An 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`

May 20, 2024

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Keywords Object detection · Model selection · Model generalization

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

1 Introduction

Define object localization and applications

Localizing livestock individuals from images or videos has became an essential task in precision livestock farming (PLF) []. Such techniques allows researchers and farm managers to monitor the health and well-being of animals in real-time, optimizing their resource management and improving sustainability []. 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, the localization tasks can be further categorized into object detection, object segmentation, and pose estimation. Object detection is the simplest form of localization, which localizes objects of interest by enclosing them within a rectangular bounding box defined by x and y coordinates, pixel width, and pixel height. <examples 1, 2>. To have a finer localization, object segmentation is deployed to outline the object contours pixel-wise, while pose estimation is achieved by orienting and marking the keypoints of the object. <another examples 3, 4>.

Model generalization, pre-training, and fine-tuning

Although implementing image-based systems in the livestock production has shown promising results, current studies merely focus on the accuracy on homogenous environments and rarely address the challenges of model generalization. Model generalization refers to how well a model can perform on unseen data, which is crucial when ones want to reproduce existing studies or models in their own environments. The generalization of a CV model can be affected by a variety of factors in the deployment environment, such as camera angles and the presence of occlusions. Deploying the same model in a new environment with different conditions cannot necessarily guarantee the same performance as reported in the original study. [] Li2021 also pointed out that lightning condition of farms in real applications can be highly variable, leading a poor generalization th new environments.

One explaination for the poor generalization is the discrepancy between the pre-training process and the specific use case. Most CV models are released with pre-trained weights, which were obtained from the results of training on a large-scale dataset. For example, the COCO dataset [] is a general-purpose dataset that contains more than 200 thousands 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 specifically 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 the 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 that is more relevant to the specific use case. Most application studies have adopted this approach to improve the model generalization on their specific tasks (examples 1-5).

Nevertheless, the 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.(2020) deployed an object detection model to recognize cow ear tags in a dairy farm. Although the model achieved a high accuracy of 92.5% on recognizing the digits on the ear tags, more than 10 thousand images were required for fine-tuning the model. Assemblying such a large dataset is labor-intensive and requires specific training in annotating the images. Because the annotated dataset is rigorously organized in specific format. For example, the COCO annotation format [] store the image information, object class, and annotations of the entire dataset in one nested JSON format []. Whereas the YOLO format [], 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 keys to valid annotations, which are usually helped by the professional annotation tools such as labelme [], CVAT [], or Roboflow [].

Model Complexity and Performance

Another perspective that affects the model generalization is model complexity. In general, model complexity is quantified by the number of learnable parameters in a model []. A more complex model often can better generalize to unseen data with higy accuracy. However, such high complexity also comes with a cost of computational resources in a form of either memory or time []. The computational cost may further limit how the 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 simonyan2014very has 138 million parameters and recommends a video memory of at least 8GB, while the ResNet-152 he2016deep has around 60 million parameters with a recommended video memory of 11GB. Additionally, recent models for object detection such as YOLOv8 [] and YOLOv9 [] have been developed in different sizes and therefore provide a flexible choice for researchers to balance between the generalization performance and the computational cost. In YOLOv8, the spectrum of model complexity ranges from the highly intricate, such as YOLOv8x containing 68.2 million parameters, to more streamlined variants YOLOv8n with only 3.2 million parameters. And the demand for the memory, solely from the model architecture without considering the intermediate results during the training or inferenecne process, is larger in a factor of 21 for YOLOv8x (136.9 megabytes) compared to YOLOv8n (6.5 megabytes). Therefore, the trade-off between the model complexity and the computational cost is a critical factor to consider in deploying CV models in real-world scenarios.

YOLO Models

Public Datasets

A public dataset helps the community to develop methodology based on the same baseline. One famous example in CV is the ImageNet dataset [], which serves as a benchmark for image classification. AlexNet [], the winner of the ImageNet Large Scale Visual Recognition Challenge in 2012, show its outstanding capability to classify images in ImageNet dataset using Rectified Linear Units (ReLU) as the activation function than the traditional sigmoid function. The success of AlexNet accelerate the developement of CV models in the following years, such as VGG [], GoogLeNet [], ResNet [], and DenseNet []. However, similar to the challenges that pre-trained models face in the specific use case, a generic public dataset, such as ImageNet and COCO, may not be sufficient to PLF applications. There were efforts to create public datasets for livestock scenario. For example, XXX [] was collected for xxx. Another example in pigs xxx.

Study Objectives

This study aims to explore model generalization across varying environmental settings and model complexities within the context of indoor cow localization. It seeks to exmaine three practical 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.
- **Increasing model complexity always leads to 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 are disminished as the training sample size increasese 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) 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

The studied cows were housed in a free-stall barn at Virginia Tech Dairy Comlex at Kentland Farm in Virginia, USA. The cow handling and image capturing were conducted following the guidelines and approval of the Virginia Tech Institutional Animal Care and Use Committee (#IACUC xxxxx).

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 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 unofficial Ring Application Programming Interface (API) [1], 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 futher 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. The image examples were shown xxx.

The image annotations were conducted using an online platform, Roboflow [], 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 infered based on the adjacent visible parts. If the cow position was too far from the camera and make the important body features, such as head, tail, and legs, unrecognizable, the cow was excluded from the dataset. The final annotations were saved in the YOLO format [], 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 was shown in Figure XXX.

