
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 22, 2024

ABSTRACT

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

10 **Keywords** Object detection · Model selection · Model generalization

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

11 1 Introduction**12 Define object localization and applications**

13 Localizing livestock individuals from images or videos has became an essential task in precision livestock farming
14 (PLF) []. Such techniques allows researchers and farm managers to monitor the health and well-being of animals in
15 real-time, optimizing their resource management and improving sustainability []. Technically speaking, in the field of
16 computer vision (CV), which is a subfield of artificial intelligence (AI) that focuses on translating visual information
17 into actionable insights, the localization tasks can be further categorized into object detection, object segmentation, and
18 pose estimation. Object detection is the simplest form of localization, which localizes objects of interest by enclosing
19 them within a rectangular bounding box defined by x and y coordinates, pixel width, and pixel height. <examples 1,
20 2>. To have a finer localization, object segmentation is deployed to outline the object contours pixel-wise, while pose
21 estimation is achieved by orienting and marking the keypoints of the object. <another examples 3, 4>.

22 Model generalization, pre-training, and fine-tuning

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

31 One explaination for the poor generalization is the discrepancy between the pre-training process and the specific use
32 case. Most CV models are released with pre-trained weights, which were obtained from the results of training on
33 a large-scale dataset. For example, the COCO dataset [] is a general-purpose dataset that contains more than 200
34 thousands images and a wide range of object categories, such as vehicles and household items. Directly deploying
35 a model pre-trained on the COCO dataset to specifically detect cows in a farm setting may not ensure satisfactory
36 performance, as the dataset does not contain enough cow instances in different view angles or occlusions. To alleviate
37 the discrepancy, fine-tuning is a common practice that modifies the prediction head of the pre-trained model and updates
38 the weights on a new dataset that is more relevant to the specific use case. Most application studies have adopted this
39 approach to improve the model generalization on their specific tasks (examples 1-5).

40 Nevertheless, the fine-tuning is not guaranteed to be successful, as the outcome depends on both the quantity and quality
41 of the annotated dataset. For example, zin et. al.(2020) deployed an object detection model to recognize cow ear tags in
42 a dairy farm. Although the model achieved a high accuracy of 92.5% on recognizing the digits on the ear tags, more
43 than 10 thousand images were required for fine-tuning the model. Assembling such a large dataset is labor-intensive

44 and requires specific training in annotating the images. Because the annotated dataset is rigorously organized in specific
45 format. For example, the COCO annotation format [] store the image information, object class, and annotations of
46 the entire dataset in one nested JSON format []. Whereas the YOLO format [], another common format for object
47 localization, stores information of one image in one text file, with each line representing one object instance in the
48 image. Additionally, unlike the COCO format that stores bounding box coordinates in absolute pixel values, the YOLO
49 format stores the coordinates in relative values to the image size. These technical details are keys to valid annotations,
50 which are usually helped by the professional annotation tools such as labelme [], CVAT [], or Roboflow [].

51 **Model Complexity and Performance**

52 Another perspective that affects the model generalization is model complexity. In general, model complexity is
53 quantified by the number of learnable parameters in a model []. A more complex model often can better generalize to
54 unseen data with higy accuracy. However, such high complexity also comes with a cost of computational resources
55 in a form of either memory or time []. The computational cost may further limit how the models can be deployed in
56 real-world applications, where real-time processing or edge computing is desired for fast or compact systems. For
57 instance, the VGG-16 model simonyan2014very has 138 million parameters and recommends a video memory of at
58 least 8GB, while the ResNet-152 he2016deep has around 60 million parameters with a recommended video memory of
59 11GB. Additionally, recent models for object detection such as YOLOv8 [] and YOLOv9 [] have been developed in
60 different sizes and therefore provide a flexible choice for researchers to balance between the generalization performance
61 and the computational cost. In YOLOv8, the spectrum of model complexity ranges from the highly intricate, such as
62 YOLOv8x containing 68.2 million parameters, to more streamlined variants YOLOv8n with only 3.2 million parameters.
63 And the demand for the memory, solely from the model architecture without considering the intermediate results during
64 the training or inferenece process, is larger in a factor of 21 for YOLOv8x (136.9 megabytes) compared to YOLOv8n
65 (6.5 megabytes). Therefore, the trade-off between the model complexity and the computational cost is a critical factor
66 to consider in deploying CV models in real-world scenarios.

