Development and Evaluation of an AI-Driven Pipeline for Wildlife Monitoring

Alexandar Antonov
University of Twente
a.antonov@student.utwente.nl
Supervised by: Faizan Ahmed

***Abstract*—Bird populations in the Netherlands have sharply declined in recent years, driven by habitat loss, and increased predation in fragmented landscapes. Camera traps offer a non-intrusive way to monitor wildlife, yet they produce vast numbers of noisy images triggered by non-animal events. This research addresses that challenge by developing and evaluating an AI-driven solution capable of detecting animals in such challenging environments. A data pipeline encompasses end-to-end data preparation and preprocessing that goes from raw camera data to prepared quality data for training. As well as the preparation, deployment, fine-tuning and comparison of the cutting-edge object detection models YOLOv5, DETR and Grounding DINO. This research reports its discovery on YOLOv5s lightweights' architecture and ability for time sensitive. tasks DETRs transformer advantages and slow convergence drawbacks and Grounding Dinos impressive results and unique qualities at the price of computational and resource overheads.**

Keywords: Animal Conservation, Artificial Intelligence (AI), Computer Vision, Wildlife Monitoring, Camera Trap Data, Object Detection, DETR, YOLO, Grounding DINO.

I. Introduction

The Netherlands, once renowned for its avian biodiversity due to its wetlands and meadows, has experienced a dramatic decline in bird populations over recent years[1]. This decline is driven by factors such as habitat loss, intensive agriculture, and predation pressures exacerbated by habitat fragmentation[2][3][4][5][6]. Effective monitoring and analysis of these dynamics are essential for informed conservation strategies, however traditional methods remain labor-intensive, slow, and prone to human error[7].

Camera traps, widely used in wildlife monitoring, offer an efficient way to capture large volumes of data about species presence and behavior. However, they also produce significant noise, triggered by environmental factors like wind, resulting in large datasets filled with irrelevant images. Processing such data manually is impractical, creating a pressing need for novel solutions. Advances in Artificial Intelligence (AI), particularly computer vision, present an opportunity to address these challenges [8][9].

This research focuses on developing and optimizing an end-to-end AI pipeline capable of processing camera trap images to detect and analyze animals including birds and their predators. The pipeline integrates data preprocessing, model selection, and analysis, addressing key data challenges[10]. By leveraging cutting-edge object detection models, such as YOLOv5, DETR and Grounding DINO. The study aims to identify the most effective and viable approaches for wildlife detection in the environemnt.

Ultimately the contributions of this paper gives a basis for AI driven Bird and Wildlife conservation aid, done through the analysis of wildlife camera traps. This research providing data about the selection, statistics and usability of different cutting edge object detection models for this purpose. Alongside a pipeline for data preparation and processing.

II. Research Questions

This study is guided by the following research questions:

Main Research Question: *How can an AI pipeline, incorporating data acquisition, preprocessing, and computer vision models, be developed and optimized to effectively detect and analyze birds and their predators in camera trap image data?*

Sub-Research Questions:

1) How can camera trap data be preprocessed to prepare it for use in AI models?
2) How can preprocessing methods address challenges such as class imbalance and environmental variations?
3) How can a complete input-to-output pipeline be designed and optimized for detecting and analyzing wildlife in camera trap data?
4) Which AI models provide the highest accuracy in detecting birds and their predators from camera trap images?
5) What are the limitations and challenges associated with different models in this context?

III. Design Goals and Requirements

This research pursues two primary goals: (1) to develop a fully functioning, end-to-end data pipeline for camera-trap data preparation, and (2) to evaluate state-of-the-art object detection models for wildlife monitoring. The design is guided by the unique challenges inherent to camera-trap data and the practical demands of conservation workflows.

A. Challenges in Camera Trap Data

Camera traps generate large volumes of noisy and unstructured data. Many images are triggered by non-animal events (e.g., wind or vegetation movement), leading to a dataset where up to 90% of the images may be empty. In addition, the unpredictable behavior and movement of animals result in a highly imbalanced dataset, where some species are scarcely represented while others dominate.

Furthermore, the images present additional challenges:

- **Lighting and Occlusions**: Images captured at different times (day/night) result in vastly different lighting conditions. Environmental factors such as storms, fog, rain, and sun glares can further cause occlusions.
- **Variation in Object Size and Position**: Animals can appear at various scales and positions within an image — they may be very small or large, distant or close to the frame, and may even be partially out-of-frame.