Model Training

The model training was implemented using the python library Ultralytics []. The model hyperparameters were set by 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 include randomly changing the image color hue, saturation, and exposure to improve the model generalizing 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 were shown in 1. 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 boudning boxes. Regardless of specific procedures among YOLO variants for computational efficiency, such as YOLOv8 that integrate objectness scores and conditional class probabilities into a single confidence score, each detection generally consist 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 as a positive detection. Otherwise, the detection is neglected. Combining the localization and classification accuracy, the mean Average Precision (mAP) averaged 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 a 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-testing strategy where models were trained on one dataset configuration and tested on another. There are five training configurations in this study:

- **Baseline:** The model was trained and evaluated on the dataset characterized for all the conditions including top-view, side-view, daylight, evening, and near-infrared images. The images were not overlapped 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.

To study how the training sample size affects the model performance in each configuration, the testing set in the cross-validation was first fixed to the same 100 images. Then, the training set size was altered iteratively from 16 to 512 images with a step size of doubling the sample size. 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 the performance on the COCO datset, 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 the model performance increases with the model complexity, five YOLO-family models were investigated in thie study. Three of the models were selected from the YOLOv8 family: YOLOv8n, YOLOv8m, and

YOLOv8x. All the YOLOv8 models shared the similar model architecture. The differences among the variants are deepen multipler, width multipler, and ratio factor, which collectively determine their parameter counts of 3.2 millions (m), 25.9m, and 68.2m, respectively. The deepen multipler determine how many convolutional layers are repeated in a C2F module, which is the novelty of the YOLOv8. The width multipler and ratio factor collectively specify the channel numbers in the convolutional operations. Correspondingly, the YOLOv8n, YOLOv8m, and YOLOv8x were defined by the deep multipliers of 0.33, 0.67, and 1.0, respectively. The width multipliers, are 0.25, 0.75, and 1.25, while the ratio factor are 2.0, 1.5, and 1.0 []. These variations enable the models to achieve different balances between computational efficiency and accuracy. The remaining two models were YOLOv9c and YOLOv9e, which are the latest models in the YOLO family and are 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 model 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 []. All models were trained on 500 images in each of four configurations described in Study 1.

In addition to the model performance, the computing speed was also evaluated. The training speed was recorded in seconds per 100 epochs, and the inference time was recorded as frames per second (FPS) on both the CPU and GPU (Apple M1 Max chip, Apple Inc.) The relationship between the model complexity and the time consumption was analyzed to provide insights into the trade-off between the model performance and the 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 the pre-trained weights, which were obtained from the large dataset containing millions of object instances (e.g., COCO[] and ImageNet[]). The pre-trained models have 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), having a model trained on a smaller but specific dataset is expected to have a better performance. However, such advantages may not necessarily persist as the training sample size increases. Having equally enough samples for both the pre-trained and fine-tuned models could diminish the performance gap between the two models. To investigate this hypothesis, we conducted a cross-validation between the top-view and side-view images. Two training/testing configurations were considered: the model was trained on the top-view images and tested on the side-view images, and vice versa. Five models, which are YOLOv8n, YOLOv8m, YOLOv8x, YOLOv9c, and YOLOv9e, were trained on 16, 64, 128, 256, and 512 images, respectively. The performance of the models was evaluated using mAP@0.5:0.95.

3 Results

Public Dataset: COLO

The organize dataset is published on the huggingface dataset repository. It's organized in two formats: YOLO and COCO

Study 1: The changes in camera view angles dramatically affect the model performance

The baseline training configuation show a good generalization capabiltiy, in which over 90% of the predctions were correct in positioning cows at the criteria of 50% IoU (mAP@0.5). Further, the generalization performance can be dissected into changes in view angles (i.e., Top2Side and Side2Top) and lighting condition (i.e., Day2Night). The lightinig condition changes did not dramatically affect the model performance in all four metrics, while changing camera views drop the performance by approxmiately 30% and 60% in (mAP@0.5 in the configurations of Side2Top and Top2Side, respectively. Across all the metrics and training sample sizes, the configuration of Top2Side consistently showed the worst performance. From the perspective of precision and recall, changing camera from top view to side view result in the model missing detecting more than 7 cows for ever 10 cows and only 50% of the detection were correct. It is noted that the performance in the Day2Night configuration is closed to the baseline in the metric precision, which only consider predictions with high confidence compared to the metric (mAP@0.5). Hence, by excluding the low-confidence predictions, changing ligthing condition did not affect the model performance. Regardless of the configuration and the evaluation metrics, the model performances always increase as the trianing sampel sizes increase.

Study 2: A higher model complexity does not always lead to better performance

From the study, it is found that the xxx of the training configuration affects the relationship between the model complexity and the performance. Based on the study 1, predicting images from the side view based on the model trained on from the top-view camera is the most challenging tasks. In this configuration, increasing the model complexity, except for few cases, the model generalization always became worse than the simple models are. However, in other configurations that show better generalization in the Study 1, the peak performance did not always found from the most complex model. For example, in the baseline, the model that performed that best is YOLOv9e in metrics of mAP@0.5:0.95, mAP@0.5, and recall. And YOLOv8m performed the best in precision. Neither of the model has the most parameter counts compared to YOLOv8x. It is also worth noting that, different model architecture show different trend in the performance over model complexity. The YOLOv8-family models more likely to have the best performance with the mid-sized model (i.e., YOLOv8m). While in YOLOv9, larger models usually performed better. Hence, the study concluded that the model performacne is determined by both training configuration and the model archtecture.

Study 3: The advantages started to disminish early as the training sample sizes when the model is simple

Validation on the external dataset

4 Discussion

4.1 Why the lighting condition has less impact on the performance drop?

4.2 Is complex model architecture always better?

4.3 How does the model generalization study help in real-world applications?

Acknowledgments

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/

References

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

5 Appendix

Hyperparameters in Ultralytics library

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

Table 1: 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