A MODEL GENERALIZATION STUDY IN LOCALIZING INDOOR COWS WITH COW LOCALIZATION (COLO) DATASET

A PREPRINT

Mautushi Das
School of Animal Sciences
Virginia Tech
Blacksburg, VA 24061
mautushid@vt.edu

Gonzalo Ferreira
School of Animal Sciences
Virginia Tech
Blacksburg, VA 24061
gonf@vt.edu

C. P. James Chen *
School of Animal Sciences
Virginia Tech
Blacksburg, VA 24061
niche@vt.edu

July 19, 2024

ABSTRACT

Precision livestock farming (PLF) increasingly relies on advanced object localization techniques to monitor livestock health and optimize resource management. This study investigates the generalization capabilities of YOLOv8 and YOLOv9 models for cow detection in indoor free-stall barn settings, focusing on varying training data characteristics such as view angles and lighting, and model complexities. Leveraging the newly released public dataset, COws LOcalization (COLO) dataset, we explore three key hypotheses: (1) Model generalization is equally influenced by changes in lighting conditions and camera angles; (2) Higher model complexity guarantees better generalization performance; (3) Fine-tuning with custom initial weights trained on relevant tasks always brings advantages to detection tasks. Our findings reveal considerable challenges in detecting cows in images taken from side views and underscore the importance of including diverse camera angles in building a detection model. Furthermore, our results emphasize that higher model complexity does not necessarily lead to better performance. The optimal model configuration heavily depends on the specific task and dataset, highlighting the need for careful model selection tailored to the particular application. Lastly, while fine-tuning with custom initial weights trained on relevant tasks offers advantages to detection tasks, simpler models do not benefit similarly from this approach. It is more efficient to train a simple model with pre-trained weights without relying on prior relevant information, which can require intensive labor efforts. Future work should focus on adaptive methods and advanced data augmentation to improve generalization and robustness. This study provides

*Corresponding author: James Chen <niche@vt.edu>

practical guidelines for PLF researchers on deploying computer vision models from existing studies, highlights generalization issues, and contributes the COLO dataset containing 1254 images and 11818 cow instances for further research.

Keywords Object detection · Cows · Model generalization · Model selection

1 Introduction

Object Localization and Its Applications

Localizing livestock individuals from images or videos has become an essential task in precision livestock farming (PLF) [1]. Such techniques allow farm operators to manage animal well-being and health in real-time, optimizing their resource management and improving sustainability [2, 3]. Technically speaking, in the field of computer vision (CV), which is a subfield of artificial intelligence (AI) that focuses on translating visual information into actionable insights, localization tasks can be further categorized into object detection, object segmentation, and pose estimation. Object detection is the simplest form among these tasks, localizing objects of interest by enclosing them within a rectangular bounding box defined by x and y coordinates, pixel width, and pixel height [4]. Successful instances in this category include YOLO (You Only Look Once) [5], Faster R-CNN (Region Convolutional Neural Networks) [6], and SSD (Single Shot MultiBox Detector) [7]. These models have been adopted and applied by animal scientists for detection in precision livestock farming. For example, a study [8] leveraged the DRN-YOLO model [9] to predict the eating behavior of dairy cows. This approach automates the assessment of feeding behavior, a critical indicator of cow health and productivity, and has saved labor efforts in complex farm settings. Another notable work is presented in [10], where the authors developed a posture detection system for pigs using deep learning models such as Faster R-CNN, SSD, and R-FCN, coupled with 2D imaging. This system accurately identifies standing and lying postures of pigs under commercial farm settings.

To achieve finer localization, object segmentation is employed to outline object contours pixel-wise, while pose estimation is performed by orienting and marking the key points of the object [11]. Some popular object segmentation models include Mask R-CNN [12], MS R-CNN [13], and U-Net [14]. This method of segmentation has also been applied in the field of PLF. In the study [15], the authors developed a method using Mask R-CNN [12] to segment and outline cattle in feedlots. Their technique enhances images and extracts key frames to accurately detect cattle, achieving superior precision with a mean pixel accuracy of 0.92. This supports advanced, real-time monitoring of cattle in PLF. Another study group [16] developed the PigMS R-CNN framework [13] to enhance the monitoring of group-housed pigs. This framework employs a 101-layer residual network along with a feature pyramid network and soft non-maximum suppression to effectively detect and segment pigs, thereby improving the accuracy of identifying and locating individual pigs in complex environments.

Model Generalization, Pre-Training, and Fine-Tuning

Although implementing image-based systems in livestock production has become more common, current studies primarily focus on accuracy in homogenous environments and rarely address the challenges of model generalization. How a model can generalize to new environments is critical when farm operators deploy existing CV models in their own settings. Good generalization performance ensures that the model can reproduce similar results as reported in the original study, even in new environments with different conditions. Factors such as camera angles and the presence of

occlusions can impact generalization in the deployment environment. Deploying the same model in a new environment does not necessarily guarantee the same performance as reported in the original study. Li et al. [17] also pointed out that the lighting conditions on farms in real applications can be highly variable, leading to poor generalization performance.

One explanation for poor generalization is the discrepancy between the pre-training process and the specific use case. Most CV models are released with pre-trained weights, obtained from training on a large-scale dataset. For example, the COCO dataset [18] is a general-purpose dataset containing over 200,000 images and a wide range of object categories, such as vehicles and household items. Directly deploying a model pre-trained on the COCO dataset to detect cows in a farm setting may not ensure satisfactory performance, as the dataset does not contain enough cow instances in different view angles or occlusions. To alleviate this discrepancy, fine-tuning is a common practice that modifies the prediction head of the pre-trained model and updates the weights on a new dataset more relevant to the specific use case. Most application studies have adopted this approach to improve model generalization on their specific tasks [19, 20, 21].

Nevertheless, fine-tuning is not guaranteed to be successful, as the outcome depends on both the quantity and quality of the annotated dataset. For example, Zin et al. [22] deployed an object detection model to recognize cow ear tags in a dairy farm. Although the model achieved a high accuracy of 92.5% in recognizing the digits on the ear tags, more than 10,000 images were required for fine-tuning. Assembling such a large dataset is labor-intensive and requires specific training in annotating the images. The annotated dataset is rigorously organized in a specific format. For example, the COCO annotation format [18] stores image information, object class, and annotations of the entire dataset in one nested JSON format. In contrast, the YOLO format [23], another common format for object localization, stores information of one image in one text file, with each line representing one object instance in the image. Additionally, unlike the COCO format that stores bounding box coordinates in absolute pixel values, the YOLO format stores the coordinates in relative values to the image size. These technical details are key to valid annotations, which are usually facilitated by professional annotation tools such as Labelme [24], CVAT [25], or Roboflow [26].

Model Complexity and Performance

Another factor affecting model generalization is model complexity. Generally, model complexity is quantified by the number of learnable parameters in a model [27]. A more complex model can often generalize better to unseen data with high accuracy. However, this high complexity also comes at the cost of computational resources in the form of memory or time [28]. The computational cost may further limit how models can be deployed in real-world applications, where real-time processing or edge computing is desired for fast or compact systems. For instance, the VGG-16 model [29] has 138 million parameters and requires a video memory of at least 8GB, while the ResNet-152 [30] has around 60 million parameters with a recommended video memory of 11GB. Recent models for object detection, such as YOLOv8 [31] and YOLOv9 [32], have been developed in different sizes, providing a flexible choice for researchers to balance between generalization performance and computational cost. In YOLOv8, the spectrum of model complexity ranges from the highly intricate YOLOv8x, containing 68.2 million parameters, to more streamlined variants like YOLOv8n with only 3.2 million parameters. The memory demand for the model architecture alone, without considering the

intermediate results during training or inference, is larger by a factor of 21 for YOLOv8x (136.9 megabytes) compared to YOLOv8n (6.5 megabytes). Therefore, the trade-off between model complexity and computational cost is a critical factor to consider when deploying CV models in real-world scenarios.

YOLO Models

Before YOLO, object detection methods typically involved either using "sliding windows with classifier" or "region proposals with classifier." The sliding windows method required running the classifier hundreds or thousands of times per image. On the other hand, advanced region proposal-based approaches divided the task into two steps: first, identifying potential object regions (i.e., region proposals) and then applying a classifier to these regions. In contrast, YOLO models are capable of performing object detection in a single pass through the network, which is why the acronym YOLO stands for "You Only Look Once."