67 **YOLO Models**

68 **Public Datasets**

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

77 **Study Objectives**

78 This study aims to explore model generalization across varying environmental settings and model complexities within
79 the context of indoor cow localization. It seeks to examine three practical hypotheses:

- 80 • **Model generalization is equally influenced by changes in lighting conditions and camera angles.** Should
81 camera angles prove more impactful than lighting conditions, it would be advisable to prioritize camera
82 placement when deploying CV models in new environments.
- 83 • **Increasing model complexity always leads to better generalization performance.** If a highly complex
84 model does not ensure superior performance, future studies might consider adopting less computationally
85 demanding models that still enhance performance.
- 86 • **The advantages of using fine-tuned models as initial training weights are persistent over pre-trained
87 models.** If the advantages are diminished as the training sample size increases in a similar cow localization
88 task but different environments, the fine-tuning efforts may be deemed unnecessary when the deployment
89 environment varies over multiple locations on a farm.

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

93 **2 Materials and Methods**

94 **Cow Husbandry**

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

98 **Image Dataset**

99 The images in this study were collected using the Amazon Ring camera model Spotlight Cam Battery Pro (Ring Inc.),
100 which offers a real-time video feed of dairy cows. Three cameras were installed in the barn: two at a height of 3.25
101 meters (10.66 feet) above ground covering an area of 33.04 square meters (355.67 square feet). One camera provided a
102 top view while the other was angled approximately 40 degrees from the horizontal to offer a side view of the cows.
103 These are hereafter referred to as *the top-view camera* and *the side-view camera*, respectively. A third camera, termed
104 *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
105 meters (835.56 square feet). Positioned 10 degrees downward from the horizontal, it captured a challenging perspective
106 prone to occlusions among cows.

107 Images were captured using an unofficial Ring Application Programming Interface (API) [1], configured to record
 108 a ten-second video clip every 30 minutes continuously for 14 days. Since the image quality relies on the camera's
 109 internet connection, which was occasionally unstable, some images were found to be tearing or unrecognizable. Hence,
 110 the resulting dataset was manually curated for consistent quality, comprising 504 images from *the top-view camera*,
 111 500 from *the side-view camera*, and 250 from *the external camera*. These images were further categorized based on
 112 the lighting conditions: for *the top-view camera*, 296 images were captured during daylight, 118 in the evening under
 113 artificial lighting, and 90 as near-infrared images without artificial light. From *the side-view camera*, 113 images were
 114 taken in the evening, and 97 as near-infrared images. All images from *the external camera* were captured during the
 115 day. The image examples were shown Figure 1

