiento, se importaron dos modelos en entornos separados, YOLOv8n y YOLOv8m, ambas versiones ligeras y rápidas de YOLO preentrenadas en el conjunto

de datos COCO. Este preentrenamiento proporciona una base sólida para el aprendizaje posterior con nuestro conjunto de datos específico.

- Formación:
  - **Imágenes:** Ambos modelos se entrenaron con 1100 imágenes (datos de entrenamiento), que representan el 70% del conjunto total de datos, se validaron con 302 imágenes (datos de validación: 20%) y se probaron con 144 imágenes (datos de prueba: 10%).
  - **Detención anticipada:** La parada anticipada se aplicó con un ajuste de paciencia de 5 para evitar el sobreajuste del modelo y mejorar su capacidad de generalización. La parada temprana es una técnica utilizada habitualmente durante el entrenamiento de los modelos de aprendizaje automático para controlar el rendimiento del modelo en un conjunto de datos de validación. Funciona deteniendo el proceso de entrenamiento si el rendimiento del modelo en el conjunto de validación deja de mejorar, incluso si la pérdida de entrenamiento sigue disminuyendo.
  - **Optimizador:** El optimizador utilizado en este caso es AdamW, elegido automáticamente por el modelo YOLOv8, una variante del optimizador Adam que incluye el decaimiento del peso, lo que ayuda a evitar el sobreajuste penalizando los pesos grandes en la función de pérdida y mejorando la generalización del modelo. Tiene una tasa de aprendizaje de 0,002, que determina la magnitud de los ajustes realizados en los pesos del modelo durante el entrenamiento, y un impulso de 0,9, un parámetro que suaviza la actualización de los pesos del modelo teniendo en cuenta la dirección y la velocidad de los cambios anteriores en los gradientes.
  - **Épocas:** Para que el modelo tuviera tiempo suficiente para aprender los patrones y características presentes en los objetos de interés de las imágenes, se seleccionaron 100 épocas, utilizando la técnica de parada temprana antes mencionada.

- Selección del modelo final:

Tras el análisis, se observó que el modelo YOLOv8n mostraba una mejora mínima en las épocas finales. Por el contrario, el modelo YOLOv8m demostró sistemáticamente un rendimiento superior, lo que indica su eficacia con respecto al modelo YOLOv8n.

Basándose en estos resultados, **YOLOv8m** fue seleccionado como el mejor modelo para el proyecto.

- Aumento de datos con albumentaciones:

El aumento de datos desempeña un papel crucial en la mejora de la diversidad y la solidez del modelo YOLO. En este proyecto, se empleó la biblioteca "augmentation" para aplicar diversas transformaciones a las imágenes de entrenamiento. Se utilizaron las siguientes transformaciones:

- **Desenfoque gaussiano:** esta transformación desenfoca la imagen mediante un filtro gaussiano, lo que ayuda a reducir el ruido y mejora la capacidad de generalización del modelo.
- **Desenfoque de mediana**: la aplicación de un filtro de mediana a la imagen ayuda a eliminar el ruido al tiempo que preserva los bordes de los objetos.
- **Conversión a escala de grises:** La conversión de la imagen a escala de grises permite al modelo centrarse en la forma y la textura de los objetos de la imagen.
- **Ecualización adaptativa del histograma con limitación de contraste (CLAHE):** CLAHE mejora el contraste local de la imagen, lo que puede ser beneficioso para resaltar detalles importantes.

1.5 Validación cruzada estratificada por pliegues k

Para garantizar una buena evaluación del rendimiento del modelo, empleamos el procedimiento de validación cruzada estratificada de k pliegues. Este método es especialmente

beneficioso cuando se trabaja con conjuntos de datos desequilibrados, ya que preserva la distribución de las clases objetivo dentro de cada pliegue.

El procedimiento es el siguiente: dividimos nuestro conjunto de datos de entrenamiento en 5 partes iguales, denominadas k=5 pliegues. A diferencia de la validación cruzada k-fold normal, que divide aleatoriamente el conjunto de datos, la versión estratificada garantiza que cada pliegue mantenga la misma proporción de clases objetivo que el conjunto de datos original.

