Implementation of Drones in Precision Livestock Farming for Animal Counting and Tracking

Sergio Andrés Angarita Camacho[a], Andrés Felipe Araque Guerrero[b], Jhonathan Alexander Murcia Galán[c]
[a,b,c]Universidad Industrial de Santander, Escuela E3T, Bucaramanga, Colombia
[a]Sergio, SA, Angarita; sergioa.angarita@gmail.com
[b]Andrés, AF, Araque; andresfearaque28@gmail.com
[c]Jhonathan, JA, Murcia; ing.jamg23@gmail.com

***Abstract*—In Colombia, livestock farming represents an important source of income, contributing 27% to agricultural Gross Domestic Product. However, this activity faces various challenges 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, collecting 1546 aerial images of livestock in 4 locations in Santander and Boyacá, Colombia. 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.**

Keywords: Aerial surveillance, cattle rustling, drone, database, livestock, YoloV8

I. Introduction

Cattle theft, technically known as "rustling", poses a significant challenge to the livestock industry in Colombia. According to information provided by the Ministry of Agriculture, livestock farming prevails in 27 out of the country's 32 departments. In 2022, over 1350 cattle were reported stolen, impacting the economy and security of livestock farmers. WRadio (2023)

In relation to this problem, various solutions have been applied, some of which involve the use of artificial intelligence. Andrew et al. (2017) presented a system utilizing images captured by drones and deep learning computer vision for the identification and localization of a specific type of cattle (Holstein Friesian), achieving precision levels of 99.3% Andrew, Greatwood, and Burghardt (2017). This work's main contribution lies in not only detecting cattle but also individualizing animals. Puetaman et al. (2014) proposed a prototype device based on XBee technology placed on cattle's necks to determine their life status by monitoring temperature. Additionally, this device can identify if the animal is within the permitted grazing area, emitting a radiofrequency signal transmitted from the device to a central system Puetaman and Báez (2014).

On the other hand, in Colombia, the agricultural sector is characterized by a relatively low level of technological adoption Vergara (2010). In the absence of technology, more traditional solutions are employed in many parts of the country, such as patrols, rural guards, protests, and, in extreme cases, the use of firearms in self-defense against theft Starn (1991). Although common, these measures are typically less effective compared to more advanced technologies developed for cattle control and supervision.

Chamoso et al. (2014) utilized aerial images captured by drones and convolutional neural networks for cattle counting on farms, achieving counting accuracies exceeding 97% Chamoso, Raveane, Parra, and González (2014). The use of artificial intelligence has emerged as a highly effective technological measure for precise cattle counting, significantly contributing to the fight against abigeato. Current cattle counting techniques in the beef industry are considered outdated and costly, primarily relying on three approaches: total counting, parcel counting, and marking and recounting Berovides, Cañizares, and Gonzalez (2005). To address this identified need, the utilization of an algorithm enabling cattle head counting through drone employment is proposed, encouraging livestock farmers to adopt this new technology for automated census purposes.

Given the challenge of finding a suitable database, we opted to create our own, tailored to the project's requirements.

II. Methods

Throughout the development of the project, we went through seven key phases. We began by acquiring the images for the database, followed by the task of image labeling. Subsequently, we conducted a search for models capable of detecting objects and selected the appropriate one. We evaluated its performance using the k-Fold cross-validation technique and concluded with the development of the mobile application, which allows for presenting the results in a user-friendly manner.

A. Acquisition

Colombia boasts several cattle farming regions. However, the towns that welcomed us are located in Santander and Boyacá. Specifically, Olival, Lebrija, Chiquinquirá, and Belén (Figure 1) were the places where we obtained permissions to fly around approximately 10 farms in various locations.

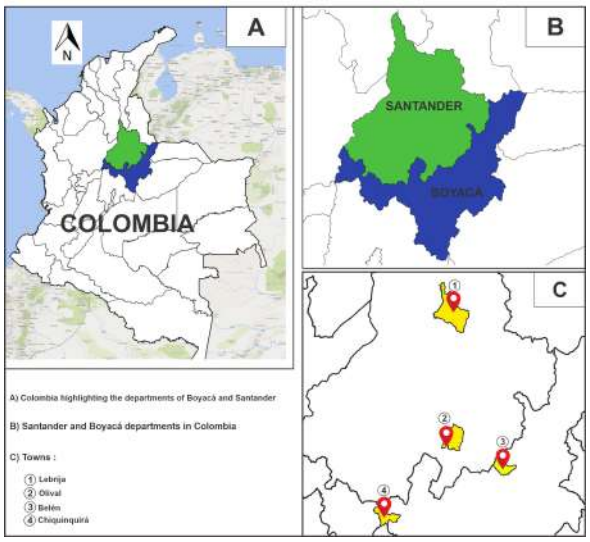

Figure 1. Colombian towns where the farms are situated, which allowed us to capture the photos for the dataset.