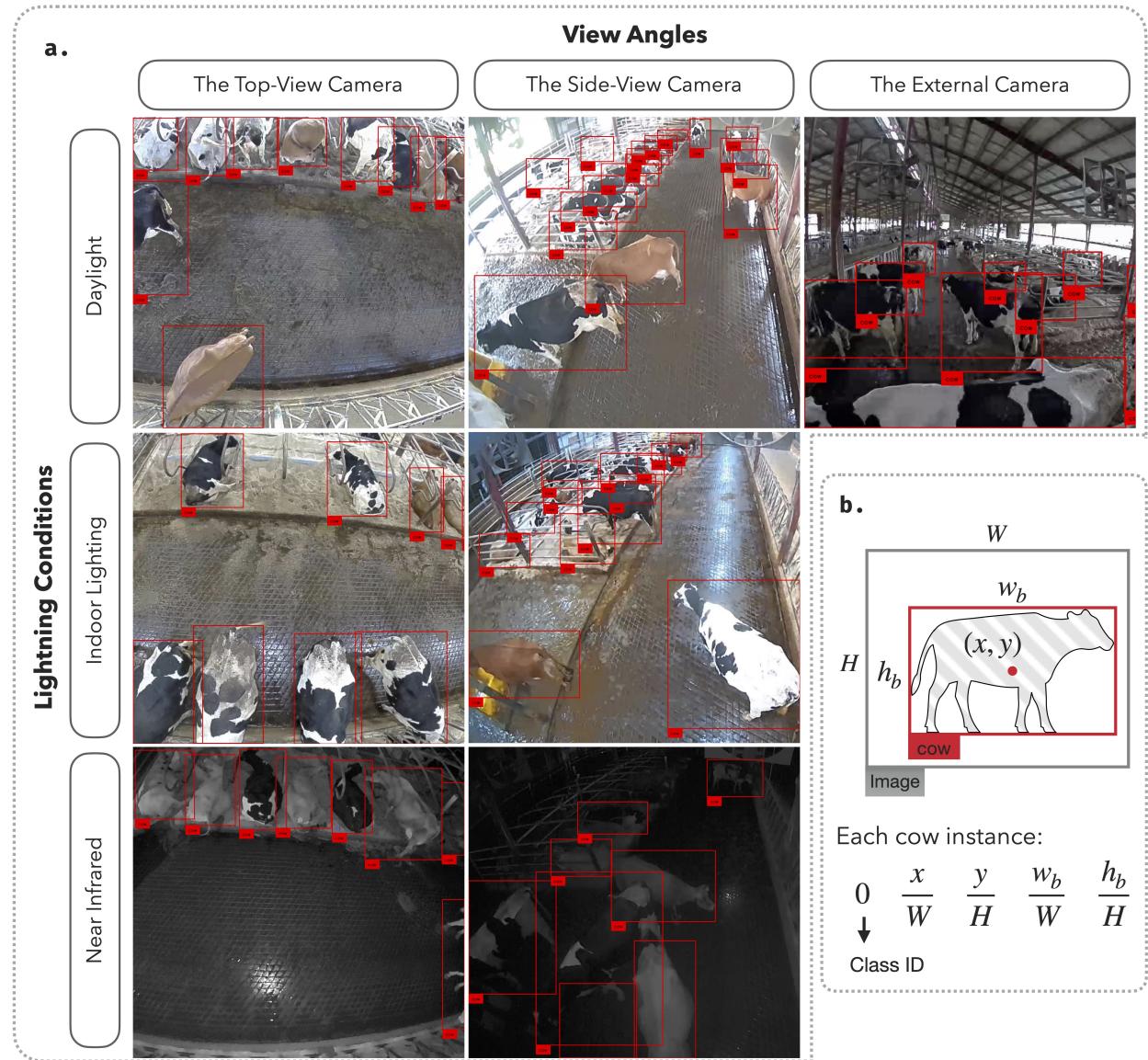


Figure 1: Examples of the annotated images.

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

124 **Model Training**

125 The model training was implemented using the python library Ultralytics []. The model hyperparameters were set by
126 the default values in the library. The training epochs were set to 100, and the batch size was set to 16. The implemented
127 data augmentation include randomly changing the image color hue, saturation, and exposure to improve the model
128 generalizing to different lighting conditions. Geometry augmentation was also applied by randomly flipping the images
129 horizontally, copying and pasting to mix up object instances across multiple images to increase data diversity, and
130 randomly scaling the images to simulate different distances between the camera and the cows. The details of the
131 hyperparameters were shown in 2. The training was conducted on an NVIDIA A100 GPU (NVIDIA, USA) with 80GB
132 video memory provided by Advanced Research Computing at Virginia Tech.

133 **Model Evaluation**

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

148 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
 149 varying IoU thresholds ranging from 0.5 to 0.95 (noted as mAP@0.5:0.95.)

150 **Study 1: Benchmarking Model Generalization Across Different Environmental Conditions**

151 To compare the performance drop between different view angles and lighting conditions, we designed a cross-testing
 152 strategy where models were trained on one dataset configuration and tested on another. There are five training
 153 configurations in this study:

- 154 • **Baseline:** The model was trained and evaluated on the dataset characterized for all the conditions including
 155 top-view, side-view, daylight, evening, and near-infrared images. The images were not overlapped between the
 156 training and evaluation sets.
- 157 • **Top2Side:** The model was trained on the top-view images and evaluated on the side-view images.
- 158 • **Side2Top:** The model was trained on the side-view images and evaluated on the top-view images.
- 159 • **Day2Night:** The model was trained on the daylight images and evaluated on the evening images, including
 160 both artificial lighting and near-infrared images.
- 161 • **External:** The model was trained on images collected by the top-view and side-view cameras and evaluated
 162 on the external camera images.