A continuación, el modelo se entrenó 5 veces, utilizando cada vez un pliegue diferente como conjunto de validación, mientras que los k-1 pliegues restantes se utilizaron para el entrenamiento. Este enfoque garantiza que el modelo se pruebe en un conjunto de datos diverso, manteniendo al mismo tiempo una representación equilibrada de las clases objetivo.

A lo largo de cada iteración de entrenamiento y validación, registramos cuatro métricas: exactitud, precisión, recuperación y puntuación f1. Estas métricas pueden consultarse en la sección de resultados.

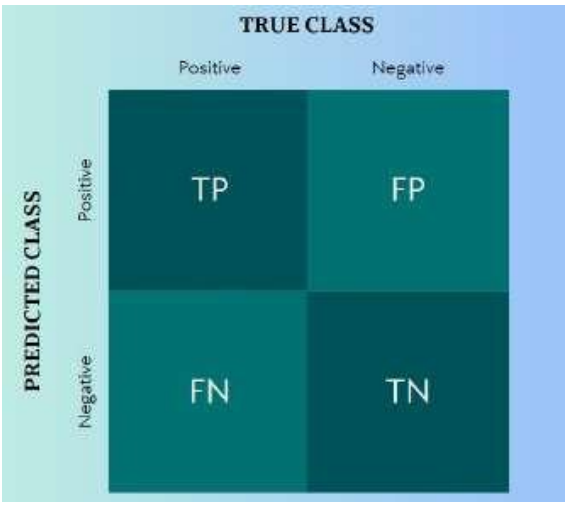
1.6 Evaluación del rendimiento

Tras entrenar el modelo con datos de prueba para evaluar su rendimiento, se aplicaron las siguientes métricas ((11)):

Matriz de confusión La matriz de confusión, como se muestra en la Figura 5, resume el rendimiento del modelo comparando sus predicciones con las etiquetas de la verdad sobre el terreno. Cada celda representa una combinación de predicción y etiqueta:

Figura 5

Imagen ilustrativa de la matriz de confusión



Los verdaderos positivos (TP) representan las predicciones correctas de la clase objetivo, cuando el modelo detecta y clasifica con precisión los objetos de interés. Los falsos positivos (FP) se producen cuando el modelo identifica incorrectamente objetos que no pertenecen a la clase objetivo como pertenecientes a ella. Por otro lado, los Verdaderos Negativos (NT) son casos en los que el modelo identifica correctamente objetos que no pertenecen a la clase objetivo como tales. Por último, los Falsos Negativos (FN) se producen cuando el modelo no detecta objetos de la clase objetivo, clasificándolos erróneamente como no pertenecientes a la misma.

La matriz de confusión permite visualizar el comportamiento del modelo para cada clase de objeto. A partir de ella se pueden calcular otras métricas, como Precisión, Recall y F1-score.

Acierto Esta métrica se calcula como la relación entre el número de predicciones correctas y el número total de predicciones:

$$\text{Acierto} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (1)$$

Esta métrica es sencilla y fácil de interpretar, ya que representa la proporción de muestras clasificadas correctamente. Sin embargo, en casos de desequilibrio de clases, otras métricas como

la precisión, la recuperación y la puntuación F1 pueden ofrecer una evaluación más detallada del rendimiento del modelo.

Precisión Esta métrica indica la proporción de predicciones positivas que son correctas, es decir, el número de TP dividido por la suma de TP y FP:

$$\text{Precisión} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

Un valor alto de Precisión indica que el modelo tiene un número bajo de FP.

La precisión refleja la fiabilidad del modelo en sus predicciones positivas. Un valor alto indica que el modelo es preciso en la identificación de objetos de la clase objetivo.

Recall (Sensibilidad) Esta métrica indica la proporción de objetos de la clase objetivo que se detectan correctamente, es decir, el número de TP dividido por la suma de TP y FN:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

Un valor Recall alto indica que el modelo tiene un número bajo de FN.

La recuperación refleja la capacidad del modelo para detectar todos los objetos de la clase objetivo. Un valor alto indica que el modelo es eficaz en la detección de objetos de la clase objetivo.

Puntuación F1 Esta métrica es la media armónica entre Precisión y Recuperación, y proporciona una medida equilibrada del rendimiento del modelo:

$$\text{Puntuación F1} = \frac{\text{Precisión x Recall}}{\text{Precisión} + \text{Recall}} \tag{4}$$

Una puntuación F1 alta indica un buen equilibrio entre la detección precisa de objetos y la minimización de errores.