Source: Google Maps.

a) ***Map access:*** In the following link you can access the interactive map with the locations: Precision Livestock Project Locations

b) ***Drone:*** The DJI (2024) Mini 2 stands out as a lightweight and compact drone, designed with portability and user-friendliness in mind. Produced by DJI, a pioneer in drone technology, this model represents an enhanced iteration of its forerunner, the DJI Mavic Mini. With a mere weight of 249 grams, the Mini 2(Figure 2) remains exempt from registration requirements in select jurisdictions, rendering it an attractive choice for both novices and enthusiasts.

Figure 2. The DJI Mini 2, a lightweight and compact drone designed for portability and ease use.

Source: DJI Mini 2.

In spite of its diminutive size, the Mini 2 boasts an array of advanced flight and imaging capabilities. Equipped with a three-axis stabilizer, it ensures smooth and steady shots, even when faced with moderate wind conditions. The integrated camera impresses, capable of recording videos in stunning 4K resolution at 30 frames per second and capturing crisp 12-megapixel photos. For a comprehensive overview of the DJI Mini 2's specifications, refer to Table I. Table I provides key specifications of the DJI Mini 2 Drone used in the dataset acquisition.

Table I
DETAILS OF DJI (2024) MINI 2 , DRONE USED IN THE DATASET ACQUISITION.

Feature	Specification
Max Range	10 kilometers
Max Flight Height	100 meters
Max Flight Time	30 minutes
Max Flight Speed	50 km/h
Image Resolution	4000 x 2250 pixels
Image Size	Approximately 4 MB
Camera	Stabilized
Flight Height	30 to 50 meters

Our dataset consists of 1546 images, taken at a height ranging from 30 to 50 meters. Within these images, there is a range of 35 to 45 cows, as illustrated in Figure 3, showcasing a diverse array of livestock and settings. The images capture scenes ranging from lush green meadows to arid paddocks, depicting cows in various stages of growth and, at times, with no animals present. This rich diversity enables the model to develop a robust understanding of both the presence and absence of cows, enhancing its discernment capabilities and bolstering its precision.

Figure 3. Illustrative image of the data set, captured at a height of 30 meters.

Source: Self dataset image.

B. *Image labeling*

Data labeling is fundamental in a machine learning project, as it provides the foundation upon which the model will learn and make predictions. Data labeling involves assigning labels or categories to each instance of the data, allowing the model to understand and learn patterns and relationships in the data.

Theos AI (2022) is an online platform that offers an annotation tool, which streamlines the process of image annotation for training machine learning models showcased in Figure 4. By using Theos, users can manually annotate their datasets, precisely and efficiently marking and categorizing objects of interest within the images.

In addition to manual annotation, Theos AI also provides an "autolabel" feature, which significantly speeds up the annotation process. This feature utilizes pre-trained models to automatically identify common objects in the images and apply the corresponding labels. This saves time and effort for the user, especially when dealing with large datasets.

In the project at hand, Theos AI's annotation tool was used to initially annotate 800 images manually. These images were then used to train a custom model. Subsequently, the autolabel feature of Theos AI was employed, leveraging the trained model, to automatically annotate the remaining 746 images. This combination of manual annotation and autolabeling enabled the acquisition of a comprehensive and accurate dataset, crucial for the development and evaluation of the object detection model used in the project.

Figure 4. Theos platform interface.

Source: Theos platform.

C. Object Detection Model Selection

Over the years, prominent models have been developed in the field of object detection. Each model has significantly contributed to the advancement of artificial intelligence and computer vision.
Here we have a brief description of some standout models:

1. **R-CNN (Region-based Convolutional Neural Networks)**
   - Introduced the region-based detection approach in 2014.
2. **Fast R-CNN**
   - Improved processing time by incorporating a single convolutional network in 2015.
3. **Faster R-CNN**
   - Introduced the use of a convolutional neural network to generate region proposals in 2015.
4. **YOLO (You Only Look Once)**
   - Proposed a real-time object detection approach in 2015.
5. **SSD (Single Shot MultiBox Detector)**
   - Similar to YOLO, but with improvements in accuracy and efficiency in 2016.
6. **RetinaNet**
   - Introduced the object detection structure known as Feature Pyramid Network in 2017.
7. **Mask R-CNN**
   - An extension of Faster R-CNN that adds semantic segmentation in 2017.