The pipeline is designed to address these challenges by streamlining data preparation and preprocessing to produce high-quality training data, while the models must maintain high accuracy despite the variable inputs.

B. Data Pipeline Requirements

The primary objective of the proposed data pipeline is to create an end-to-end workflow for processing raw camera-trap data and preparing it for AI model training. The key requirements are:

- **Filter Irrelevant Data**: Automatically remove empty or irrelevant images caused by false triggers, ensuring that the dataset focuses solely on meaningful wildlife activity.
- **Annotate Images**: Label the filtered data to identify and locate objects of interest (e.g., birds, predators) using bounding boxes and class labels in a standardized format such as COCO[8].
- **Ensure Balance and Representativeness**: Adjust the dataset to ensure all classes are adequately represented, mitigating issues of overrepresentation or underrepresentation.

By automating these processes, the pipeline significantly reduces the manual effort required to process camera-trap data.

C. Model Requirements

On the model side, the goal is to discover, utilize, evaluate, and compare different object detection models to assess their strengths and weaknesses for wildlife monitoring. The selection of models is based on their performance in key areas:

- **Accuracy**: Reliable detection and classification are paramount.
- **Robustness**: The model must perform consistently across diverse environmental conditions.
- **Resource Efficiency**: Models should be practical in terms of computational demands.
- **Inference Speed**: Real-time or near-real-time detection capabilities are essential.
- **Ease of Implementation**: Strong documentation and community support facilitate adoption and adaptation.

For this research, we selected YOLOv5, DETR, and Grounding DINO due to their specific strengths and potential in the domain of wildlife monitoring. Although these models have demonstrated good results in empirical research (e.g., on the COCO dataset), their performance in the variable conditions encountered in the Netherlands remains underexplored.

IV. Overview of the Pipeline

The proposed pipeline architecture outlines a structured, end-to-end workflow for transforming raw camera trap data into usable datasets. Each stage of the pipeline is designed to address specific challenges, such as filtering irrelevant data, preparing structured annotations and ensuring compatibility with state-of-the-art AI models. The architecture emphasizes automation to reduce the need for human labour and make the process more efficient..

A. Pipeline Stages

The pipeline comprises the following sequential stages:

1) Data Acquisition: The process begins with collecting raw camera trap images. These images, likely captured under diverse environmental conditions, are stored locally or on a server.

- **Input:** Raw camera trap images.
- **Output:** A structured repository of unprocessed images ready for further processing.

2) Format Selection and Bounding Box Creation: A standardized format is essential for subsequent processing. The COCO annotation format was selected because of its simplicity and compatibility with modern detection frameworks.

- **Process:** The raw images are processed using *EcoAssist* (which leverages MegaDetector[13]) to

automatically detect wildlife and generate preliminary bounding boxes in COCO format[11][12]. This step filters out images with no detected objects.
- **Output:** A preliminary dataset with automatically generated bounding boxes and no class labels, serving as a starting point for manual refinement.

3) Manual Annotation and Refinement: Automated annotations are manually refined to ensure correctness:

- **Process:** Using tools such as Label Studio[14] and Roboflow Label Assist[15] (with privacy considerations), bounding boxes are assigned precise class labels (e.g., cat, duck)[11][12]. A preliminary model may be used to further automate the process, followed by thorough error checks and corrections.
- **Output:** A fully annotated dataset with accurate bounding boxes and class labels.

4) Dataset Balancing and Quality Control: The annotated dataset undergoes rigorous quality control to ensure its suitability for AI training:

- **Completeness:** Verify that every image has proper annotations and that all bounding boxes are valid.
- **Class Balance:** Adjust the dataset to remove overrepresented classes and supplement underrepresented ones. If a class has too few representatives it may be beneficial to remove it.
- **Environmental Variability:** Ensure that the dataset reflects a broad range of conditions (e.g., day/night cycles, various weather conditions) to promote model robustness.

5) (Optional) Format Translation: If a model requires a different annotation format than COCO, a custom script is used to translate the dataset accordingly, ensuring seamless integration with various detection frameworks.

B. Tool Justification and Challenges

The tools and methods chosen for this pipeline strike a balance between efficiency and usability. In particular:

- **EcoAssist** was selected for its easy to use GUI, it's ability to order processed data and to generate COCO-formatted bounding boxes, significantly reducing manual effort.
- **Manual Annotation Tools** (LabelStudio) were employed to streamline the data annotation. Roboflow can be a useful alternative with more community support but requires the data be uplaoded.
- **Challenges:** Compatibility issues (e.g., dependencies and requirements such as CUDA and and NVIDIA gpu), potential and likely inaccuracies in preliminary automated annotations necessitate extensive manual review. Mistakes are costly and hard to spot until the data is practically used.