La puntuación F1 es una métrica equilibrada que tiene en cuenta tanto la precisión como la recuperación. Una puntuación F1 alta indica un buen equilibrio entre la detección precisa de objetos y la minimización de errores.

Intersección sobre Unión (IoU) es una métrica ampliamente utilizada en tareas de detección de objetos para evaluar la precisión de las detecciones. Se calcula como el área de intersección entre el cuadro delimitador previsto y la verdad sobre el terreno, dividida por el área de unión de ambos cuadros. La fórmula de IoU se expresa como

$$\text{IoU} = \frac{\text{Área de intersección}}{\text{Área de Unión}} \tag{5}$$

Un valor alto de IoU, normalmente cercano a 1, indica un solapamiento significativo entre la predicción y la verdad sobre el terreno, lo que se considera una detección precisa. Por el contrario, un valor de IoU bajo, cercano a 0, indica un solapamiento escaso o nulo entre las detecciones, lo que indica errores en la predicción.

Precisión media (mAP) Esta métrica resume la precisión del modelo en diferentes niveles de IoU calculando la media de precisión a lo largo de una curva Precisión-Recuperación. Un valor alto de mAP indica un buen rendimiento general del modelo en la detección de objetos.

mAP es una métrica global que resume el rendimiento del modelo en diferentes niveles de IoU. Un valor alto de mAP indica que el modelo tiene un buen rendimiento general en la detección de objetos.

1.7 Desarrollo de aplicaciones móviles

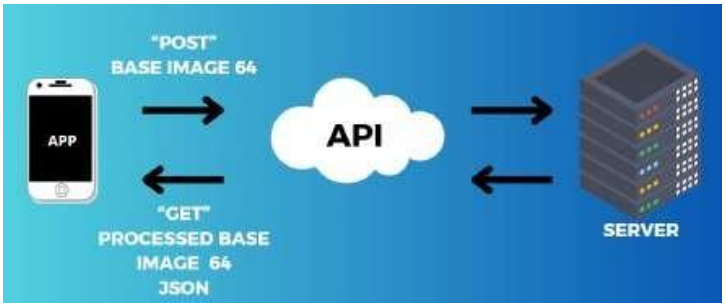
El proceso de desarrollo de aplicaciones móviles se estructura en diferentes fases que, en conjunto, crean una aplicación capaz de:

- Capturar una imagen seleccionada por el usuario.
- Procesar la imagen mediante un modelo de inteligencia artificial.
- Presentar la imagen junto con el número de detecciones realizadas.
- Mostrar la imagen resultante junto con su respectiva detección.

Experiencia de usuario (UX) En este caso, se adopta un enfoque centrado en la simplicidad para la aplicación con el objetivo de hacerla accesible a cualquier usuario. Esto se consigue mediante un flujo de usuario minimalista y sencillo.

Figura 6

Diagrama de flujo de usuarios



Como se ve en la figura 6, el diagrama de flujo de la aplicación sigue una secuencia lineal: el usuario accede a la aplicación, elige la imagen que quiere procesar de su galería o de Google Fotos, y la envía al servidor mediante el "botón" Subir imagen". La imagen se almacena en una carpeta local hasta que está lista para ser procesada. Al pulsar el botón "Procesar imagen", la imagen procesada se recupera de la API y se muestra en la pantalla. Si el usuario desea procesar otra imagen, puede volver a la opción "Seleccionar imagen" y reiniciar el flujo. En caso de que el usuario salga de la aplicación, se han previsto varias situaciones que podrían provocar errores:

- Si el usuario pulsa "Subir imagen" sin seleccionar ninguna imagen: No se enviará ninguna imagen al Servidor, por lo que la API no tendrá ninguna imagen que procesar y su respuesta estará vacía.
- Si el usuario pulsa "Procesar imagen" sin seleccionar ninguna imagen: La última imagen procesada será recuperada de la API y mostrada en la aplicación.