You Only Look Once (YOLO)Jiang, Ergu, Liu, Cai, and Ma (2022) is a popular object detection algorithm known for its speed and accuracy. It works by dividing the image into a grid and predicting bounding boxes and class probabilities for each grid cell. The original YOLO model, introduced in 2015, revolutionized object detection by providing real-time inference capabilities.

In recent years, YOLO has undergone several changes, with each version building upon the successes and addressing the limitations of its predecessor.

The selection of YOLOv8 is grounded in the fact that it represents the most updated version of YOLO, with a combination of accuracy and speed that renders it ideal for our purpose. Furthermore, its robustness and extensive validation in diverse scenarios, backed by an active community and readily available learning resources, solidify YOLOv8 as the most robust and reliable choice for our study.

Table II
COMPARISON OF YOLOV8 MODELS

Model	mAPval	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	Params (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

The table II presents a comparison of various YOLOv8 models based on their size, mean Average Precision (mAPval), inference speed on CPU using ONNX (ms), inference speed on A100 using TensorRT (ms), and number of parameters (M). These models, YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, and YOLOv8x, offer different trade-offs between model size, speed, and performance.

YOLOv8n, despite its slightly lower mAPval compared to other models such as YOLOv8s and YOLOv8m, offers a respectable performance with an mAPval of 37.3. It stands out as the lightest and fastest model in the list, with only 3.2 million parameters and a processing time of 0.99 ms on A100 TensorRT. This feature makes it an ideal choice for applications where computational efficiency and speed are crucial.

On the other hand, YOLOv8m excels with a significantly better mAPval than YOLOv8n, reaching a value of 50.2, indicating higher precision in object detection. Although it has more parameters than YOLOv8n, with 25.9 million, it remains a moderately sized model compared to YOLOv8l and YOLOv8x. Additionally, with processing times of 1.83 ms on A100 TensorRT, YOLOv8m manages to maintain considerable efficiency while improving precision in object detection.

D. Model training

The selection of the **YOLOv8n** and **YOLOv8m** models was based on a consideration of the specific needs of the case. We opted for these lighter and more efficient models due to their ability to provide an optimal balance between performance and computational efficiency.
The training process was as follows:

- Preparation:
  - **Downloading Dataset:** The livestock image dataset was downloaded from the cloud and decompressed within the Colab workspace.
  - **Library Installation:** The Ultralytics library was installed, which is essential for YOLO's functioning.
  - **Model Import:** After assessing hardware constraints and processing times, two models were imported into separate environments, YOLOv8n and YOLOv8m, both lightweight and fast versions of YOLO pre-trained on the COCO dataset. This pre-training provides a solid foundation for subsequent learning with our specific dataset.
- Training:
  - **Images:** Both models were trained on 1100 images(Train data), representing 70% of the total dataset, validated on 302 images(Validation data: 20%) and tested on 144 images(Test data: 10%).
  - **Early stopping:** Early stopping was implemented with a patience setting of 5 to prevent model overfitting and enhance its generalization capability. Early stopping is a technique commonly used during

the training of machine learning models to monitor the model's performance on a validation dataset. It works by stopping the training process if the model's performance on the validation set stops improving, even if the training loss continues to decrease.

- **Optimizer:** The optimizer used in this case is AdamW automatically chosen by the YOLOv8 model, a variant of the Adam optimizer that includes weight decay, which helps prevent overfitting by penalizing large weights in the loss function and improving the model's generalization. It has a learning rate of 0.002, determining the magnitude of adjustments made to the model's weights during training, and a momentum of 0.9, a parameter that smooths the update of the model's weights by considering the direction and speed of previous changes in gradients.
- **Epochs:** In order for the model to have sufficient time to learn the patterns and characteristics present in the objects of interest in the images, 100 epochs were selected, utilizing the aforementioned early stopping technique.