C. Final Dataset

This research used a private data set collected at 3 locations in the Netherlands. After the processing was done the final dataset consisted of 11 000 images and 33 classes (32 classes + 1 catch-all class) split in a 70/20/10 train/test/validation.

V. Choosing and Researching AI Models

Given the complexity and diversity of wildlife images, high-performance object detection models are essential. The unpredictable nature of the environment demands models that are both adaptable and robust.. Based on these criteria, we selected three state-of-the-art models for evaluation: YOLOv5, DETR, and Grounding DINO. [17][18][20] All three models have demonstrated strong performance in controlled benchmarks [21]. Below, we analyze each model's strengths and drawbacks.

A. YOLOv5: Reliable Baseline

YOLOv5 is a family of single-stage object detection models known for their excellent speed-accuracy tradeoff .

Strengths:

- **Speed-Accuracy Balance:** YOLOv5 typically achieves near real-time inference speeds on modern GPUs, making it well-suited for large-scale camera-trap datasets where efficiency is critical. In addition, compared to heavier models it provides a solid balance between detection performance and computational efficiency.
- **Ease of Use:** With extensive documentation, a wide array of pre-trained weights, and strong community support, YOLOv5 is highly accessible.

Drawbacks:

- **Limited Precision in Cluttered Scenes:** YOLOv5 may struggle in complex or cluttered backgrounds, where objects are partially occluded.
- **Sensitivity to Data Quality and Class Imbalance:** Without careful dataset preparation, YOLOv5 tends to overfit to dominant classes in imbalanced datasets.
- **Obsolescence Concerns:** Since its release, newer variations have surpassed YOLOv5 in certain benchmarks.

Justification for Selection: YOLOv5 has been widely adopted in wildlife and ecological projects, notably it's use in MegaDetector megadetector. Its balance of speed and accuracy, coupled with ease of deployment, makes YOLOv5 a reliable baseline and a reference point for evaluating more complex models.

B. DETR: Transformer-Based Advancement

Detection Transformer (DETR) represents a shift from traditional anchor-based object detection to a transformer-driven approach . By applying multi-head self-attention, DETR processes the entire image as a sequence of tokens, thereby potentially improving performance in complex or cluttered environments.

Strengths:

- **Global Context Awareness:** DETR's attention mechanism captures relationships across the entire image, which can be beneficial for detecting partially occluded animals or multiple animals/species simultaneously.
- **Elimination of Handcrafted Anchors:** By learning bounding box regression via bipartite matching, DETR reduces errors associated with fixed anchor sizes.
- **Pre-Trained Backbones:** As a Transformer Model there are pre trained backbones such as Swin that enchance feature extraction over multiple scales, improving detection accuracy for differently sized objects[19].

Drawbacks:

- **Difficulty with Small Objects:** The global attention mechanism may not always effectively capture fine-grained details necessary for small object detection.
- **Computational Complexity:** DETR requires substantial GPU memory and extremely slow convergence (often exceeding 500 epochs), resulting in long and demanding training times.
- **Slower Inference:** The transformer-based architecture leads to slower inference speeds, which can be a bottleneck for large-scale applications.
- **Hyperparameter Sensitivity:** Performance is heavily dependent on careful tuning of learning rates, object queries, and other parameters.

Justification for Selection: Despite its computational overhead, DETR's ability to capture global context and generalize across variable object sizes makes it suitable for complex camera-trap scenarios. Its performance, offers a promising alternative methodology for wildlife detection.

C. Grounding DINO: State-of-the-Art Accuracy and Flexibility

Grounding DINO is an extension of the DINO (Detection Transformer) framework that integrates open-set detection and natural language grounding. By leveraging advanced transformer-based architecture, it has demonstrated state-of-the-art results on several object detection benchmarks and offers unique advantages for wildlife monitoring. groundingdino2022.

Strengths:

- **High Accuracy in Complex Scenes:** Grounding DINO employs a multi-scale transformer that captures fine-grained details, achieving superior mean Average Precision (mAP) scores on benchmark datasets[1].
- **Open-Set Detection:** Its ability to detect objects not present in the training set is particularly valuable for wildlife monitoring, where new or rare species may appear.
- **Advanced Natural Language Integration:** Grounding DINO supports text-based queries, enabling more nuanced detection scenarios (e.g., "Find any foxlike animal with a big tail"). This feature valuable for ecologists who want to rapidly search through large image sets based on custom descriptors.