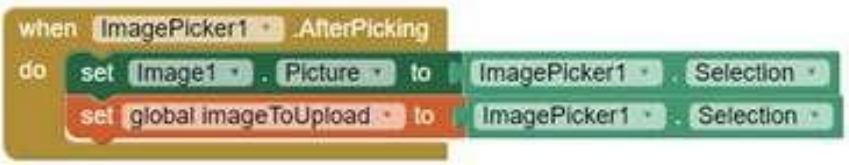
  Ninguno de los errores mencionados impedirá el funcionamiento de la aplicación.

Etapa de diseño de la aplicación móvil Para este proceso se utilizó la herramienta de desarrollo de aplicaciones móviles App Inventor. Mediante el uso de programación por bloques, se desarrollan los siguientes procesos:

- Selección de Imagen: Utilizando el componente ImagePicker, se ha creado un botón que permite al usuario acceder a su galería y a Google Imágenes para seleccionar la imagen que desea procesar. Esta imagen se muestra en pantalla mediante un elemento "Image" y se guarda en una variable global llamada "ImageToUpload".

Figura 7

Bloques de selección de imágenes

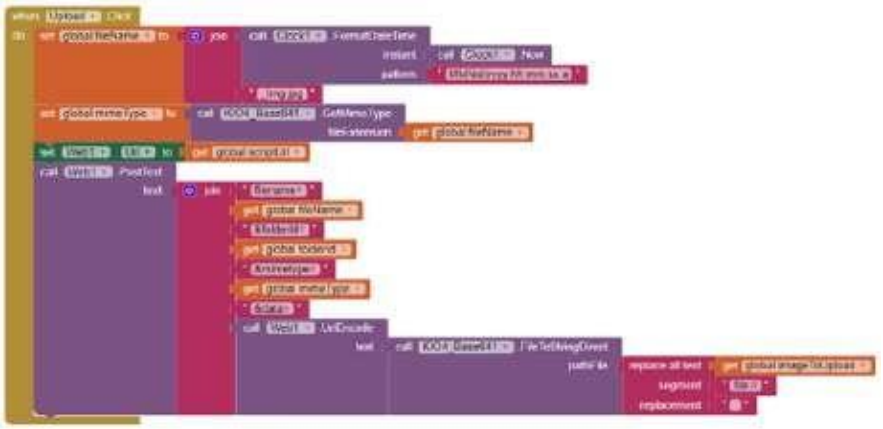


Nota. Adaptado de App inventor

- Cargar imagen: Se implementa un botón que, al seleccionarlo, convierte la imagen almacenada en la variable local "ImageToUpload" en una cadena de texto en formato Base64. Esta cadena se envía mediante el método HTTP POST a una API desarrollada en Python, que permite tomar el fichero en formato Base64, convertirlo de nuevo en imagen y almacenarlo en una carpeta local.

Figura 8

Bloques de carga de imágenes

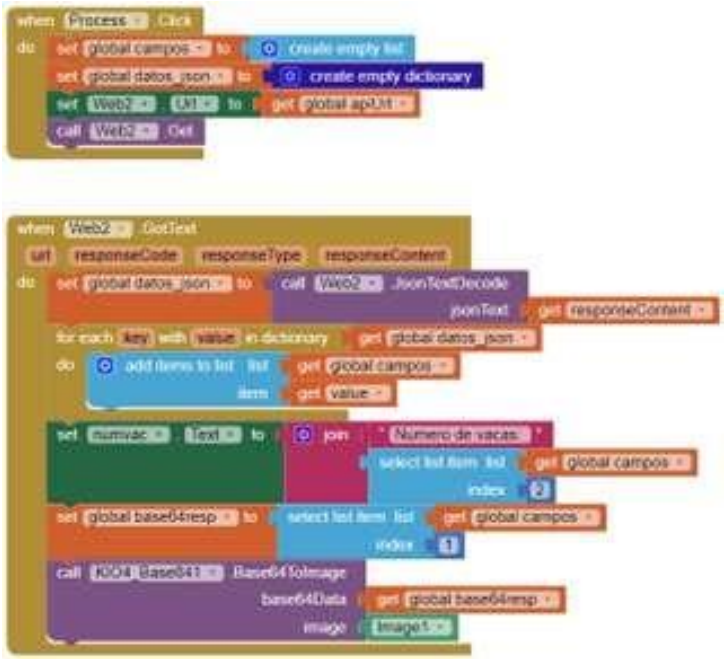


Nota. Adaptado de App inventor

- Descargar la Imagen: Se ha implementado un botón denominado "Procesar Imagen" que, a través del método HTTP GET, obtiene un objeto JSON de la API diseñada. Este objeto JSON se transforma en un diccionario que contiene dos elementos: el número de vacas contadas y la imagen procesada por el modelo en formato Base64. Mediante un elemento Label, se muestra en pantalla el número de vacas encontradas. Además, el elemento Base64 se convierte de nuevo en una imagen que se muestra en la pantalla mediante un elemento apropiado.