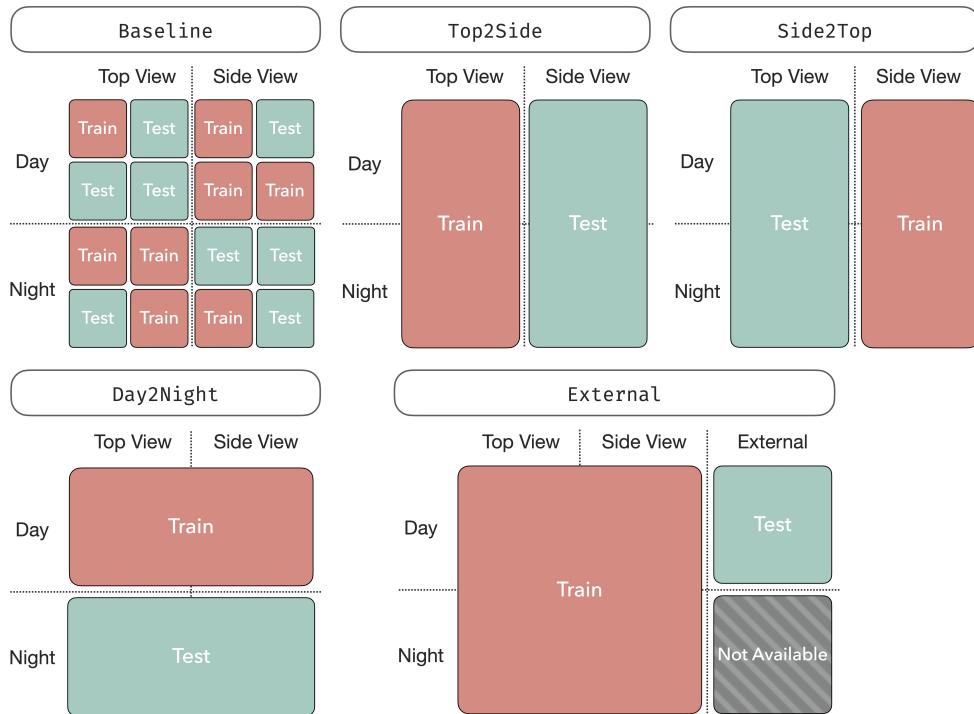


Figure 2: Examples of the annotated images.

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

168 **Study 2: The correlation between model complexity and performance on the tasks of localizing cows**

169 To investigate whether the model performance increases with the model complexity, five YOLO-family models were
170 investigated in thie study. Three of the models were selected from the YOLOv8 family: YOLOv8n, YOLOv8m, and
171 YOLOv8x. All the YOLOv8 models shared the similar model architecture. The differences among the variants are
172 deepen multiplier, width multiplier, and ratio factor, which collectively determine their parameter counts of 3.2 millions
173 (m), 25.9m, and 68.2m, respectively. The deepen multiplier determine how many convolutional layers are repeated
174 in a C2F module, which is the novelty of the YOLOv8. The width multiplier and ratio factor collectively specify the
175 channel numbers in the convolutional operations. Correspondingly, the YOLOv8n, YOLOv8m, and YOLOv8x were
176 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,
177 while the ratio factor are 2.0, 1.5, and 1.0 []. These variations enable the models to achieve different balances between
178 computational efficiency and accuracy. The remaining two models were YOLOv9c and YOLOv9e, which are the latest
179 models in the YOLO family and are with parameter counts of 25.6M and 58.2M, respectively. Unlike YOLOv8 models,
180 these models have slightly different backbone architectures. Although the majority of the model components between
181 YOLOv9c and YOLOv9e are the same, they primarily differ in their layer counts, module complexities, and depth
182 configurations. YOLOv9c has 618 layers and uses simpler modules, resulting in a more efficient model with lower
183 computational demands. Conversely, YOLOv9e has 1225 layers and employs more advanced modules []. All models
184 were trained on 500 images in each of four configurations described in Study 1.

185 In addition to the model performance, the computing speed was also evaluated. The training speed was recorded in
186 seconds per 100 epochs, and the inference time was recorded as frames per second (FPS) on both the CPU and GPU
187 (Apple M1 Max chip, Apple Inc.) The relationship between the model complexity and the time consumption was
188 analyzed to provide insights into the trade-off between the model performance and the computational cost.

189 **Study 3: Assessing the advantages of using fine-tuned model over the pre-trained model as initial model weights**

190 Most models are released with the pre-trained weights, which were obtained from the large dataset containing millions
191 of object instances (e.g., COCO[] and ImageNet[]). The pre-trained models have general capability in recognizing
192 common objects such as vehicles, animals, and household items. When the model is required to recognize specific
193 objects (i.e., cows in this study), having a model trained on a smaller but specific dataset is expected to have a better
194 performance. However, such advantages may not necessarily persist as the training sample size increases. Having
195 equally enough samples for both the pre-trained and fine-tuned models could diminish the performance gap between

196 the two models. To investigate this hypothesis, this study evaluated the performance of the fine-tuned models with
 197 two different initial weights. The first initial weight was the default weights from the pre-trained model on the COCO
 198 dataset, while the second initial weight was the weights from the fine-tuned model on the opposite view angle. The
 199 cross-validation settings were described in the Table 1.