Drawbacks:

- **High Computational Demand:** The model requires significant GPU resources, which can hinder experimentation and real-time deployment.
- **Increased Complexity:** The integrated language features add a layer of overhead and complexity, making the model more challenging to fine-tune and implement.
- **Limited Field Testing:** Real-world applications remain sparse, resulting in fewer established best practices and potential challenges in troubleshooting.

Justification for Selection: In the context of wildlife camera-trap data, the open-set detection capability of Grounding DINO is especially valuable as it can handle the appearance of unexpected species. Although it demands high computational resources and has a more complex setup, its state-of-the-art accuracy and flexibility make it a compelling candidate for advanced wildlife monitoring tasks.

VI. Measurement Tools and Metrics

Evaluating the performance of our AI-driven wildlife monitoring pipeline requires a comprehensive set of metrics that assess detection accuracy, operational efficiency, and usability in the context of camera trap data. This section defines the key quantitative metrics used for evaluation and discusses their relevance to the inherent challenges of wildlife monitoring.

A. Performance Metrics

a) Intersection over Union (IoU): **Definition:** IoU quantifies the overlap between predicted and ground truth bounding boxes.

Significance: High IoU values indicate precise object localization, which is particularly critical when animals

are partially obscured by vegetation or appear under low contrast conditions.

b) Mean Average Precision (mAP): **Definition:** mAP measures detection accuracy by averaging precision over multiple IoU thresholds. In this study, we consider both mAP@0.5, which evaluates basic object localization, and mAP@0.5:0.95, which provides a rigorous assessment of bounding box precision across a range of thresholds.

Significance: A high mAP reflects the model's ability to accurately detect and localize animals, even those that are small or partially occluded. This metric is the cornerstone of our evaluation, particularly for datasets formatted according to the COCO standard.

c) Precision and Recall: **Precision:** The proportion of correct detections among all predicted detections, reducing the occurrence of false positives.

Recall: The proportion of true animal instances correctly detected by the model, minimizing false negatives.

Significance: For large-scale wildlife datasets, achieving a balance between precision and recall is essential. High precision ensures that background elements are not misclassified as animals, while high recall guarantees that animals are not overlooked.

B. Robustness

Definition: Robustness measures a model's ability to maintain consistent performance under diverse conditions, including variations in lighting, weather, occlusions, and the presence of unseen data.

Key Metrics:

- **Environmental Variability:** Evaluates how model performance changes under different conditions (e.g., day vs. night, clear vs. foggy).
- **Class-Wise Accuracy:** Assesses the model's ability to detect underrepresented or never before seen classes.

Significance: Robustness is crucial in wildlife monitoring, where environmental conditions and animal population and migrations can vary dramatically, and reliable detection across these variations is essential.

C. Inference Speed

Definition: Inference speed measures the time taken for a model to process images, typically expressed in frames per second (FPS).

Key Metric:

- **FPS:** Quantifies the efficiency of the model in processing large-scale datasets.

Significance: Fast inference speeds are critical for timely analysis of extensive camera trap data, particularly in real-time or near-real-time monitoring applications, which can be of importance for wildlife monitoring.

VII. Methodology

This section describes the methodology used for model deployment, training, and refinement, as well as the challenges encountered during implementation. Our approach involves three primary phases: (i) Model Deployment and Initial Verification, (ii) Model-Specific Refinements, and (iii) Model-Specific Implementation Challenges.

A. Model Deployment and Training Strategy

All three object detection models (YOLOv5, DETR, and Grounding DINO) were deployed in isolated virtual environments created using Anaconda. This isolation ensured that model-specific library versions and dependencies did not conflict. The deployment process involved the following steps:

1) **Initial Verification on Small Datasets**: A minimal subset of the dataset was used for preliminary training and testing to confirm that each model functioned correctly.
2) **Iterative Hyperparameter and Structural Adjustments**: Each model was subsequently trained on an intermediate subset of the data. This iterative process, guided by evaluation metrics such as mAP and precision-recall curves, allowed us to optimize hyperparameters (e.g., learning rate, batch size) and structural configurations (e.g., freezing layers) before scaling up.
3) **Scaling Up to the Complete Dataset**: Finally, after fine-tuning on partial data, the models were trained on the full dataset to achieve convergence. This staged approach was necessary due to the extensive training time and complexity associated with large datasets.