Figura 9

Bloques de descarga de imágenes



Nota. Adaptado de App inventor

Creación de la API Esta etapa es crucial para garantizar el rendimiento óptimo de la aplicación. Se ha seleccionado una API REST para su implementación debido a que proporciona una interfaz estandarizada, permitiendo el intercambio seguro de información entre dos sistemas informáticos a través del protocolo de comunicación HTTP. El desarrollo de esta API se ha llevado a cabo utilizando el framework Flask y el lenguaje de programación Python, siguiendo el enfoque de programación descrito.

- Entorno virtual: Se crea un entorno virtual para aislar y gestionar de forma independiente las dependencias y configuraciones del proyecto a través de la herramienta virtualenv. Este entorno cuenta con python 3 y diferentes dependencias que se pueden ver en el archivo requirements.txt entre ellas las más relevantes: flask (framework para desarrollar laAPI),

ultralytics (Herramienta para poder utilizar el modelo de IA). Con un entorno definido con las herramientas a utilizar, procedemos a empezar a programar.

- Index.py: Este código Python implementa un servicio web utilizando Flask. La aplicación define una ruta raíz mediante la decoración @app.route('/'), lo que significa que responde a peticiones HTTP GET y POST dirigidas a la raíz del servidor web. Dentro de esta ruta, el servidor realiza una serie de acciones: Primero, carga una imagen local desde la ruta especificada en image path. A continuación, carga un modelo YOLOv8 preentrenado utilizando la biblioteca Ultralytics. A continuación, ejecuta inferencias de detección de objetos en la imagen cargada utilizando este modelo. Una vez completadas las inferencias, genera una nueva imagen con las detecciones iluminadas. El siguiente paso es convertir esta imagen al formato base64, que facilita su representación y transferencia a través de la web. Utilizando la imagen convertida, construye un objeto JSON de respuesta que contiene la imagen codificada en base64 junto con el número de detecciones de objetos realizadas. Finalmente, este JSON se devuelve como respuesta a la petición HTTP.

2. Resultados

Conjunto de Datos

En la fase inicial de los resultados obtenidos en el desarrollo de este proyecto, destaca la con- strucción de la base de datos, que consta de un conjunto total de 1546 imágenes etiquetadas. Un ejemplo de estas imágenes se muestra en la Figura 10. En cada imagen se etiquetó la ubicación de cada vaca. Es importante mencionar que esta base de datos fue diseñada para ser compatible con la arquitectura YOLO.

Figura 10

Imagen etiquetada

Nota. Adaptado de Theos (1)

Las etiquetas de cada una de las imágenes se presentan en formato de archivo TXT, como se ilustra en la figura 11. Estas etiquetas contienen tanto la clase como las coordenadas de cada objeto dentro de la imagen. Es importante señalar que cada línea representa un objeto de la imagen. La estructura de las etiquetas es la siguiente:

- El primer número representa la clase del objeto. En este caso, como sólo hay un objeto, el primer número de todas las etiquetas es "0".
- El segundo número indica la posición en el eje X del centro del objeto, normalizado con respecto a la anchura de la imagen.
- El tercer número indica la posición en el eje Y del centro del objeto, normalizada con respecto a la altura de la imagen.
- El cuarto número indica la anchura del objeto, también normalizada con respecto a la imagen.
- El quinto número indica la altura del objeto, normalizada con respecto a la altura de la imagen

Figura 11

Estructura de la etiqueta

```
0 0.447625 0.45066666666666666 0.03875 0.035333333333333335
0 0.3225 0.429 0.0205 0.058666666666666666
0 0.42925 0.21966666666666668 0.02 0.030666666666666665
```

La base de datos se estructuró de la siguiente manera:

- 1100 imágenes de entrenamiento (71 %).
- 302 imágenes de validación (20 %).
- 144 imágenes de prueba (9 %)..