YOLOv8 [31], building on the YOLOv5 [33] architecture, incorporates the C2F module (cross-stage partial bottleneck with two convolutions), a refinement of the CSPLayer of YOLOv5 featuring two convolutional operations. It employs SiLU activation over traditional ReLU and Sigmoid [34] for smoother gradient flow, enhancing CNN performance. The module divides input from a convolutional layer, processes one half through bottleneck layers (offering two types: with and without shortcuts similar to ResNet [35]), then merges it back for further convolution. This design, along with a spatial pyramid pooling fast (SPPF) layer in its backbone, supports efficient feature pooling and multi-scale detection by using three distinct heads, thereby optimizing object detection across varying sizes. Furthermore, YOLOv8 innovates with an anchor-free approach, directly predicting bounding boxes and confidence scores, thus simplifying the network and reducing computational overhead [36, 37, 38].

Deep learning models, including the YOLO family, encounter an information bottleneck issue [39, 40], where the retention of input information diminishes as data is compressed into features. This loss is exacerbated in deeper network layers, often leading to reduced model efficacy. One approach to mitigate this involves expanding the model's width, i.e., increasing the number of parameters, which allows for broader feature mapping and potentially recaptures lost information. However, simply increasing model size can lead to unreliable data outputs and does not proportionally enhance model performance.

YOLOv9 addresses these challenges through innovations like Programmable Gradient Information (PGI) and the Generalized Efficient Layer Aggregation Network (GELAN) [32]. PGI optimizes gradient generation to minimize deep layer information loss, featuring a main branch for inference and auxiliary branches for enhanced training. GELAN, by integrating and pooling convolutional layers, ensures robust feature retention. This adaptive system notably boosts inference speed by 20% [32] on the COCO dataset [18], while its multi-level auxiliary information facilitates the detection of objects across varying sizes, making YOLOv9 particularly effective in identifying smaller objects compared to its predecessors.

Public Datasets

A public dataset helps the community to develop methodology based on the same baseline. One famous example in computer vision is the ImageNet dataset [41], which serves as a benchmark for image classification. AlexNet [42], the winner of the ImageNet Large Scale Visual Recognition Challenge in 2012, demonstrated its outstanding capability to classify images in the ImageNet dataset using Rectified Linear Units (ReLU) as the activation function, rather than the traditional sigmoid function. The success of AlexNet accelerated the development of CV models in the following years, such as VGG [43], GoogLeNet [44], ResNet [35], and DenseNet [45]. However, similar to the challenges that pre-trained models face in specific use cases, a generic public dataset, such as ImageNet [41] and COCO [18], may not be sufficient for PLF applications.

There have been efforts to create public datasets for livestock scenarios. For example, the CattleEyeView dataset was collected to support applications like cattle pose estimation and behavior analysis, providing extensive annotations across 30,703 frames from top-down video sequences of cows [46]. Another study [47] leverages a public dataset for pigs comprising 3600 images from 12 videos of group-housed pigs. The dataset is particularly designed for applications such as pig tracking. Additionally, the "OpenCows2020" dataset, developed by researchers from the University of Bristol, is a public dataset specifically designed for advancing non-intrusive monitoring of cattle. It supports precision farming applications such as automated productivity assessment, health and welfare monitoring, and veterinary research, including behavioral analysis and disease outbreak tracing. The dataset consists of 11,779 images with 13,026 labeled objects, mainly focusing on cattle [48].

Study Objectives

This study aims to use YOLO-family models to explore model generalization across varying environmental settings and model complexities within the context of indoor cow localization. Object detection, being the simplest form of localization, serves as an ideal baseline for extending to more complex tasks such as segmentation and posture estimation. Starting with object detection allows for a clear and foundational understanding of model behavior and performance, which can then inform and enhance the approach to more complex tasks. Consequently, this study examines three hypotheses:

- **Model generalization is equally influenced by changes in lighting conditions and camera angles.** Should camera angles prove more impactful than lighting conditions, it would be advisable to prioritize camera placement when deploying CV models in new environments.
- **Higher model complexity guarantees better generalization performance.** If a highly complex model does not ensure superior performance, future studies might consider adopting less computationally demanding models that still enhance performance.
- **The advantages of using fine-tuned models as initial training weights are persistent over pre-trained models.** If the advantages diminish as the training sample size increases in a similar cow localization task but

different environments, the fine-tuning efforts may be deemed unnecessary when the deployment environment varies over multiple locations on a farm.

To facilitate these investigations, a public dataset named COws LOcalization (COLO) [49] will be developed and made available to the community. The findings of this study are expected to provide practical guidelines for Precision Livestock Farming (PLF) researchers on deploying CV models, considering available resources and anticipated performance.

2 Materials and Methods

Cow Husbandry

All procedures involving cow handling and image capturing were conducted in accordance with ethical guidelines and approved by the Virginia Tech Institutional Animal Care and Use Committee (IACUC #22-146). The cows studied were part of the dairy herd at the Virginia Tech Dairy Complex in Blacksburg, Virginia, USA, which comprises approximately 80% Holstein and 20% Jersey cows. For the 'External' setting, the study included 100% Holstein cows. The milking cows were housed in pens within a free-stall barn, featuring two rows of sand-bedded stalls, headlocks at the feed bunk, and two water troughs per pen. The stocking density was maintained at 100% (i.e., one cow per stall). Heat stress was managed using automatic 48-inch diameter fans positioned over the stalls and feeding alleys. Cows were milked twice daily at 1:00 am and 12:00 pm in a double-twelve parallel milking parlor. They were fed ad libitum (with less than 5% refusals) once daily at 8:00 am with a total mixed ration (TMR) consisting of approximately 42% corn silage, 8% grass hay, and 50% concentrate on a dry matter basis. Manure from the stalls was removed at each milking session by personnel driving the cows to milking. Manure from the walking alleys within the pen was cleared two or three times daily using an automatic flushing system with recycled water. Fresh or recycled sand was added on a weekly basis.

Image Dataset

The images in this study were collected using the Amazon Ring camera model Spotlight Cam Battery Pro (Ring Inc.), which offers a real-time video feed of dairy cows. Three cameras were installed in the barn: two at a height of 3.25 meters (10.66 feet) above the ground covering an area of 33.04 square meters (355.67 square feet). One camera provided a top view while the other was angled approximately 40 degrees from the horizontal to offer a side view of the cows. These are hereafter referred to as *the Top-View camera* and *the Side-View camera*, respectively. A third camera, termed *the external camera*, was set at a lower height of 2.74 meters (9.00 feet) and covered a larger area of 77.63 square meters (835.56 square feet). Positioned 10 degrees downward from the horizontal, it captured a challenging perspective prone to occlusions among cows.

Images were captured using an Ring Application Programming Interface (API) [50], configured to record a ten-second video clip every 30 minutes continuously for 14 days. Since the image quality relies on the camera's internet connection, which was occasionally unstable, some images were found to be tearing or unrecognizable. Hence, the resulting dataset was manually curated for consistent quality, comprising 504 images from *the Top-View camera*, 500 from *the Side-View*

camera, and 250 from *the external camera*. These images were further categorized based on the lighting conditions: for *the Top-View camera*, 296 images were captured during daylight, 118 in the evening under artificial lighting, and 90 as near-infrared images without artificial light. From *the Side-View camera*, 113 images were taken in the evening, and 97 as near-infrared images. All images from *the external camera* were captured during the day. Image examples are shown in **Figure 1a**.