- Final Model Selection:
  Upon analysis, it was noted that the YOLOv8n model exhibited minimal improvement in the final epochs. Conversely, the YOLOv8m model consistently demonstrated superior performance, indicating its effectiveness over the YOLOv8n counterpart.
  Based on these results, **YOLOv8m** was selected as the best model for the project.
- Data Augmentation with Albumentations:
  Data augmentation plays a crucial role in enhancing the diversity and robustness of the YOLO model. In this project, the 'augmentation' library was employed to apply a variety of transformations to the training images. The following transformations were utilized:
  - **Gaussian Blur:** This transformation blurs the image using a Gaussian filter, aiding in noise reduction and improving the model's ability to generalize.
  - **Median Blur:** Application of a median filter to the image helps in noise removal while preserving the edges of objects.
  - **Grayscale Conversion:** Converting the image to grayscale allows the model to focus on the shape and texture of objects in the image.
  - **Contrast Limited Adaptive Histogram Equalization (CLAHE):** CLAHE enhances the local contrast of the image, which can be beneficial in emphasizing important details.

E. Stratified k-Fold Cross-Validation

To ensure a good evaluation of the model's performance, we employed the Stratified k-Fold Cross-Validation procedure. This method is particularly beneficial when dealing with imbalanced datasets, as it preserves the distribution of target classes within each fold.

Here's how it worked: we divided our training dataset into 5 equal parts, denoted by k=5 folds. Unlike regular k-Fold Cross-Validation, which randomly divides the dataset, the stratified version ensures that each fold maintains the same proportion of target classes as the original dataset.

Next, the model was trained 5 times, each time using a different fold as the validation set while the remaining k-1 folds were used for training. This approach ensures that the model is tested on a diverse set of data while still maintaining a balanced representation of the target classes.

Throughout each iteration of training and validation, we recorded four metrics: accuracy, precision, recall, and f1-score. These metrics can be viewed in the results section.

F. Performance Assessment

After training the model with test data to evaluate its performance, the following metrics(Géron (2022)) were applied:

a) Confusion Matrix: The confusion matrix, as shown in the Figure 5, summarizes the model's performance by comparing its predictions with the ground truth labels. Each cell represents a combination of prediction and label:

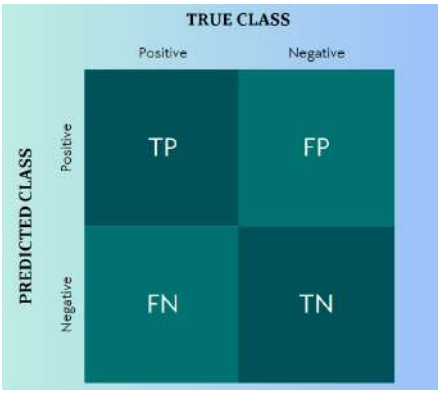

Figure 5. Illustrative image of the Confusion matrix

Source: Self made.

True Positives (TP) represent the correct predictions of the target class, where the model accurately detects and classifies the objects of interest. False Positives (FP) occur when the model incorrectly identifies objects that do not belong to the target class as belonging to it. On the other hand, True Negatives (TN) are instances where the model correctly identifies objects that do not belong to the target class as such. Lastly, False Negatives (FN) occur when the model fails to detect objects of the target class, mistakenly classifying them as not belonging to the class.

The confusion matrix allows visualizing the model's behavior for each object class. From it, other metrics can be calculated, such as Precision, Recall, and F1-score.

b) Accuracy: This metric is calculated as the ratio between the number of correct predictions and the total number of predictions:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

This metric is simple and easy to interpret, representing the proportion of samples classified correctly. However, in cases of class imbalance, other metrics such as precision, recall, and F1-score can offer a more detailed evaluation of the model's performance.

c) Precision: This metric indicates the proportion of positive predictions that are correct, i.e., the number of TP divided by the sum of TP and FP:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1}$$

A high Precision value indicates that the model has a low number of FP.
Precision reflects the model's reliability in its positive predictions. A high value indicates that the model is accurate in identifying objects of the target class.

d) Recall (Sensitivity): This metric indicates the proportion of target class objects that are correctly detected, i.e., the number of TP divided by the sum of TP and FN:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

A high Recall value indicates that the model has a low number of FN.
Recall reflects the model's ability to detect all objects of the target class. A high value indicates that the model is effective in detecting objects of the target class.

e) F1-score: This metric is the harmonic mean between Precision and Recall, providing a balanced measure of the model's performance:

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

A high F1-score indicates a good balance between accurate object detection and error minimization.
F1-score is a balanced metric that considers both Precision and Recall. A high F1-score indicates a good balance between accurate object detection and error minimization.

f) Intersection over Union: Intersection over Union (IoU) is a widely used metric in object detection tasks to evaluate the accuracy of detections. It is calculated as the intersection area between the predicted bounding box and the ground truth, divided by the union area of both boxes. The IoU formula is expressed as:

$$IoU = \frac{\text{Intersection Area}}{\text{Union Area}} \tag{4}$$

A high IoU value, typically close to 1, indicates a significant overlap between the prediction and the ground truth, which is considered an accurate detection. Conversely, a low IoU value, closer to 0, indicates little to no overlap between the detections, indicating errors in the prediction.

g) Mean Average Precision (mAP): This metric summarizes the model's accuracy at different IoU levels by calculating the mean of precision along a Precision-Recall curve. A high mAP value indicates a good overall performance of the model in object detection.
mAP is a comprehensive metric that summarizes the model's performance at different IoU levels. A high mAP value indicates that the model has a good overall performance in object detection.

G. Mobile Application Development

The mobile application development process is structured in different phases that, together, create an application capable of:

- Capturing an image selected by the user.
- Processing the image using an artificial intelligence model.
- Presenting the image along with the number of detections performed.
- Displaying the resulting image together with its respective detection.

a) User Experience (UX): In this case, a simplicity-focused approach is adopted for the application with the aim of making it accessible to any user. This is achieved through a minimalist and simple user flow.

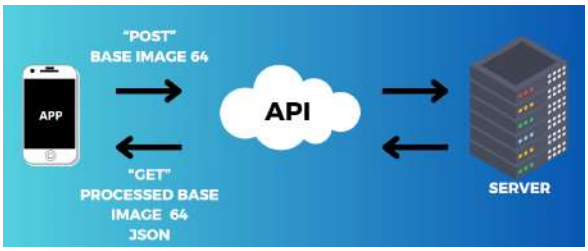

Figure 6. User flow diagram

Source: Self made.

As seen in figure 6, the app flow diagram follows a linear sequence: the user accesses the application, chooses the image they want to process from their gallery or from Google Photos, and sends it to the server using the "button" Upload image". The image is stored in a local folder until it is ready to be processed. By pressing the "Process Image" button, the processed image is retrieved from the API and displayed on the screen. If the user wants to process another image, they can return to the "Select Image" option and restart the flow. In the event that the user exits the application, several situations have been foreseen that could cause errors:

- If the user presses "Upload image" without selecting any image: No image will be sent to Server, so the API will not have an image to process and its response will be empty.
- If the user presses "Process Image" without selecting any image: The last processed image will be retrieved from the API and displayed in the application.

None of the mentioned errors will stop the application from working.

b) Mobile Application Design Stage: For this process, the mobile application development tool App Inventor was used. Through the use of block programming, the following processes are developed:

- Image Selection: Using the ImagePicker component, a button has been created that allows the user to access their gallery and Google Images to select the image they want to process. This image is displayed on the screen using an "Image" element and saved in a global variable called "ImageToUpload".

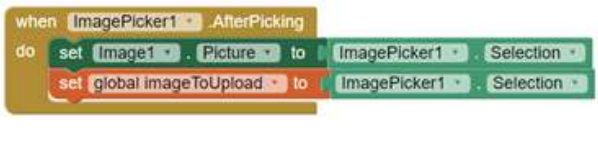

Figure 7. Image select blocks.

Source: App inventor.

- Upload Image: A button is implemented that, when selected, converts the image stored in the local variable "ImageToUpload" into a text string in Base64 format. This string is sent via the HTTP POST method to an API developed in Python, which allows you to take the file in Base64 format, convert it back into an image and store it in a local folder.

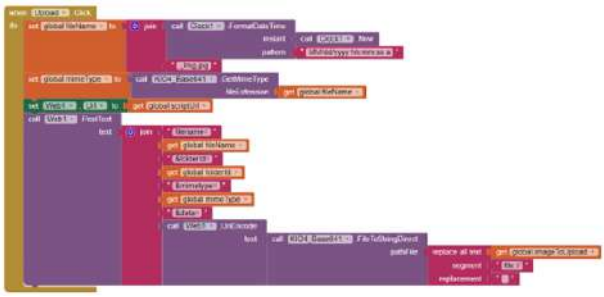

Figure 8. Image upload blocks.