Formación de modelos

2.1 Validación del modelo durante el entrenamiento

El modelo se entrenó utilizando dos algoritmos YOLO diferentes, en este caso, YOLOv8m y YOLOv8n. Tras el entrenamiento, YOLO proporciona algunas métricas de rendimiento, entre las que destaca la precisión media promedio (mAP), que alcanza 0,993 en ambos modelos. Por lo tanto, se analizó la matriz de confusión de cada modelo. Éstas se presentan en la Figura 12, mostrando el número de vacas que fueron correctamente clasificadas, así como los falsos positivos, falsos negativos y verdaderos positivos. Cabe señalar que, en este caso, no hay verdaderos negativos, ya que el modelo sólo detecta si se trata de una vaca o no. Por lo tanto, si no es una vaca, no hay otra clase definida que pueda contener verdaderos negativos.

Figura 12

Matriz de confusión para los datos de validación

En este caso, la matriz de confusión se construye utilizando datos de validación, que comprenden un total de 3998 instancias que se pretende clasificar correctamente. A partir de esta matriz se obtienen varias métricas de rendimiento del modelo, como la exactitud, la precisión, la recuperación y la métrica F1. Los resultados de estas métricas se resumen en la Tabla 3, que ofrece una visión general del rendimiento alcanzado por cada modelo evaluado.

Es importante destacar que, en esta aplicación específica, se busca una mayor precisión del modelo. Sin embargo, existe una notable superioridad en todas las métricas de rendimiento para el modelo YOLOv8m. En consecuencia, se selecciona este modelo como el preferido para su aplicación.

Tabla 3

Métricas de rendimiento de los datos de validación

Parámetro	YOLOv8m	YOLOv8n
Exactitud	0.951	0.928
Precisión	0.961	0.941
Recall	0.989	0.985
F1	0.974	0.962

2.2 Validación cruzada k-Fold

Una vez elegido el modelo, se realiza una validación cruzada estratificada k-fold para evaluar el modelo, como se observa en la Tabla 4.

Tabla 4

Resultados de la validación cruzada k-Fold

Parámetro	Media ± Desviación Estándar
Exactitud	0.9400 ± 0.0116
Precisión	0.9523 ± 0.0120
Recall	0.9864 ± 0.0019
F1	0.9690 ± 0.0062

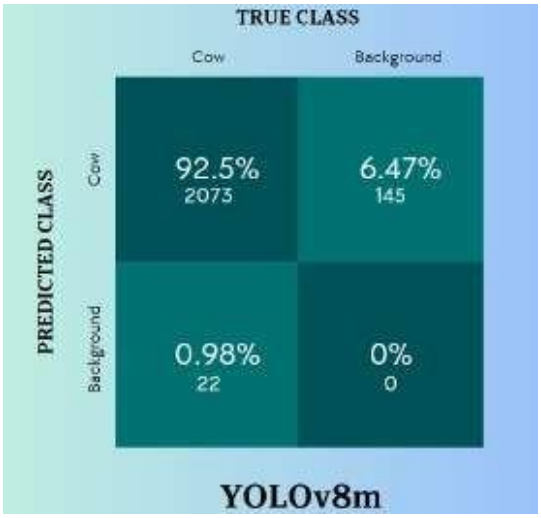
Los datos presentados en la Tabla 4 revelan desviaciones estándar mínimas en las métricas Precisión, Precisión, Recuperación y Puntuación F1. Esta coherencia sugiere que el modelo funciona de forma fiable en varios pliegues, lo que refleja su rendimiento estable.

2.3 Métricas de rendimiento para datos de prueba

Con la confianza puesta en que el modelo YOLO V8M presentaba mejoras significativas en la fase de validación, la evaluación del rendimiento se llevó a cabo utilizando datos de prueba. Una vez realizadas las predicciones, se obtuvieron los siguientes resultados:

Figura 13

Matriz de confusión de los datos de prueba



Como se observa en la Figura 13, se obtienen resultados próximos a los obtenidos con los datos de validación. Para un análisis más detallado de las métricas de rendimiento de este modelo en los datos de prueba, se presenta la siguiente tabla:

Tabla 5

Métricas de rendimiento para datos de prueba

Parámetro	YOLOv8m
Exactitud	0.925
Precisión	0.934
Recall	0.989
F1	0.9607

Con los datos obtenidos en la tabla 5, se aprecia que el modelo también consigue buenos resultados para los datos de prueba, tal y como se esperaba. Esto sugiere una alta capacidad de gener- alización y validez del modelo con nuevos datos fuera del conjunto de entrenamiento.