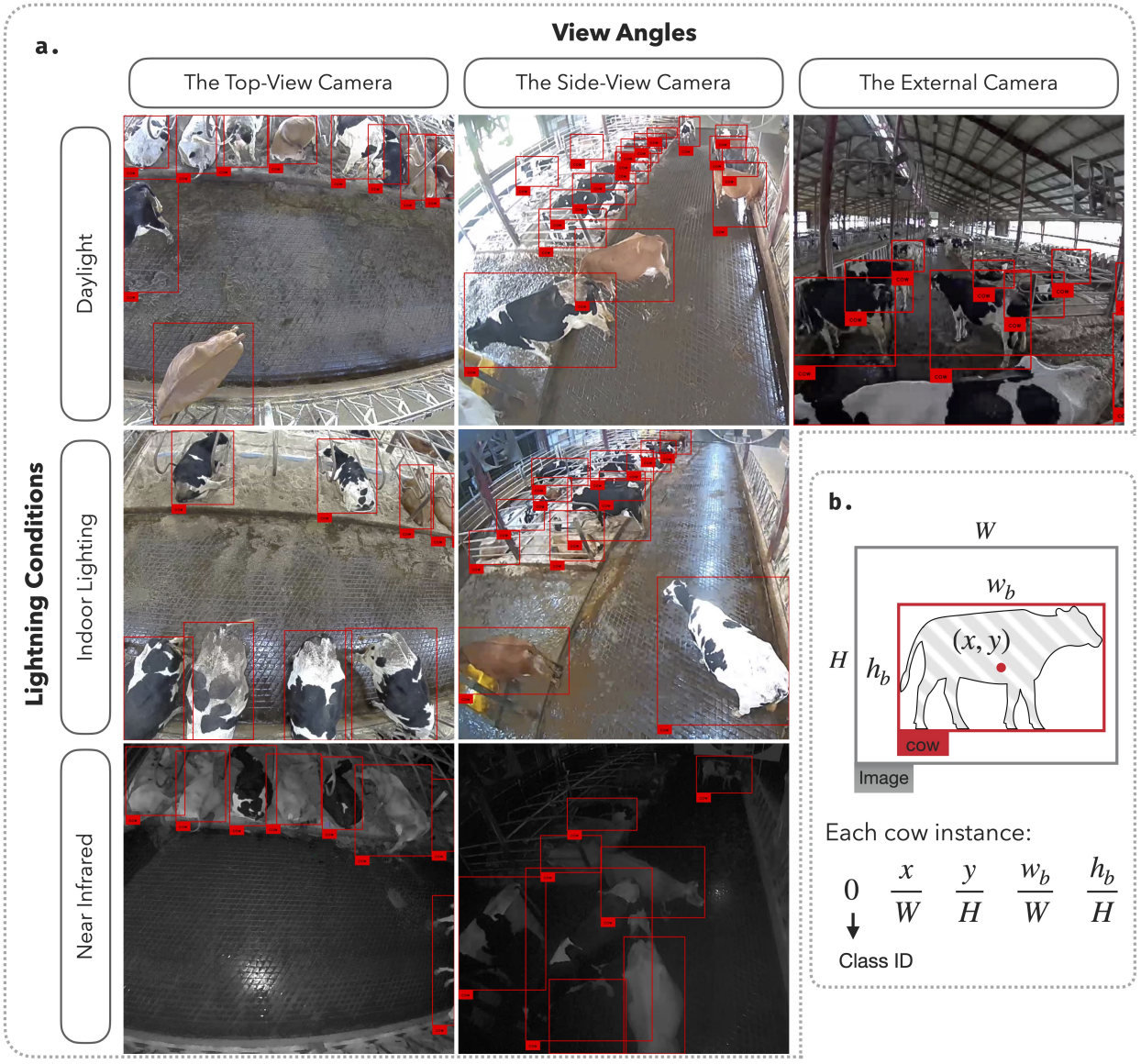

Figure 1: Overview of the COLO dataset. 1a. Seven instance images from the dataset with red bounding boxes labeling the location of cows. The columns show three different view angles: top-view, side-view, and external. The rows show three different lighting conditions: daylight, indoor, and near-infrared. 1b. An example of the annotated image in YOLO format. W, H, w_b, and h_b represent the width, height, width of the bounding box, and height of the bounding box, respectively. x and y represent the center coordinates of the bounding box.

The image annotations were conducted using an online platform, Roboflow [26], to define cow positions in the images. The bounding boxes were manually drawn to enclose the cow contours, providing the coordinates of the top-left corners and the width and height of the boxes. If cows were partially occluded, the invisible parts were inferred based on the adjacent visible parts. If the cow position was too far from the camera, making important body features such as the head, tail, and legs unrecognizable, the cow was excluded from the dataset. The final annotations were saved in the YOLO format [23], where annotations were stored in a text file with one row per cow in the image, each row containing the cow's class, center coordinates, width, and height of the bounding box. The graphical representation of the annotated images is shown in **Figure 1b**.

Model Training

The model training was implemented using the Python library Ultralytics [51]. The model hyperparameters were set to the default values in the library. The training epochs were set to 100, and the batch size was set to 16. The implemented data augmentation included randomly changing the image color hue, saturation, and exposure to improve the model's generalization to different lighting conditions. Geometry augmentation was also applied by randomly flipping the images horizontally, copying and pasting to mix up object instances across multiple images to increase data diversity, and randomly scaling the images to simulate different distances between the camera and the cows. The details of the hyperparameters are shown in **Table 3**. The training was conducted on an NVIDIA A100 GPU (NVIDIA, USA) with 80GB video memory provided by Advanced Research Computing at Virginia Tech.

Model Evaluation

The examined YOLO models are object detection models that return positions of detected objects (i.e., cows in this study) for the evaluated images. The detections are represented by a list of bounding boxes. Regardless of specific procedures among YOLO variants for computational efficiency, such as YOLOv8, which integrates objectness scores and conditional class probabilities into a single confidence score, each detection generally consists of $4 + c$ elements: the xy-coordinates, width, and height of the bounding box, and the c confidence scores indicating the probability of the object belonging to each of the c classes. The class with the highest confidence score is considered the predicted class of the object. To evaluate the model performance, two aspects are considered: the localization accuracy and the classification accuracy. The localization accuracy is measured by the Intersection over Union (IoU) between the predicted bounding box and the ground truth bounding box. On the other hand, the classification accuracy is measured by the precision and recall given the confidence threshold. If the confidence score of a detection is higher than the threshold, the detection is considered a positive detection. Otherwise, the detection is neglected. Combining the localization and classification accuracy, the mean Average Precision (mAP) averages the area under the precision-recall curve across all the classes. The curve is generated by varying the confidence threshold from 0 to 1 given an IoU threshold. In this study, four metrics were used in the evaluation: the precision and recall at the confidence threshold of

0.25 and IoU threshold of 0.5, the mAP at the IoU threshold of 0.5 (noted as mAP@0.5), and the averaged mAP at varying IoU thresholds ranging from 0.5 to 0.95 (noted as mAP@0.5:0.95).

Study 1: Benchmarking Model Generalization Across Different Environmental Conditions

To compare the performance drop between different view angles and lighting conditions, we designed a cross-validation strategy where models were trained on one dataset configuration and tested on another. There are five training configurations in this study (**Figure 2**):

- **Baseline:** The model was trained and evaluated on the dataset characterized for all conditions, including top-view, side-view, daylight, evening, and near-infrared images. The images did not overlap between the training and evaluation sets.
- **Top2Side:** The model was trained on the top-view images and evaluated on the side-view images.
- **Side2Top:** The model was trained on the side-view images and evaluated on the top-view images.
- **Day2Night:** The model was trained on the daylight images and evaluated on the evening images, including both artificial lighting and near-infrared images.
- **External:** The model was trained on images collected by the Top-View and Side-View cameras and evaluated on the external camera images.

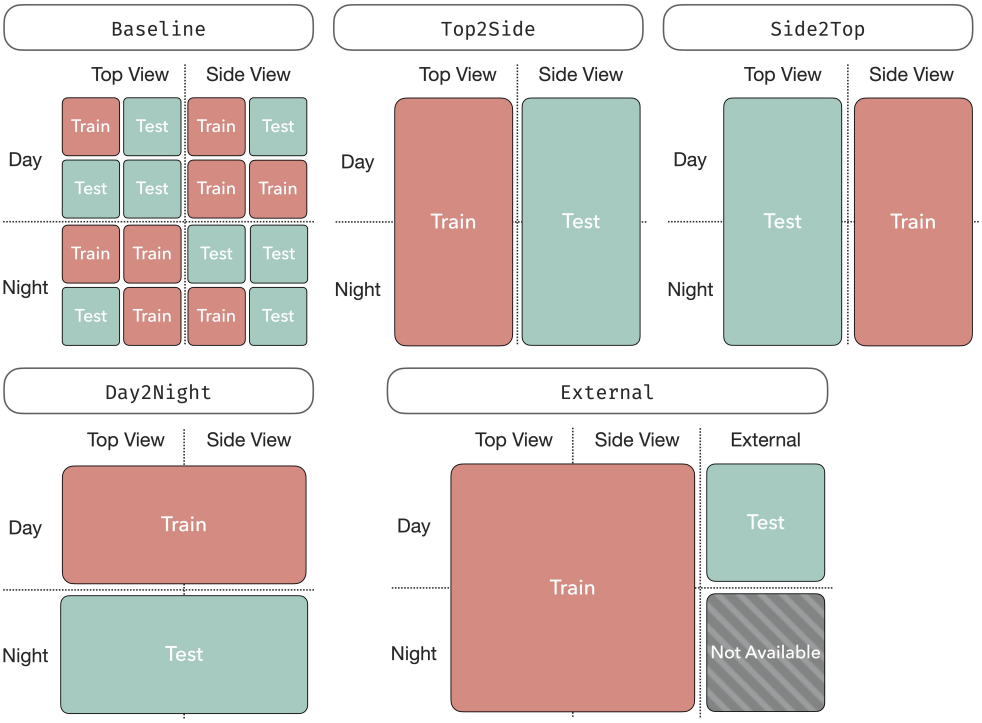

Figure 2: Cross-validation configurations. The training and testing sets were split into five different configurations: Baseline, Top2Side, Side2Top, Day2Night, and External.