Source: App inventor.

- Download the Image: A button called ”Process Image” has been implemented that, through the HTTP GET method, obtains a JSON object from the designed API. This JSON object is transformed into a dictionary that contains two elements: the number of cows counted and the image processed by the model in Base64 format. Using a Label element, the number of cows found is displayed on the screen. Additionally, the Base64 element is converted back to an image that is displayed on the screen by an appropriate element.

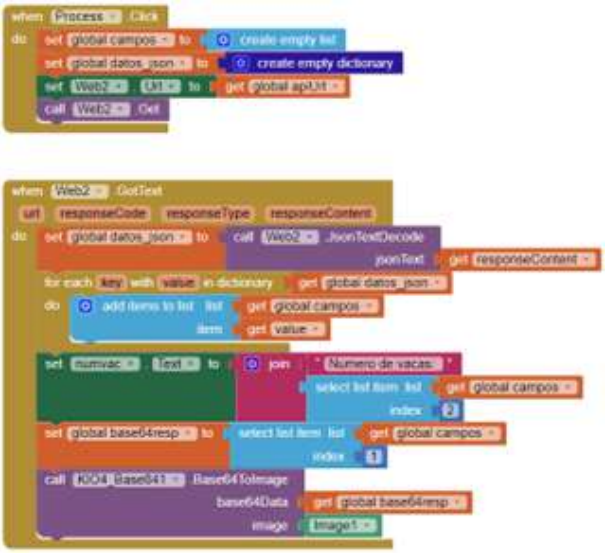

Figure 9. Image download blocks.

Source: App inventor.

c) API Creation: This stage is crucial to ensure the optimal performance of the application. A REST API has been selected for implementation due to its provision of a standardized interface, enabling secure information exchange between two computer systems through the HTTP communication protocol. The development of this API was conducted using the Flask framework and the Python programming language, following the outlined programming approach.

- Virtual environment: A virtual environment is created to isolate and independently manage the dependencies and configurations of the project through the virtualenv tool. This environment has python 3 and different dependencies that can be seen in the requirements.txt file among them the most relevant: flask (framework to develop the API), ultralytics (Tool to be able to use the AI model). With a defined environment with the tools to use, we proceed to start programming.
- Index.py: This Python code implements a web service using Flask. The application defines a root route via the decoration @app.route(’/’), which means that it responds to HTTP GET and POST requests directed to the root of the web server. Within this path, the server performs a series of actions: First, it loads a local image from the path specified in image path. Next, load a pre-trained YOLOv8 model using the Ultralytics library. It then runs object detection inferences on the loaded image using this model. Once the inferences are complete, it generates a new image with the detections highlighted. The next step is to convert this image to base64 format, which makes it easier to represent and transfer over the web. Using the converted image, construct a response JSON object that contains the base64-encoded image along with the number of object detections made. Finally, this JSON is returned as a response to the HTTP request.

III. Results

Dataset

During the initial stage of the results obtained in the development of this project, the construction of the database stands out, which consists of a total set of 1546 labeled images. An example of these images is shown in Figure 10. The location of each cow was labeled in each image. It is important to mention that this database was designed to be compatible with the YOLO architecture.

Figure 10. labeled image.

Source: Theos AI (2022).

The labels for each of the images are presented in TXT file format, as illustrated in Figure 11. These labels contain both the class and the coordinates of each object within the image. It is important to note that each line represents an object within the image. The structure of the labels is as follows:

- The first number represents the class of the object. In this case, since only one object is present, the first number in all the labels is ”0”.
- The second number indicates the position on the X-axis of the center of the object, normalized with respect to the width of the image.
- The third number indicates the position on the Y-axis of the center of the object, normalized with respect to the height of the image.
- The fourth number indicates the width of the object, also normalized with respect to the image.
- The fifth number indicates the height of the object, normalized with respect to the height of the image.

```
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
```

Figure 11. Label structure.

Source: Theos AI (2022).

The database was structured as follows:

- 1100 images for training (71%).
- 302 Validation images (20 %).
- 144 Test images (9 %).

MODEL TRAINING

A. *Model validation during training*

The model was trained using two different YOLO algorithms, in this case, YOLOv8m and YOLOv8n. After training, YOLO provides some performance metrics, among which the mean average precision (mAP) stands out, achieving 0.993 in both models. Therefore, the confusion matrix for each model was analyzed. These are presented in Figure 12, showing the number of cows that were correctly classified, as well as false positives, false negatives, and true positives. It should be noted that in this case, there are no true negatives since the model only detects whether it's a cow or not. Thus, if it's not a cow, there is no other class defined that could contain true negatives.