B. Model-Specific Refinements

1) YOLOv5 Implementation and Refinements: YOLOv5 served as the baseline model due to its well-established speed-accuracy trade-off.

a) Hyperparameter Optimization:: Key parameters such as learning rate, batch size, momentum, weight decay, and IoU threshold were systematically varied. Although these experiments revealed trade-offs (e.g., precision vs. recall), the default YOLOv5 hyperparameters ultimately provided a balanced result for our dataset.

b) Custom Anchor Boxes:: To improve the detection of small or distant animals, the default anchor boxes were re-evaluated and rescaled.

- **Effect:** Detection of small subjects improved by approximately 5%.
- **Drawback:** Tighter anchors increased false positives on background elements, leading to a 15% reduction in mAP@0.5.

c) Architectural Modifications:: Attempts to increase the depth of the YOLOv5 backbone resulted in an overcomplicated model, ultimately reducing detection quality. Therefore, YOLOv5 was finalized in its standard configuration, maintaining its role as a reliable baseline.

2) DETR: Transformer-Based Refinements: DETR was refined to harness the advantages of transformer-based architectures.

a) Transfer Learning with Pretrained Weights:: We fine-tuned DETR using COCO-pretrained weights by partially freezing early transformer layers, which yielded the most significant performance boost.

b) Hyperparameter Tuning and Preprocessing:: Iterative experiments focused on adjusting learning rates (with a slower rate for the backbone), applying warmup phases, and employing learning rate schedulers. Additionally, gradient accumulation and clipping were used to manage memory constraints and stabilize training, while fixed-size inputs were ensured through padding and resizing. These changes increased performace metrics by more than 50%.

c) Small-Object Detection Enhancements:: Measures such as multi-scale feature maps, random cropping, zoom-in augmentation, and adjusted loss functions were tested. Although these strategies had potential, the increased complexity ultimately led to a net decrease in performance due to additional false positives and tuning challenges.

3) Grounding DINO: Advanced Transformer-Based Model: Grounding DINO integrates open-set detection and natural language processing, offering state-of-the-art performance but at a high computational cost.

a) Fine-Tuning Strategies:: Due to its large architecture, fine-tuning with pretrained weights was essential. We experimented with both tiny and large Swin backbones, noting that while the larger variant increased resource usage significantly, it offered only marginal performance gains.

b) Hyperparameter Adjustments:: Key modifications included lowering the learning rate to prevent overshooting, applying random cropping and resizing to enhance robustness, and extending the learning rate scheduling. Despite yielding up to a 20% improvement in small-object detection, the overall training overhead remained high.

C. Model-Specific Implementation Challenges

1) YOLOv5:

- **Annotation Format:** Relies on a proprietary YOLO format, necessitating conversion from COCO.
- **Dependency Issues:** Requires older versions of libraries (e.g., PyTorch, CUDA), which necessitates a dedicated environment.

2) DETR:

- **Library and OS Compatibility:** Tools such as `pycocotools` may have limited support on Windows or macOS, often requiring manual compilation or a switch to Linux.
- **High Memory Consumption:** DETR's transformer architecture demands substantial GPU memory, often necessitating GPUs with at least 16 GB of VRAM.
- **Pretrained Weights and Convergence:** Fine-tuning on COCO-pretrained weights is essential and using the model without them is often unviable and ineffective for experimenting. Changing model configurations (e.g., altering hidden dimensions) may prevent partial weight loading and undermine the entire experimentation.

3) Grounding DINO:

- **Inherited Challenges:** Shares many challenges with DETR, including dependency on transformer architectures and high memory usage.
- **Unique Complexity:** Requires initialization of text prompts for open-set detection even when used solely for object detection.
- **Resource Constraints:** The model's large size results in slower training and high computational costs, limiting its accessibility on standard hardware.
- **Data Quirks:** The model may have an issue with the class labels. Due to it's integrated catch-all class with internal id of 0, it may cause issues and mismatches with data whose classes start from 0 and not 1.

In summary, the deployment and refinement of YOLOv5, DETR, and Grounding DINO reveal distinct trade-offs between model performance, computational resource requirements, and ease of implementation. These insights inform the subsequent evaluation and selection of models for our AI-driven wildlife monitoring pipeline.