To study how the training sample size affects model performance in each configuration, the testing set in the cross-validation was fixed to the same 100 images. Then, the training set size was iteratively altered from 16 to 512 images, with the sample size doubled at each step. Each training sample size was repeated 50 times with different random seeds to ensure the robustness of the results. The YOLOv9e, which is the most capable model in the YOLO family to date according to its performance on the COCO dataset, was used as the base model for this study.

Study 2: The Correlation Between Model Complexity and Performance on the Tasks of Localizing Cows

To investigate whether model performance increases with model complexity, five YOLO-family models were examined in this study. Three of the models were selected from the YOLOv8 family: YOLOv8n, YOLOv8m, and YOLOv8x. All YOLOv8 models share a similar architecture, differing in their depth multiplier, width multiplier, and ratio factor, which collectively determine their parameter counts of 3.2 million (m), 25.9m, and 68.2m, respectively. The depth multiplier determines how many convolutional layers are repeated in a C2F module, the novelty of YOLOv8. The width multiplier and ratio factor collectively specify the channel numbers in the convolutional operations. Correspondingly, YOLOv8n, YOLOv8m, and YOLOv8x are defined by depth multipliers of 0.33, 0.67, and 1.0, respectively. The width multipliers are 0.25, 0.75, and 1.25, while the ratio factors are 2.0, 1.5, and 1.0 [52]. These variations enable the models to achieve different balances between computational efficiency and accuracy.

The remaining two models were YOLOv9c and YOLOv9e, the latest models in the YOLO family, with parameter counts of 25.6M and 58.2M, respectively. Unlike YOLOv8 models, these models have slightly different backbone architectures. Although the majority of the components between YOLOv9c and YOLOv9e are the same, they primarily differ in their layer counts, module complexities, and depth configurations. YOLOv9c has 618 layers and uses simpler modules, resulting in a more efficient model with lower computational demands. Conversely, YOLOv9e has 1225 layers and employs more advanced modules [53].

All models were trained on 500 images in the five cross-validation configurations: Baseline, Top2Side, Side2Top, Day2Night, and External (**Figure 2**). In addition to model performance, computing speed was also evaluated. The training speed was recorded in seconds per 100 epochs on NVIDIA A100 GPUs (NVIDIA, USA), and the inference time was recorded as frames per second (FPS) on both the CPU and GPU (Apple M1 Max chip, Apple Inc., USA). The relationship between model complexity and time consumption was analyzed to provide insights into the trade-off between model performance and computational cost.

Study 3: Assessing the Advantages of Using Fine-Tuned Model Over the Pre-Trained Model as Initial Model Weights

Most models are released with pre-trained weights obtained from large datasets containing millions of object instances (e.g., COCO [18] and ImageNet [41]). The pre-trained models have a general capability in recognizing common objects such as vehicles, animals, and household items. When the model is required to recognize specific objects (i.e., cows in this study), a model trained on a smaller but specific dataset is expected to have better performance. However,

such advantages may not necessarily persist as the training sample size increases. Having an equally large number of samples for both the pre-trained and fine-tuned models could diminish the performance gap between the two models. To investigate this hypothesis, this study evaluated the performance of fine-tuned models with two different initial weights. The first initial weight was the default weight from the pre-trained model on the COCO dataset, while the second initial weight was the weight from the fine-tuned model on the opposite view angle. The cross-validation settings are described in **Table 1**.

Table 1: Finetuning configurations with different initial weights

Finetuning and Prediction Task	**Initial Weights**
Top-View Camera	COCO (pre-trained) Side-View Camera (fine-tuned)
Side-View Camera	COCO (pre-trained) Top-View Camera (fine-tuned)
External Camera	COCO (pre-trained) Top-View and Side-View Cameras (fine-tuned)

The backbones of all models (i.e., YOLOv8n, YOLOv8m, YOLOv8x, YOLOv9c, and YOLOv9e) were fine-tuned across different training sample sizes: 16, 32, 64, 128, 256, and 500. The goal was to determine whether the advantage of using the fine-tuned weights persists under the interaction between model complexity and different fine-tuning samples. The performance of the models was evaluated using mAP@0.5:0.95.

3 Results and Discussion

Public Dataset: COLO

The COLO dataset contains 1254 images and 11818 cow instances captured from an indoor farm environment. The dataset is organized in YOLO and COCO formats and published on the online platforms GitHub (https://github.com/Niche-Squad/COLO/) and Huggingface (https://huggingface.co/datasets/Niche-Squad/COLO). The dataset consists of eight configurations (**Table 2**): *0_all*, *1_top*, *2_side*, *3_external*, *a1_t2s*, *a2_s2t*, *b_light*, and *c_external*. The *0_all* configuration serves as the baseline for this study, featuring non-overlapping training and testing images collected from both the Top-View Camera and Side-View Camera. The *1_top*, *2_side*, and *3_external* configurations contain images from their respective cameras. The *a1_t2s*, *a2_s2t*, and *b_light* configurations include training/testing splits for the Top2Side, Side2Top, and Day2Night scenarios, respectively. The *c_external* configuration features training images from the Top-View and Side-View Cameras, with testing images from the External Camera. The dataset hosted on GitHub is available as a compressed zip file for public access. In contrast, the dataset on Huggingface requires the Python package "datasets" [54] to download. The Huggingface version offers additional functionality to resize the images and annotations to specific resolutions, providing greater flexibility for various applications.

Table 2: Description of the COLO dataset configurations.

Configuration	Training Samples	Testing Samples
0_all	Top-View + Side-View	Top-View + Side-View
1_top	Top-View	Top-View
2_side	Side-View	Side-View
3_external	External	External
a1_t2s	Top-View	Side-View
a2_s2t	Side-View	Top-View
b_light	Day	Night
c_external	Top-View + Side-View	External

Evaluation Metrics

To assess the performance of the YOLO models, we used four key metrics: mAP@0.5:0.95, mAP@0.5, precision, and recall. These metrics provide a comprehensive understanding of how well the models detect and localize cows in the images from the COLO dataset. A pair-wise comparison of these metrics is presented in **Figure 3** to illustrate their interrelationships.

The mAP@0.5:0.95 metric is the most stringent, requiring the model to achieve both high positioning accuracy (i.e., high IoU) and high precision across IoU thresholds from 0.5 to 0.95. Because it is less likely to be influenced by high-confidence predictions alone, it serves as a reliable indicator of overall model performance. Achieving an accuracy greater than 0.90 on this metric is generally unrealistic; typically, a value of 0.7 is considered good and is sufficient to yield precision and recall of around 0.9.

In contrast, mAP@0.5 is more lenient, requiring high confidence but only moderate IoU. It measures the average precision at an IoU threshold of 0.5. For applications where counting cows is more important than precise positioning, an mAP@0.5 value of 0.9 is sufficient. For example, our results showed that the YOLOv8n model, trained on 32 samples, achieved an mAP@0.5 of 0.9, making it suitable for such applications.

Precision and recall metrics focus on the accuracy and completeness of the detections. Precision is the ratio of true positive detections to the total number of positive detections (true positives + false positives), measuring how accurate the positive predictions are. Recall is the ratio of true positive detections to the total number of actual positives (true positives + false negatives), measuring the model's ability to detect all relevant objects. Generally, higher precision is associated with higher recall. However, in some configurations, such as Side2Top and External with smaller sample sizes, models exhibited high recall but low precision. This indicates a tendency to misclassify non-cow objects as cows more frequently than missing actual cows, suggesting a tendency to overestimate rather than underestimate the number of cows in the images.

Our observations emphasize that for applications where counting cows is more critical than precise positioning, achieving a high mAP@0.5 is adequate, while the stringent mAP@0.5:0.95 metric serves as a comprehensive indicator

of overall model performance. These metrics provide insights into both the localization and classification capabilities of the models, helping to identify strengths and weaknesses under different environmental conditions and camera angles.