Figure 12. Confusion matrix for validation data.

Source: Self made.

In this case, the confusion matrix is constructed using validation data, comprising a total of 3998 instances intended to be correctly classified. Various performance metrics for the model are derived from this matrix, including accuracy, precision, recall, and the F1 metric. The results of these metrics are summarized in Table III, providing an overall view of the performance achieved by each evaluated model. It is important to highlight that, in this specific application, a higher model accuracy is sought. However, there is a notable superiority across all performance metrics for the YOLOv8m model. Consequently, this model is selected as the preferred one for implementation.

Table III
PERFORMANCE METRICS FOR VALIDATION DATA.

PARAMETER	YOLOv8m	YOLOv8n
Accuracy	0.951	0.928
Precision	0.961	0.941
Recall	0.989	0.985
F1	0.974	0.962

B. *Cross-Validation k-Fold*

Once the model is chosen, stratified k-fold cross-validation is performed in order to evaluate the model, as observed in the tableIV.

Table IV
RESULTS OF CROSS-VALIDATION K-FOLD

Parameter	Average ± Standard Deviations
Accuracy	0.9400 ± 0.0116
Precision	0.9523 ± 0.0120
Recall	0.9864 ± 0.0019
F1-Score	0.9690 ± 0.0062

The data presented in Table IV reveals minimal standard deviations across Accuracy, Precision, Recall, and F1-Score metrics. This consistency suggests that the model performs reliably across various folds, reflecting its stable performance.

C. *Performance metrics for test data*

With confidence in the YOLO V8M model exhibiting significant improvements in the validation phase, the performance evaluation was conducted using test data. After making predictions, the following results were obtained:

Figure 13. Confusion Matrix for test data.

Source: Self made.

As observed in Figure 13, results close to those obtained with the validation data are achieved. For a more detailed analysis of the performance metrics of this model on the test data, the following table is presented:

Table V
PERFORMANCE METRICS FOR TEST DATA.

PARAMETER	YOLOv8m
Accuracy	0.925
Precision	0.934
Recall	0.989
F1	0.9607

With the data obtained from the table V, it is appreciated that the model also achieves good results for the test data as these were expected. This suggests a high capacity for generalization and validity of the model with new data outside the training set.

With a defined model, the model is tested with images simulating a real-world application where the model needs to be put to the test. By implementing this through Colab, an example of how the result is obtained, including the identification and counting of the cows, can be seen in Figure 14.

Figure 14. Image of the result with the predictions.

Source: Self result image from Colab.

D. Considerations for Model Effectiveness

In turn, when analyzing the results obtained, certain cases were identified that could compromise the effectiveness of the model. Among them, the following stand out:

- There are obstacles in the image, such as trees or objects that may interfere with photographic capture. For this reason, it is crucial to consider the environment in which the model is deployed.
- The number of cows in the image also represents a critical factor. The data set used includes photographs with a maximum average of 35 to 45 cows. When the model is faced with images with a substantially larger number of cows (for example, 100 or 200), it may experience inaccuracies since it has not been trained on images featuring that specific number.
- The presence of animals other than cows in the image constitutes another challenge. The model tends to identify them as cows, since its training focused solely on detecting this type of cattle. To address this question, it would be necessary to train the model to recognize other types of animals as well. However, this adaptation could be applied specifically to farms that present this variability.

Mobile App

The implementation of the Flask web server for object detection has been successful. Upon accessing the server's root route, the local image was loaded correctly, and the pretrained YOLOv8m model performed precise object detection inference. The number of detected objects, in this case, cows, was counted, and an annotated image highlighting the detections was effectively generated. Furthermore, the conversion of the annotated image to base64 was completed without issues, allowing its transmission as part of the JSON response, providing a comprehensive output that is retrieved by the application built with App Inventor. Although the processed image may exhibit a different tone compared to the original due to the base64 conversion and vice versa, the counting and identification of the livestock were not affected. Overall, these results, as depicted in Figure 10 and in the repository hosting everything related to the application, validate the effectiveness and integrity of the Flask server implemented for object detection.