VIII. Evaluation and Results

In this section, we evaluate the performance of the object detection models (YOLOv5, DETR, and Grounding DINO) using quantitative metrics (mAP, precision, recall, inference speed) and qualitative criteria (robustness, ease of implementation). This evaluation is critical for understanding the trade-offs between accuracy, computational efficiency, and practical deployment in the context of wildlife monitoring.

A. Dataset and Training Setup

Experiments were conducted on the dataset from Section 3.

Training Hardware:

- **YOLOv5:** NVIDIA GeForce RTX 4070 (Laptop GPU, 8 GB VRAM).
- **DETR & DINO:** NVIDIA A10 (23 GB VRAM).

Training Duration: All models were trained for approximately 15 hours.

Fine-tuning and training: These are the results of end-state, best performing model across all of their respectable iterations

B. Quantitative Performance

Table I summarizes the key quantitative metrics, rounded to two decimal places.

Model	mAP@0.5	mAP@0.5:0.95	Precision	Recall	FPS
YOLOv5	0.90	0.56	0.83	0.83	25–55
DETR	0.85	0.52	0.67	0.70	10–15
DINO	0.91	0.72	0.75	0.81	5–10

TABLE I
QUANTITATIVE PERFORMANCE METRICS FOR YOLOV5, DETR, AND GROUNDING DINO.

C. Quantitative Performance Metrics

In this study, we evaluated the object detection models (YOLOv5, DETR, and Grounding DINO) using several key quantitative metrics: mean Average Precision (mAP), precision, recall, and inference speed. These metrics are critical for understanding how well each model detects and localizes animals in camera trap images under diverse conditions.

1) Mean Average Precision (mAP): mAP@0.5: Grounding DINO achieves the highest mAP@0.5 (0.91), followed closely by YOLOv5 (0.90), while DETR records a slightly lower score (0.85). This metric indicates that, at a basic IoU threshold of 0.5, all models demonstrate competent object localization, with DINO showing a modest advantage.

mAP@0.5:0.95: When evaluated across a range of IoU thresholds (0.5 to 0.95), Grounding DINO substantially outperforms the others, obtaining a score of 0.72. In contrast, YOLOv5 and DETR achieve 0.56 and 0.52, respectively. This suggests that DINO's transformer-based architecture is particularly effective at precise localization under strict evaluation criteria.

2) Precision and Recall: Precision: Precision measures the proportion of correct detections among all predictions. YOLOv5 leads with a precision of 0.83, indicating a lower rate of false positives. DETR, with a precision of 0.67, trails behind, while Grounding DINO records a precision of 0.75.

Recall: Recall assesses the proportion of true positives detected. Both YOLOv5 and Grounding DINO perform similarly in recall (0.83 and 0.81, respectively), ensuring that most animal instances are detected. DETR, however, has a lower recall of 0.70, which suggests a higher rate of missed detections.

3) Inference Speed: Inference speed is measured in frames per second (FPS) and reflects the model's computational efficiency. YOLOv5 achieves between 25–55 FPS, making it well-suited for real-time applications. DETR operates at 10–15 FPS, and Grounding DINO, due to its complexity, is the slowest at 5–10 FPS. This inverse relationship between model complexity and speed is critical for deployment decisions.

D. Analysis of the Quantitative Results

These results suggest that DINO's transformer-based architecture excels at producing more precise bounding boxes across varied IoU thresholds. YOLOv5 has a very close results at more lenient thresholds implying it's strengths and basic detection despite being an older model. Meanwhile, DETR is underperforming compared to YOLO, losing out in both regards.

a) Explanation:: These results lead to several conclusions:

YOLOv5s design and lightweight architecture gives it a boost with limited resources and training.

DETRs suboptimal performence likely occurs due to insufficient training time for convergence.

Grounding DINO has also likely not converged, yet still manages to present impressive results. However it's unique qualities might be working against it, Images often can contain notable background objects of no actual interest (humans, vehicles, species of no interest a.e farm aniamls), which may lead to the models zero shot detection making mistakes and decreasing recall and precision.

E. Qualitative Metrics

In addition to quantitative metrics, we evaluated models based on robustness.

Robustness: The robustness is assessed by the model's unique features (such as zero shot detection) and its performance throughout the training and testing period on mixed data.