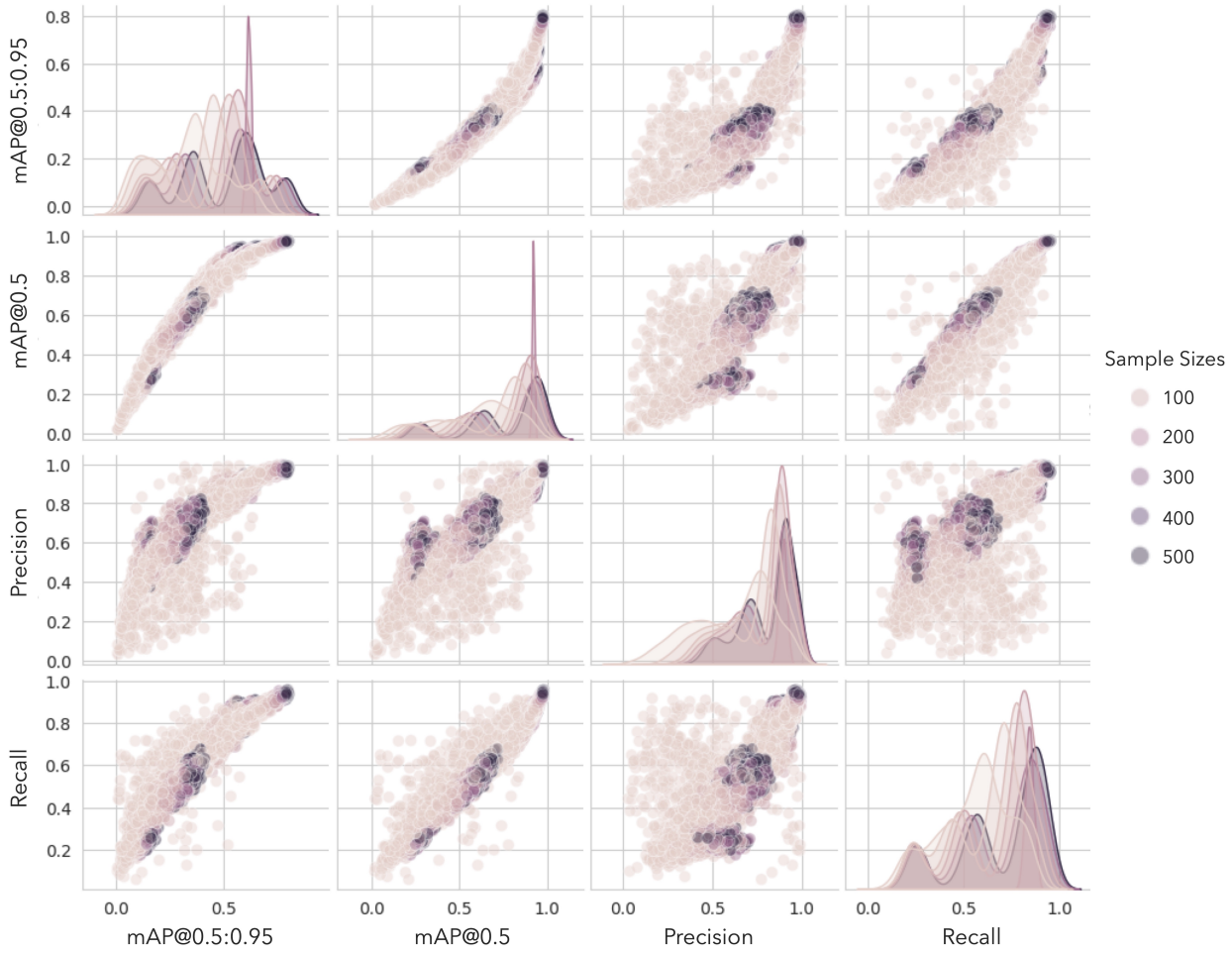

Figure 3: Pairwise scatter plots of the evaluation metrics: mAP@0.5:0.95, mAP@0.5, precision, and recall. Each point represents a different model configuration, with the color indicating the training sample size.

Study 1: The Changes in Camera View Angles Dramatically Affect Model Performance

The baseline training configuration showed good generalization capability, with over 90% of the predictions correctly positioning cows at the 50% IoU criterion (mAP@0.5). The generalization performance can be dissected into changes in view angles (i.e., Top2Side and Side2Top) and lighting conditions (i.e., Day2Night). Changes in lighting conditions did not dramatically affect model performance across all four metrics. However, changing camera views resulted in a performance drop of approximately 30% and 60% in mAP@0.5 for the Side2Top and Top2Side configurations, respectively. Across all metrics and training sample sizes, the Top2Side configuration consistently showed the worst performance.

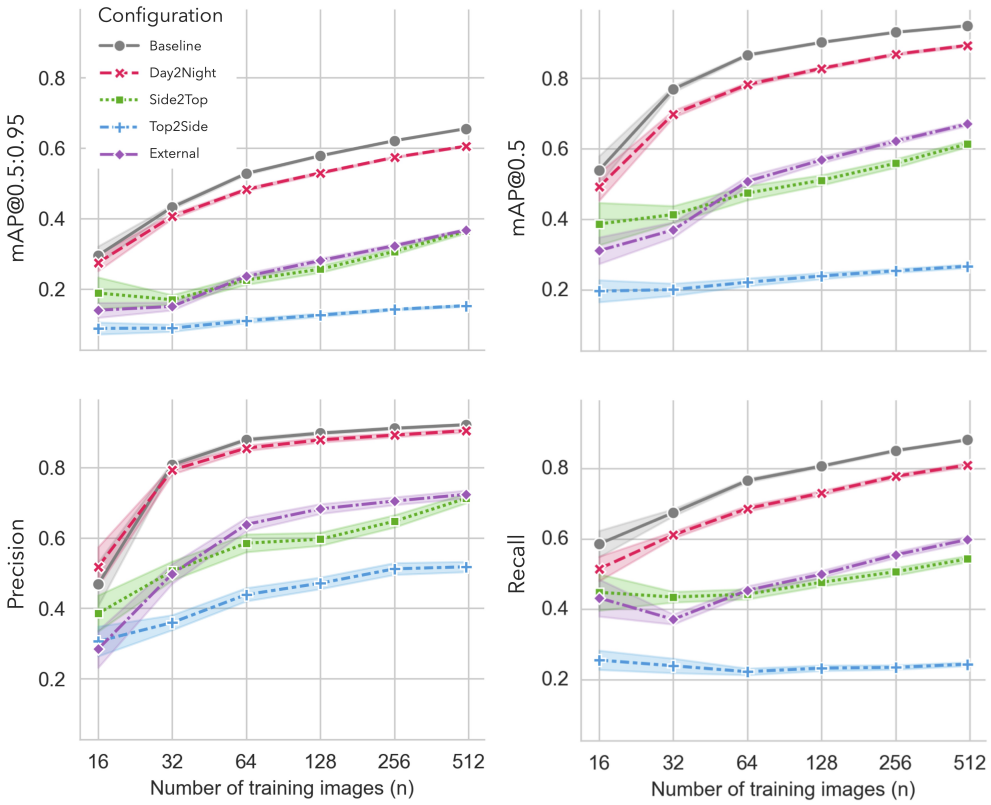

Figure 4: The generalization performance of YOLOv9e across various data configurations and training sample sizes. Sample sizes are depicted on the horizontal axis using a logarithmic scale with a base of 2, and the data configurations are represented by different colors and marker shapes. The upper left and right plots display the metrics mAP@0.5:0.95 and mAP@0.5, respectively, for different training samples across diverse data configurations. The lower left and right plots depict precision and recall values, also for varying training samples and configurations.

From the perspective of precision and recall, changing the camera view from Top2Side resulted in the model missing more than 7 out of every 10 cows, with only 50% of the detections being correct. For the 'External' configuration, our model identified 6 out of every 10 cows, which is not ideal but also not the worst performance observed. Notably, performance in the Day2Night configuration was close to the baseline in terms of precision, which only considers predictions with high confidence compared to mAP@0.5. Hence, by excluding low-confidence predictions, changes in lighting conditions did not affect model performance. Regardless of the configuration and evaluation metrics, model performance always increased as the training sample sizes increased.

This study provides a comparative analysis of the behavior of the model in various data configurations. It is clear that the 'Day2Night' configuration shows much better performance relative to the heterogeneous viewpoint-oriented configurations 'Top2Side' and 'Side2Top'.

Despite the various challenges in adapting models from day to night conditions, the 'Day2Night' configuration consistently maintains high precision, closely mirroring the 'Baseline' configuration across all training sample sizes. This suggests that changes in lighting have less impact on the model's ability to detect objects compared to changes in viewpoint. This robustness to lighting could be attributed to the inclusion of diverse lighting conditions in the training

phase. Specifically, model performance benefited from pixel-wise augmentation techniques such as adjustments to hue, saturation, and value (HSV). These augmentations introduced a variety of color variations to the images, enhancing the model's ability to generalize across different visual conditions. Moreover, these YOLO models benefit from pre-training on the COCO dataset, which is characterized by a wide array of images with varied lighting, aiding their adaptability to shifts in light.