Although the results obtained by both the application and the API are accurate, it is important to note that the API has been implemented in an environment located on a local server, specifically on the computer where the code has been developed. This consideration is crucial, as the client-server communication architecture employed by the application implies that the availability of the application is directly tied to the state of the server, namely, to the availability and functionality of the computer hosting the server. Therefore, it is necessary to ensure that the server is powered on and operational for the application to be accessible and able to provide its services effectively.

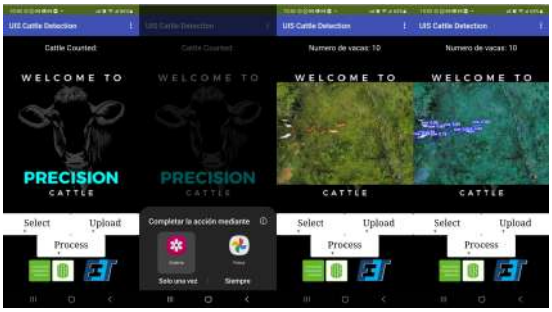

Figure 15. Mobile application in operation.

IV. Conclusions

A YOLOv8-based machine learning model was developed that was trained with a database of livestock images. The model achieved a precision of 94%, a precision of 95%, a recall of 98% and an F1 score of 96%, demonstrating its effectiveness for the livestock counting task.

This work has great potential to contribute to the management of livestock autonomously, facilitating and economizing the safety and control of livestock for ranchers. It differs from previous research by using modern technology such as drones and YOLOv8, obtaining more precise results.

The validation of the methodological approach has demonstrated its effectiveness. The results obtained are consistent with the theoretical expectations and the hypotheses raised.

The findings of this work can be easily applied to livestock farms that seek to automate processes. The model can be trained with specific images of the farm for greater accuracy. This work can have a significant impact on the livestock industry, society and future research.

However, the project also has some limitations. Adverse weather conditions can affect the flight of the drone and the quality of the images. Additionally, the model was trained with images with no more than 50 animals in the herd, so its performance could be affected in larger herds.

For future research, it is recommended to explore the application of the model in the eastern Colombian plains or Orinoquía Region where the largest livestock region is located and the herds are largest. It is also recommended to improve the mobile application to store data, georeference it and perform analysis for better livestock management.

In general, the experience of developing this project has been enriching and has allowed us to learn about the needs of ranchers and the operation of the YOLOv8 model. The knowledge and skills acquired will be used for future projects.

REFERENCES

AI, T. (2022). *Get started with object detection.* Retrieved 2024-03-21, from https://docs.theos.ai/get-started/object-detection

Andrew, W., Greatwood, C., & Burghardt, T. (2017). Visual localisation and individual identification of holstein friesian cattle via deep learning. In *Proc. ieee international conference on computer vision (iccv)* (pp. 22–29). Venice, Italy.

Berovides, V., Cañizares, M., & Gonzalez, A. (2005). Metodos de conteo de animales y plantas terrestres. , 1–15.

Chamoso, P., Raveane, W., Parra, V., & González, A. (2014). Uavs applied to the counting and monitoring of animals. In *Ambient intelligence-software and applications* (pp. 71–80). Springer.

DJI. (2024). *Dji mini 2 - product support.* Retrieved 2024-03-21, from https://www.dji.com/global/support/product/mini-2

Géron, A. (2022). *Hands-on machine learning with scikit-learn, keras, and tensorflow*. " O'Reilly Media, Inc.".

Jiang, P., Ergu, D., Liu, F., Cai, Y., & Ma, B. (2022). A review of yolo algorithm developments. *Procedia computer science*, *199*, 1066–1073.

Puetaman, I. A., & Báez, C. M. (2014). Prototipo de sistema de vigilancia para fincas ganaderas como prevención al abigeato. *Rev. UNIMAR*, *32*(1), 67–81.

Starn, O. (1991). *Reflexiones sobre rondas campesinas, protesta rural y nuevos movimientos sociales*. Lima.

Vergara, W. (2010). La ganadería extensiva y el problema agrario. el reto de un modelo de desarrollo rural sustentable para colombia. *Rev. Cienc. Anim.*, *3*, 45–53.

WRadio. (2023). *Las pérdidas que quedaron en 2022 para los ganaderos por el robo de bovinos.* https://www.wradio.com.co/2023/02/14/las-perdidas-que-quedaron-en-2022-para-los-ganaderos-por-el-robo-de-bovinos/. ([Consultado el 21 de marzo de 2024])