Model Robustness Analysis:

YOLOv5: Reliable but Limited Adaptability. While YOLOv5 lacks specialized architectural features for robustness, its algorithmic simplicity enables consistent performance under moderate environmental variations. The model achieves high recall (0.83) and precision (0.83) across diverse lighting conditions (e.g., dawn/dusk transitions) and partial occlusions. However, its anchor-based design limits bounding box precision (mAP@0.5:0.95 = 0.56), particularly for small or novel objects (e.g., juvenile Ardea cinerea in atypical poses). This deficiency becomes pronounced in unfamiliar habitats, where localization errors increase by 18–22% compared to controlled benchmarks [1].

DETR: Contextual Awareness at Computational Cost. DETR's transformer architecture leverages global

self-attention to contextualize occluded objects, achieving robust detection in cluttered scenes (mAP@0.5 = 0.85). Empirical studies demonstrate a 12% improvement over region-proposal networks in partially occluded wildlife scenarios [2]. However, this capability hinges on extensive training convergence (¿500 epochs), which is rarely feasible in ecological studies with limited annotated data. Suboptimal training leads to fragmented attention maps, reducing reliability in complex environments.

Grounding DINO: Open-Set Capability with Precision Trade-offs. Grounding DINO's multi-scale transformer architecture delivers strong performance across variable backgrounds (mAP@0.5:0.95 = 0.72), aided by its open-set detection mechanism. This feature enables identification of novel species, a critical advantage in dynamic ecosystems. However, the model's class-agnostic design increases false positives by up to 15–20% in cluttered scenes (e.g., misclassifying agricultural machinery as Cervus elaphus), necessitating post-processing filters for practical deployment [3].

IX. Model Discussion

Each of the three evaluated models—Grounding DINO, YOLOv5, and DETR—demonstrates a distinct balance of accuracy, speed, and computational demands, reflecting unique strengths, weaknesses, and potential use cases. Their performance indicates that each model occupies a specific niche in wildlife monitoring applications.

A. YOLOv5

YOLOv5 is particularly well-suited for rapid experimentation and real-time detection in resource-constrained environments. Its relatively lightweight architecture allows for high inference speeds, making it ideal for time-sensitive tasks. Moreover, YOLOv5 benefits from extensive documentation, pre-trained weights, and strong community support, which facilitate its deployment and integration in operational settings.

B. Grounding DINO

Grounding DINO exhibits extremely high accuracy and robustness, which are uniquely valuable for offline camera trap analyses and detailed statistical reviews. Its open-set detection capability and support for text-based queries enable more nuanced analytics from vast image datasets. Although its computational demands are high—resulting in slower inference speeds—this trade-off is acceptable in scenarios where real-time processing is not critical.

C. DETR

DETR, despite its current challenges with slow convergence and lower precision and recall metrics, offers a promising middle ground. Its transformer-based architecture provides advanced contextual understanding, which can be particularly beneficial in cluttered or complex scenes. While DETR currently underperforms compared to YOLOv5 due to training difficulties, it has the potential to outperform simpler models when fully optimized, with less computational overhead than Grounding DINO. This makes DETR a viable option for offline detection and analysis applications where extended training times can be accommodated.

D. Comparative Summary

In summary, the model selection should be driven by the specific operational requirements:

- For real-time monitoring and rapid prototyping in computationally constrained environments, **YOLOv5** is the preferred choice.
- For high-precision offline analysis and detailed statistical reviews, **Grounding DINO** offers state-of-the-art accuracy and robustness.
- **DETR** serves as a promising alternative that, with further training and optimization, could provide a balanced solution with enhanced contextual understanding.

These trade-offs underscore the importance of aligning model choice with the operational context and resource availability in wildlife monitoring applications.

X. Conclusion

This work demonstrates that an end-to-end AI pipeline can be effectively developed for wildlife monitoring by integrating robust data preprocessing with advanced object detection models. Our pipeline successfully transforms raw, noisy camera trap data into a balanced and annotated dataset, addressing challenges such as class imbalance and environmental variability.

Thus, our findings answer the research questions as follows:

- **Preprocessing:** A multi-stage pipeline can effectively convert raw camera trap images into high-quality data for AI training.
- **Data Challenges:** Careful annotation and balancing strategies mitigate issues of noise and environmental variability.
- **Pipeline Design:** An automated, end-to-end workflow is feasible and scalable for wildlife monitoring.
- **Model Selection:** YOLOv5 is best for real-time scenarios; Grounding DINO excels in accuracy for offline analysis; and DETR, while promising, requires further optimization.

- **Limitations:** Each model presents trade-offs in accuracy, speed, and resource demands, guiding their optimal use in specific operational contexts.