On the other hand, the models perform suboptimally in scenarios involving changes in viewpoint. Each new viewpoint introduces fundamentally different object features that are not replicated through standard data augmentation methods such as lighting or affine transformations. For example, when the camera is placed at a lower angle, cows are more frequently occluded by stalls and fences. These additional objects introduce variations that cannot be mitigated by augmentations in HSV space or image translation. Consequently, 'Top2Side' performs the worst, as it is particularly challenging to identify cows from the side. Even for the 'External' configuration, the model struggles to generalize well despite being trained on the 'Baseline' configuration because the camera angle is changed again in the 'External' setup. In summary, camera view angle is crucial for model generalization, with side views being the most challenging.

Study 2: A Higher Model Complexity Does Not Always Lead to Better Performance

The study found that the training configuration significantly affects the relationship between model complexity and performance. Based on Study 1, predicting images from a side view using a model trained on Top-View camera images is one of the most challenging tasks. In this configuration, increasing model complexity generally resulted in poorer generalization, with simpler models often performing better. However, in other configurations that demonstrated better generalization in Study 1, the peak performance was not always achieved by the most complex model. For example, in the baseline configuration, the YOLOv9e model performed best in terms of mAP@0.5:0.95, mAP@0.5, and recall, while the YOLOv8m model excelled in precision. Neither of these models had the highest parameter counts compared to YOLOv8x. It is also worth noting that different model architectures showed different performance trends with varying complexities. The YOLOv8-family models tended to perform best with mid-sized models (i.e., YOLOv8m), whereas larger models in the YOLOv9 family usually performed better. Hence, the study concluded that model performance is determined by both the training configuration and the model architecture.

The study results, as shown in **Figure 5**, indicate that although both YOLOv8 [31] and YOLOv9 [32] models exhibit an increase in mAP@0.5 with more parameters when trained on the COCO dataset [18], this does not support a definitive conclusion that more parameters consistently improve model performance. This may be because the prior work's findings were based on the COCO dataset, which includes 80 classes and mainly features standalone images. In contrast, this study uses an indoor farm dataset focused exclusively on a single class: cows. Consequently, the model may not need as many parameters to effectively detect cows. This suggests that researchers should not rely solely on public dataset performance, as model generalization is specific to the task and dataset.

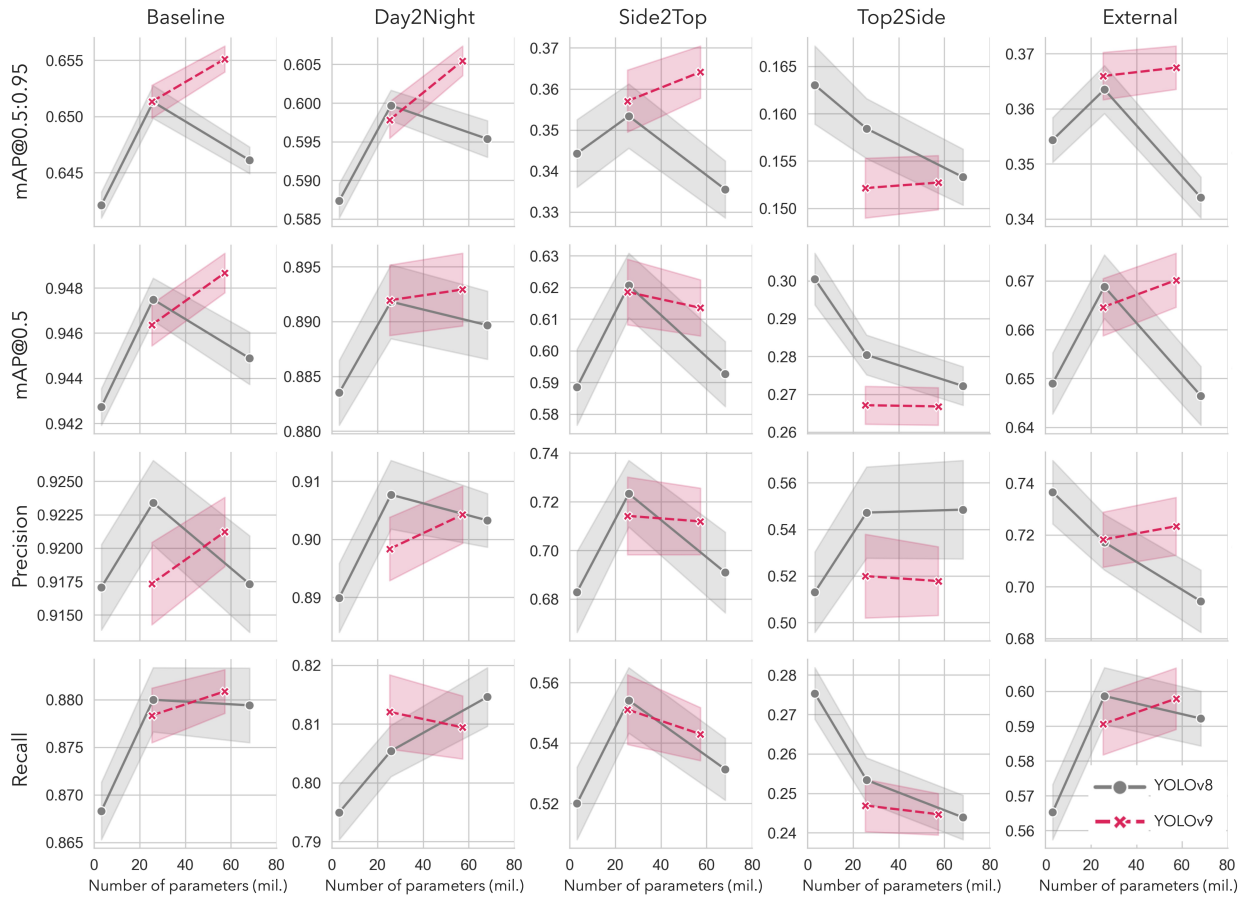

Figure 5: The performance of YOLOv8 and YOLOv9 models across various model parameters and data configurations, evaluated using four metrics: mAP@0.5:0.95, mAP@0.5, precision, and recall. Each column indicates a different data configuration, starting from top left to bottom right: 'Baseline', 'Day2Night', 'Side2Top', 'Top2Side', and 'External'. The horizontal axis of all plots indicates the number of model parameters.

Additionally, this study found that a small model such as YOLOv8n, with only 3.2M parameters, can yield 90% accuracy with a relatively small size of training samples. This indicates that when one encounters a simple and homogenous task like positioning cows, deploying a small model is optimal in balancing computing time and prediction accuracy. This further underscores the importance of considering the specific characteristics of the task and dataset when choosing a model, rather than defaulting to more complex models under the assumption they will perform better.

Overall, our findings emphasize that higher model complexity does not necessarily lead to better performance. The optimal model configuration depends heavily on the specific task and dataset, highlighting the need for careful model selection tailored to the particular application at hand.

Study 3: The Advantages of Custom Initial Weights are Limited When the Model is Simple

The results presented in Figure 6 indicate that the benefit of using fine-tuned initial weights is minimal for simpler models. Specifically, when employing YOLOv8n, the performance difference between the default and fine-tuned

weights was insignificant when fine-tuning data from the Top-View Camera and Side-View Camera. However, as model complexity increased, a greater number of fine-tuning samples were required for the two different initial weights to achieve similar performance. For instance, in the case of YOLOv9e, the performance gap was eliminated when the number of fine-tuning samples reached 128 and 64 for the Top-View Camera and Side-View Camera data sources, respectively. A similar trend was observed with the External camera, where a significant performance gap of more than 25% in mAP@0.5:0.95 was observed for YOLOv9e when the sample size was 16. It is also noted that, although the performance gap was closed to zero for the Top-View Camera and Side-View Camera data sources, the gap was never closed for the External camera.