Con un modelo definido, se prueba el modelo con imágenes que simulan una aplicación del mundo real en la que es necesario poner a prueba el modelo. Implementando esto a través de Colab, en la Figura 14 se puede ver un ejemplo de cómo se obtiene el resultado, incluyendo la identificación y el recuento de las vacas.

Figura 14

Imagen del resultado con las predicciones

2.4 Consideraciones sobre la eficacia del modelo

A su vez, al analizar los resultados obtenidos, se identificaron algunos casos que podrían comprometer la eficacia del modelo. Entre ellos, destacan los siguientes:

- Existen obstáculos en la imagen, como árboles u objetos que pueden interferir en la captura fotográfica. Por este motivo, es fundamental tener en cuenta el entorno en el que se despliega el modelo.
- El número de vacas en la imagen también representa un factor crítico. El conjunto de datos utilizado incluye fotografías con una media máxima de 35 a 45 vacas. Cuando el modelo se enfrenta a imágenes con un número sustancialmente mayor de vacas (por ejemplo, 100

o 200), puede experimentar imprecisiones, ya que no ha sido entrenado con imágenes que presenten ese número específico.

- La presencia de animales distintos de las vacas en la imagen constituye otro reto. El modelo tiende a identificarlos como vacas, ya que su entrenamiento se centró únicamente en la detección de este tipo de ganado. Para abordar esta cuestión, sería necesario entrenar el modelo para que reconozca también otros tipos de animales. Sin embargo, esta adaptación podría aplicarse específicamente a las explotaciones que presentan esta variabilidad.

Aplicación Móvil

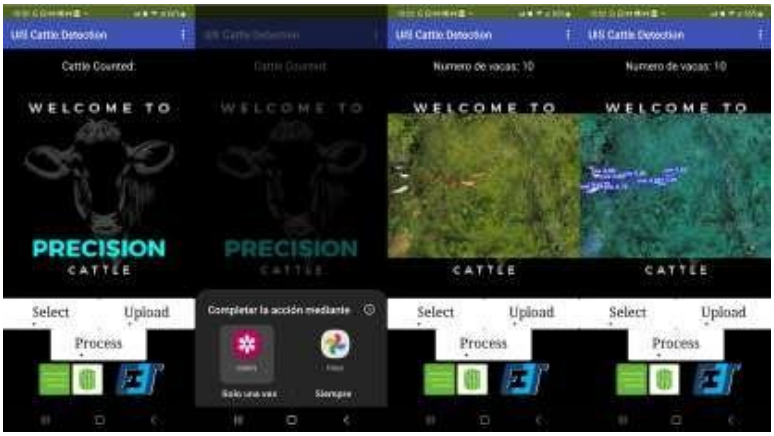
La implementación del servidor web Flask para la detección de objetos ha sido un éxito. Al acceder a la ruta raíz del servidor, la imagen local se cargó correctamente, y el modelo YOLOv8m preentrenado realizó una inferencia precisa de detección de objetos. Se contabilizó el número de objetos detectados, en este caso vacas, y se generó una imagen anotada en la que se destacaban las detecciones. Además, la conversión de la imagen anotada a base64 se completó sin problemas, permitiendo su transmisión como parte de la respuesta JSON, proporcionando una salida completa que es recuperada por la aplicación construida con App Inventor. Aunque la imagen procesada puede mostrar un tono diferente al original debido a la conversión a base64 y viceversa, el recuento y la identificación del ganado no se vieron afectados. En conjunto, estos resultados, representados en la Figura 10 y en el repositorio que aloja todo lo relacionado con la aplicación, validan la eficacia e integridad del servidor Flask implementado para la detección de objetos.