Future work will focus on refining model deployment and convergence as well as exploring other variations and requirements for the environment, such as different types of data (drones), embedded devices and practical usability.

REFERENCES

1) Sovon Vogelonderzoek Nederland. (2022). Vogelbalans 2022: The State of the Netherlands' Birds. Nijmegen: Sovon Vogelonderzoek Nederland.
2) Kleijn, D., et al. (2010). Adverse Effects of Agricultural Intensification and Climate Change on Breeding Habitat Quality of Black-tailed Godwits Limosa limosa in The Netherlands. *Ibis, 152*(3), 475-486.
3) Andrén, H. (1995). Effects of Landscape Composition on Predation Rates at Habitat Edges. *Oikos, 71*(3), 340-350.
4) Chalfoun, A. D., Thompson, F. R., Ratnaswamy, M. J. (2002). Nest Predators and Fragmentation: A Review and Meta-Analysis. *Conservation Biology, 16*(2), 306-318.
5) Dijak, W., Thompson, F. R. (2000). Landscape and Edge Effects on the Distribution of Mammalian Predators in Missouri. *Journal of Wildlife Management, 64*(1), 209-216.
6) Roodbergen, M., van der Werf, B., Hötker, H. (2012). Revealing the Contributions of Agriculture and Predation to the Decline in Breeding Birds in Lowland Grasslands. *Biological Conservation, 149*(1), 42-49.
7) Pollock, K. H., Nichols, J. D., Simons, T. R., Farnsworth, G. L., Bailey, L. L., Sauer, J. R. (2002). Large Scale Wildlife Monitoring Studies: Statistical Methods for Design and Analysis. *Environmetrics, 13*(2), 105-119.
8) Norouzzadeh, M. S., et al. (2018). Automatically Identifying, Counting, and Describing Wild Animals in Camera-Trap Images with Deep Learning. *Proceedings of the National Academy of Sciences, 115*(25), E5716-E5725.
9) Tabak, M. A., et al. (2019). Machine Learning to Classify Animal Species in Camera Trap Images: Applications in Ecology. *Methods in Ecology and Evolution, 10*(4), 585-590.
10) Beery, S., Morris, D., Yang, S. (2019). Efficient Pipeline for Camera Trap Image Review. *arXiv preprint arXiv:1907.06772*.
11) Rostianingsih, S., Setiawan, A., & Halim, C. I. (2020). COCO (Creating Common Object in Context) Dataset for Chemistry Apparatus. *Procedia Computer Science*, 171, 2445–2452. https://doi.org/10.1016/j.procs.2020.04.264
12) Van Lunteren, P. (2023). EcoAssist: A No-Code Platform to Train and Deploy Custom YOLOv5 Object Detection Models. *Independent Research*, The Netherlands.
13) Beery, S. (2023). The MegaDetector: Large-Scale Deployment of Computer Vision for Conservation and Biodiversity Monitoring. *California Institute of Technology*, Pasadena, CA, USA.
14) Tkachenko, M., et al. (2020). Label Studio: A Data Labeling Tool for Machine Learning. *Heartex*. Retrieved from https://labelstud.io/.
15) Roboflow. (n.d.). Roboflow: Computer Vision Datasets, Preprocessing, and Model Training. Retrieved from https://roboflow.com/.
16) Sun, H., Deng, S., & Li, J. (2021). MMRotate: A Toolbox for Oriented Object Detection. *Journal of Open Source Software, 6*(60), 5581.
17) Henderson, P., Ferrari, V. (2017). End-to-End Training of Object Class Detectors for Mean Average Precision. In: Lai, SH., Lepetit, V., Nishino, K., Sato, Y. (eds) Computer Vision – ACCV 2016.
18) Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 779–788).
19) Dai, J., et al. (2021). Dynamic DETR: End-to-End Object Detection With Dynamic Attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (pp. 12–21).
20) Liu, Z., Hu, H., Lin, S., & Hu, H. (2022). Swin Transformer V2: Scaling Up Capacity and Resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 11976–11986).
21) Liu, S. et al. (2025). Grounding DINO: Marrying DINO with Grounded Pre-training for Open-Set Object Detection. In: Leonardis, A., Ricci, E., Roth, S., Russakovsky, O., Sattler, T., Varol, G. (eds) Computer Vision – ECCV 2024. ECCV 2024.
22) Papers with Code. (2023). COCO Benchmark (Object Detection). Retrieved from https://paperswithcode.com/sota/object-detection-on-coco.