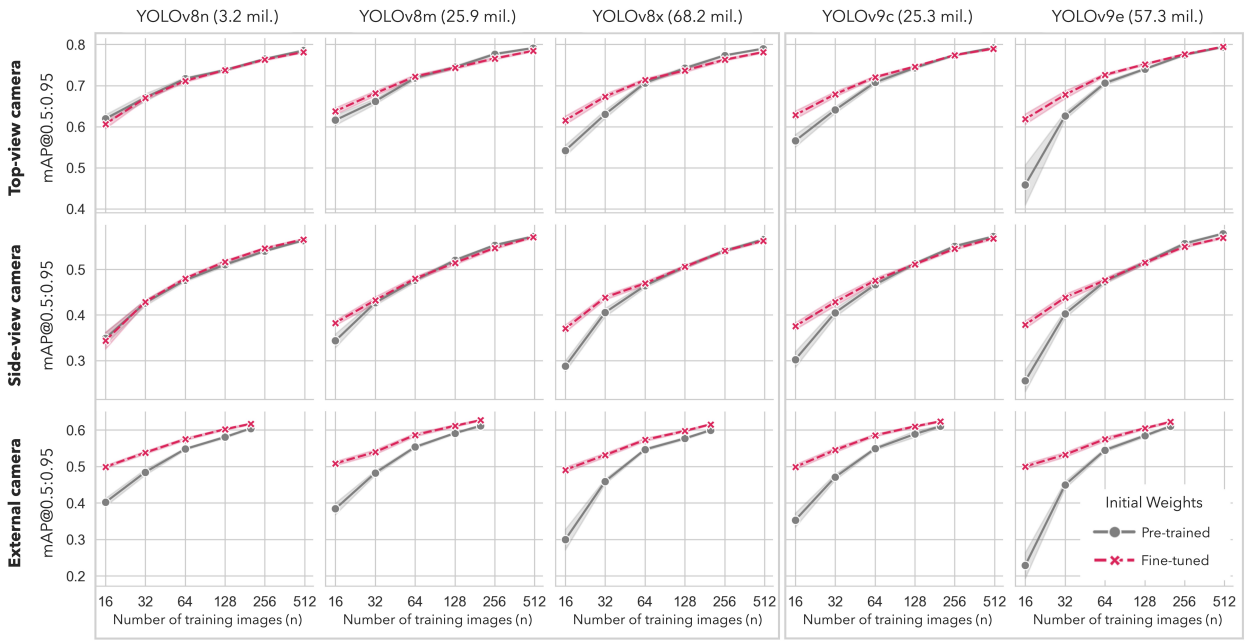

Figure 6: Varied generalization performance in mAP@0.5:0.95 with different initial weights. Red lines represent instances where weights were initialized with fine-tuned weights from other data configuration, while grey lines indicate scenarios employing pre-trained weights (i.e., trained with the COCO dataset). The horizontal axis indicates the number of training samples used for the fine-tuning procedure.

This study suggests that, for YOLO models with fewer parameters, such as YOLOv8n and YOLOv8m, the choice of weight initialization does not make a significant difference in fine-tuning performance. In contrast, larger models like YOLOv8x, YOLOv9c, and YOLOv9e exhibit improved performance when weights are initialized from a model that has been previously fine-tuned in a similar data configuration, as described in Table 1. Therefore, when fine-tuning larger models with a limited dataset, it is beneficial to utilize weights previously fine-tuned on various data configurations. This approach leverages the additional learned features and adaptability from the initial fine-tuning, resulting in better performance even with a small amount of new data. For example, our results showed that YOLOv9e achieved optimal performance with fewer fine-tuning samples when initialized with fine-tuned weights compared to default weights. Conversely, for smaller models, the weight initialization strategy does not significantly impact fine-tuning performance. This is likely due to the lower complexity and fewer parameters of these models, which makes them less dependent on

the initial weight configuration to achieve good performance. In practical terms, this means that for simpler models, researchers can save time and computational resources by directly fine-tuning without the need for customized weight initialization.

The analysis of Figure 6 also provides insight into performance across homogeneous viewpoint data configurations, specifically 'Top-View Camera' and 'Side-View Camera'. The data demonstrates that the 'Top-View Camera' configuration consistently yields higher mAP values regardless of the training sample size and weight initialization conditions. This implies that the 'Side-View Camera' configuration, where both training and test images are captured from the side view, presents a more formidable challenge for cow detection compared to the 'Top-View Camera' configuration. The side view poses difficulties due to occlusions by neighboring cows and additional distractions, such as obstacles in aisles and fences. Furthermore, cows located further away in side-view images may not be as visible, complicating feature extraction. In contrast, the 'Top-View Camera' configuration benefits from an unobstructed aerial perspective, ensuring that the top view of all cow instances is clearly visible and free from such obstructions. This distinction in visibility between the two configurations contributes to the ease of feature extraction and ultimately, the performance disparity observed.

These findings align with the results from Study 1, which demonstrated that changes in camera view angles dramatically affect model performance. In Study 1, we found that models trained on Top-View datasets struggled the most to detect cows from side-view images, with performance dropping by approximately 60% in mAP@0.5. This significant drop in performance is attributed to the same reasons identified in Study 3: the side view introduces occlusions and distractions that are not present in the top view, making feature extraction more challenging.

This study highlights that when working with external or unseen datasets, fine-tuning with custom initial weights trained on relevant tasks brings advantages to the detection tasks. On the other hand, simpler models do not benefit much from customized weights, suggesting that it is more efficient to train a simple model with pre-trained weights without relying on prior relevant information, which sometimes requires intensive labor efforts.

Computational Resource Requirements

The evaluation of computational resource requirements is crucial for understanding the feasibility of deploying YOLO models in real-world applications, especially in environments with limited computational resources. This section compares training time (Figure 7a), inference time (Figure 7b), and model weight sizes (Figure 7c) for various YOLO models.

The training time for each model was measured and expressed as a multiple of the baseline training time, which is the time required to train the YOLOv8n model with 32 samples. The results indicated that using the largest model, YOLOv8x, which has 20 times more parameters, increased training time by 4 to 6 times, depending on the training sample size. Additionally, the YOLOv9 models generally required more training time and had slower inference frames

Figure 7: Comparative evaluation of computational resource requirements. (a) Training time (expressed as a multiple of baseline time) versus number of parameters for YOLOv8 and YOLOv9 models, presented for training sample sizes of 32 (left), 128 (middle), and 500 (right). (b) Inference frequency versus number of parameters for YOLOv8 and YOLOv9 models on GPU (left) and CPU (right). (c) A table displaying the weight sizes and parameter counts of various YOLOv8 and YOLOv9 models.

per second (FPS) compared to the YOLOv8 models. The gap in training time expanded as the number of training samples increased.

Inference time was measured as the average FPS in a batch of 64 images. Running the models on a CPU with the smallest model (YOLOv8n) was slower than running the largest model (YOLOv8x) on a GPU. For example, the FPS for the small, YOLOv8n, on a CPU was 19.77, while the FPS for YOLOv8x on a GPU was 29.21. High FPS models are essential for real-time inference, which usually requires a model with an FPS higher than 30. The results indicate that implementing YOLO models on a CPU may not meet real-time requirements, especially for larger models.

Lastly, model weight sizes were also considered, impacting memory requirements and deployment feasibility, especially in edge computing environments. The weight sizes and parameter counts of various YOLO models are displayed in Figure 7c.

In conclusion, this evaluation highlights the trade-offs between model complexity and computational efficiency. The larger YOLO models, while offering potentially better performance, require significantly more computational resources.

This analysis helps researchers and practitioners select the appropriate model based on the available computational resources and the specific requirements of their application.

4 Conclusion

This study examined the impact of various training configurations and model complexities on the performance of YOLOv8 and YOLOv9 models for cow detection in indoor farm environments. Our results indicate that model performance is highly dependent on camera viewpoints, with side views presenting the greatest challenges. Additionally, fine-tuning models with weights from similar datasets substantially enhances performance, particularly for complex models in scenarios with limited data. We also introduce a public cow localization dataset, 'COLO', to support the research community.

The findings indicate that while increasing model complexity can improve performance, this is not always the case, especially in challenging configurations like 'Top2Side', which predict images from a side view using a model trained on top-view images. Models trained on a single viewpoint exhibit limited generalization, underscoring the importance of incorporating diverse and consistent camera angles in the training data.

Despite the promising results, this study has certain limitations. The models' performance was evaluated under specific indoor farm conditions, which may not generalize to all livestock environments. Moreover, the reliance on pre-defined configurations may limit the applicability of our findings to more dynamic settings.

Future work should explore adaptive methods for enhancing model generalization across varied viewpoints and environmental conditions. Additionally, investigating the integration of advanced data augmentation techniques and more diverse datasets could further improve detection accuracy and robustness.

In conclusion, this study offers practical insights into reproducing model performance in new environmental settings and provides the public 'COLO' dataset to facilitate further research and advancements in the field.

Acknowledgments

This research was supported by the USDA Hatch Research Project funding VA-160196. The authors acknowledge Advanced Research Computing at Virginia Tech for providing computational resources and technical support that have contributed to the results reported within this paper. URL: https://arc.vt.edu/. During the preparation of this work the author(s) used ChatGPT in order to ensure the grammar and clarity of the manuscript. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

References

[1] Arthur Francisco Araújo Fernandes, João Ricardo Rebouças Dórea, and Guilherme Jordão de Magalhães Rosa. Image analysis and computer vision applications in animal sciences: an overview. *Frontiers in Veterinary Science*, 7:551269, 2020.

[2] Sarah Morrone, Corrado Dimauro, Filippo Gambella, and Maria Grazia Cappai. Industry 4.0 and precision livestock farming (plf): an up to date overview across animal productions. *Sensors*, 22(12):4319, 2022.