Aunque los resultados obtenidos tanto por la aplicación como por la API son precisos, es importante señalar que la API se ha implementado en un entorno ubicado en un servidor local, concretamente en el ordenador donde se ha desarrollado el código. Esta consideración es crucial,

ya que la arquitectura de comunicación cliente-servidor empleada por la aplicación implica que la disponibilidad de la aplicación está directamente ligada al estado del servidor, es decir, a la disponibilidad y funcionalidad del ordenador que aloja el servidor. Por lo tanto, es necesario asegurarse de que el servidor está encendido y operativo para que la aplicación sea accesible y pueda prestar sus servicios eficazmente.

Figura 15

Aplicación móvil en funcionamiento



3. Conclusiones

Se desarrolló un modelo de aprendizaje automático basado en YOLOv8 que se entrenó con una base de datos de imágenes de ganado. El modelo alcanzó una exactitud del 94%, una precisión del 95%, un recall del 98% y una puntuación F1 del 96%, demostrando su eficacia para la tarea de recuento de ganado.

Este trabajo tiene un gran potencial para contribuir a la gestión del ganado de forma autónoma, facilitando y economizando la seguridad y el control del ganado para los ganaderos. Se diferencia de investigaciones anteriores al utilizar tecnología moderna como drones y YOLOv8, obteniendo resultados más precisos.

La validación del enfoque metodológico ha demostrado su eficacia. Los resultados obtenidos son coherentes con las expectativas teóricas y las hipótesis planteadas.

Las conclusiones de este trabajo pueden aplicarse fácilmente a explotaciones ganaderas que busquen automatizar procesos. El modelo puede ser entrenado con imágenes específicas de la explotación para una mayor precisión. Este trabajo puede tener un impacto significativo en la industria ganadera, en la sociedad y en futuras investigaciones.

Sin embargo, el proyecto también tiene algunas limitaciones. Las condiciones meteorológicas adversas pueden afectar al vuelo del dron y a la calidad de las imágenes. Además, el modelo se entrenó con imágenes con no más de 50 animales en el rebaño, por lo que su rendimiento podría verse afectado en rebaños más grandes.

Para futuras investigaciones, se recomienda explorar la aplicación del modelo en los llanos orientales colombianos o en la región de la Orinoquía, donde se encuentra la mayor región

ganadera y los hatos son más grandes. También se recomienda mejorar la aplicación móvil para almacenar datos, georreferenciarlos y realizar análisis para una mejor gestión ganadera.

En general, la experiencia de desarrollar este proyecto ha sido enriquecedora y nos ha permitido conocer las necesidades de los ganaderos y el funcionamiento del modelo YOLOv8. Los conocimientos y habilidades adquiridos servirán para futuros proyectos.

Referencias Bibliográficas

[1] T. AI. (2022) Get started with object detection. [Online]. Available: https://docs.theos.ai/get-started/object-detection

[2] DJI. (2024) Dji mini 2 - product support. [Online]. Available: https://www.dji.com/global/support/product/mini-2

[3] WRadio, "Las pérdidas que quedaron en 2022 para los ganaderos por el robo de bovinos," https://www.wradio.com.co/, 2023, [Consultado el 21 de marzo de 2024].

[4] W. Andrew, C. Greatwood, and T. Burghardt, "Visual localisation and individual identification of holstein friesian cattle via deep learning," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, 2017, pp. 22–29.

[5] I. A. Puetaman and C. M. Báez, "Prototipo de sistema de vigilancia para fincas ganaderas como prevención al abigeato," *Rev. UNIMAR*, vol. 32, no. 1, pp. 67–81, 2014.

[6] W. Vergara, "La ganadería extensiva y el problema agrario. el reto de un modelo de desarrollo rural sustentable para colombia," *Rev. Cienc. Anim.*, vol. 3, pp. 45–53, 2010.

[7] O. Starn, *Reflexiones sobre rondas campesinas, protesta rural y nuevos movimientos sociales*, Lima, 1991.

[8] P. Chamoso, W. Raveane, V. Parra, and A. González, "Uavs applied to the counting and monitoring of animals," in *Ambient Intelligence-Software and Applications*. Springer, 2014, pp. 71–80.

[9] V. Berovides, M. Cañizares, and A. Gonzalez, “Metodos de conteo de animales y plantasterrestres,” pp. 1–15, 2005.

[10] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, “A review of yolo algorithm developments,” *Procedia computer science*, vol. 199, pp. 1066–1073, 2022.

[11] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. "O’Reilly Media, Inc.", 2022.