[3] Wangli Hao, Chao Ren, Meng Han, Li Zhang, Fuzhong Li, and Zhenyu Liu. Cattle body detection based on yolov5-ema for precision livestock farming. *Animals*, 13(22):3535, 2023.

[4] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee, 2001.

[5] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[6] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[7] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016.

[8] Zhenwei Yu, Yuehua Liu, Sufang Yu, Ruixue Wang, Zhanhua Song, Yinfa Yan, Fade Li, Zhonghua Wang, and Fuyang Tian. Automatic detection method of dairy cow feeding behaviour based on yolo improved model and edge computing. *Sensors*, 22(9):3271, 2022.

[9] Danqing Xu and Yiquan Wu. Improved yolo-v3 with densenet for multi-scale remote sensing target detection. *Sensors*, 20(15):4276, 2020.

[10] Abozar Nasirahmadi, Barbara Sturm, Sandra Edwards, Knut-Håkan Jeppsson, Anne-Charlotte Olsson, Simone Müller, and Oliver Hensel. Deep learning and machine vision approaches for posture detection of individual pigs. *Sensors*, 19(17):3738, 2019.

[11] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 447–456, 2015.

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[13] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring r-cnn. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6409–6418, 2019.

[14] Nahian Siddique, Sidike Paheding, Colin P Elkin, and Vijay Devabhaktuni. U-net and its variants for medical image segmentation: A review of theory and applications. *Ieee Access*, 9:82031–82057, 2021.

[15] Su Myat Noe, Thi Thi Zin, Pyke Tin, and Ikuo Kobayashi. Automatic detection and tracking of mounting behavior in cattle using a deep learning-based instance segmentation model. *Int. J. Innov. Comput. Inf. Control*, 18(1):211–220, 2022.

[16] Shuqin Tu, Weijun Yuan, Yun Liang, Fan Wang, and Hua Wan. Automatic detection and segmentation for group-housed pigs based on pigms r-cnn. *Sensors*, 21(9):3251, 2021.

[17] Guoming Li, Yanbo Huang, Zhiqian Chen, Gary D Chesser Jr, Joseph L Purswell, John Linhoss, and Yang Zhao. Practices and applications of convolutional neural network-based computer vision systems in animal farming: A review. *Sensors*, 21(4):1492, 2021.

[18] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[19] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. Pre-trained models: Past, present and future. *AI Open*, 2:225–250, 2021.

[20] Karim Guirguis, Ahmed Hendawy, George Eskandar, Mohamed Abdelsamad, Matthias Kayser, and Jürgen Beyerer. Cfa: Constraint-based finetuning approach for generalized few-shot object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4039–4049, 2022.

[21] Chhaya Gupta, Nasib Singh Gill, Preeti Gulia, and Jyotir Moy Chatterjee. A novel finetuned yolov6 transfer learning model for real-time object detection. *Journal of Real-Time Image Processing*, 20(3):42, 2023.

[22] Thi Thi Zin, Moe Zet Pwint, Pann Thinzar Seint, Shin Thant, Shuhei Misawa, Kosuke Sumi, and Kyohiro Yoshida. Automatic Cow Location Tracking System Using Ear Tag Visual Analysis. *Sensors*, 20(12):3564, January 2020. Number: 12 Publisher: Multidisciplinary Digital Publishing Institute.

[23] Ultralytics. Ultralytics datasets documentation. `https://docs.ultralytics.com/datasets/detect/`, 2023.

[24] Massachusetts Institute of Technology. Labelme: Image annotation tool. `http://labelme.csail.mit.edu/Release3.0/`, 2023.

[25] OpenCV. Cvat: Computer vision annotation tool. `https://www.cvat.ai/`, 2023.

[26] Roboflow. Roboflow: Organize, annotate, and improve machine learning datasets. `https://roboflow.com/`, 2023.

[27] Xia Hu, Lingyang Chu, Jian Pei, Weiqing Liu, and Jiang Bian. Model complexity of deep learning: A survey. *Knowledge and Information Systems*, 63:2585–2619, 2021.

[28] Daniel Justus, John Brennan, Stephen Bonner, and Andrew Stephen McGough. Predicting the computational cost of deep learning models. In *2018 IEEE international conference on big data (Big Data)*, pages 3873–3882. IEEE, 2018.

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[31] Ultralytics. YOLOv8 — docs.ultralytics.com. `https://docs.ultralytics.com/models/yolov8/#overview`, Januray 2023.

[32] Chien-Yao Wang and Hong-Yuan Mark Liao. YOLOv9: Learning what you want to learn using programmable gradient information. 2024.

[33] Glenn Jocher. YOLOv5 by Ultralytics. `https://github.com/ultralytics/yolov5`, 2020. Accessed on: 28 February 2023.

[34] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 107:3–11, 2018.

[35] Sasha Targ, Diogo Almeida, and Kevin Lyman. Resnet in resnet: Generalizing residual architectures. *arXiv preprint arXiv:1603.08029*, 2016.

[36] H. Law and J. Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 734–750, Munich, Germany, 8–14 September 2018.

[37] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6569–6578, Seoul, Republic of Korea, 27 October–2 November 2019.

[38] Z. Tian, C. Shen, H. Chen, and T. He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9627–9636, Seoul, Republic of Korea, 27 October–2 November 2019.

[39] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pages 1–5. IEEE, 2015.

[40] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

[41] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[42] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[43] Simonyan Karen. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv: 1409.1556*, 2014.

[44] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[45] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[46] Kian Eng Ong, Sivaji Retta, Ramarajulu Srinivasan, Shawn Tan, and Jun Liu. Cattleeyeview: A multi-task top-down view cattle dataset for smarter precision livestock farming. In *2023 IEEE International Conference on Visual Communications and Image Processing (VCIP)*, pages 1–5. IEEE, 2023.

[47] Eric T. Psota, Ty Schmidt, Benny Mote, and Lance C. Pérez. Long-term tracking of group-housed livestock using keypoint detection and map estimation for individual animal identification. *Sensors*, 20(13):3670, 2020.

[48] Dataset Ninja. Visualization tools for opencow2020 dataset. `https://datasetninja.com/opencows2020`, may 2024. visited on 2024-05-21.

[49] Colo: A large-scale dataset for conversational question answering over knowledge graphs. `https://huggingface.co/datasets/Niche-Squad/COLO`, 2023. Accessed: 2024-04-09.

[50] Dusty Greif. dgreif/ring, April 2024. original-date: 2018-10-12T22:53:01Z.

[51] Ultralytics. Ultralytics models documentation. `https://docs.ultralytics.com/models/`. Accessed: 2024-05-21.

[52] Ultralytics v8 models. `https://github.com/ultralytics/ultralytics/blob/main/ultralytics/cfg/models/v8/yolov8.yaml`, 2023.

[53] Ultralytics v9 models. `https://github.com/ultralytics/ultralytics/tree/main/ultralytics/cfg/models/v9`, 2023.

[54] Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Marija Čuklina, Simon Brandeis, Teven Le Scao, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Idan Ben-Ami, Olga Filippova, Martin d'Hoffschmidt, Sebastien Gérard, Brendan Lane, Leo Ansell, Lars Buitinck, Damien Esposito, Mathis Raison, Jacob Klein, Thibault Nguyen, Tomoki Mikami, Victor Sanh, Vishrav Chaudhary, Nicolas Patry, Wilson Y. Chang, Julien Froment, Jonas Buhmann, Quentin Malartic, Victor Winschel, Charlie Watson, Rajarshi Pradeep, Gunjan Chhablani, Manuela Rohrbach, Maxim Jenny, John Bolton, Jason Phang,

Theo Löw, Alexander Rush, and Thomas Wolf. Datasets: A community library for natural language processing. https://github.com/huggingface/datasets, 2021.

Appendix

Hyperparameters in Ultralytics library

The table below show the hyperparameters used in the Ultralytics library for training the models in this study.

Table 3: Hyperparameters for the training procedure

Hyperparameters	Description	Value
epochs	Number of training epochs	100
batch	Number of images in each batch	16
optimizer	Optimizer used for training	auto
hsv_h	Altering the hue value of the image	0.015
hsv_s	Altering the saturation of the image by a fraction	0.7
hsv_v	Altering the brightness of the image by a fraction	0.4
translate	Randomly translating the image by a fraction of the image size	0.1
scale	Randomly scaling the image by a fraction of the image size	0.5
fliplr	Randomly flipping the image horizontally with the given probability	0.5
mosaic	Combining four images into one mosaic image with the given probability	1.0
mixup	Randomly mixing up the object instances across multiple images with the given probability	0.15
copy_paste	Randomly copying and pasting the object instances across multiple images with the given probability	0.3