Case	Finetuning Data Sources	Initial Weights
1	Top-View Camera	COCO (default)
2	Top-View Camera	Side-View Camera
3	Side-View Camera	COCO (default)
4	Side-View Camera	Top-View Camera

Table 1: Train and test data configurations with and initial weight criteria.

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

204 3 Results and Discussion

205 Public Dataset: COLO

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

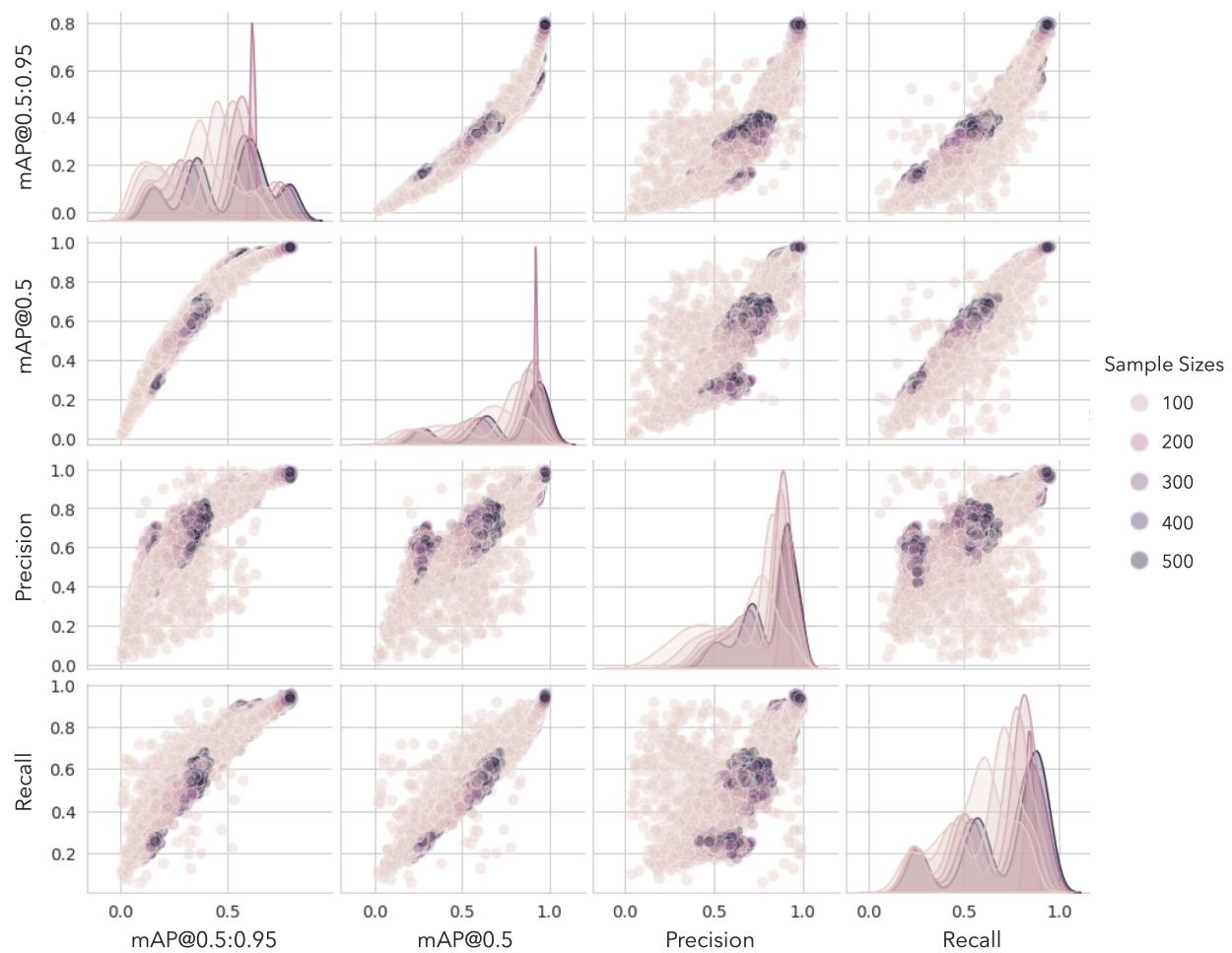
208 **Evaluation Metrics**

Figure 3: Examples of the annotated images.

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

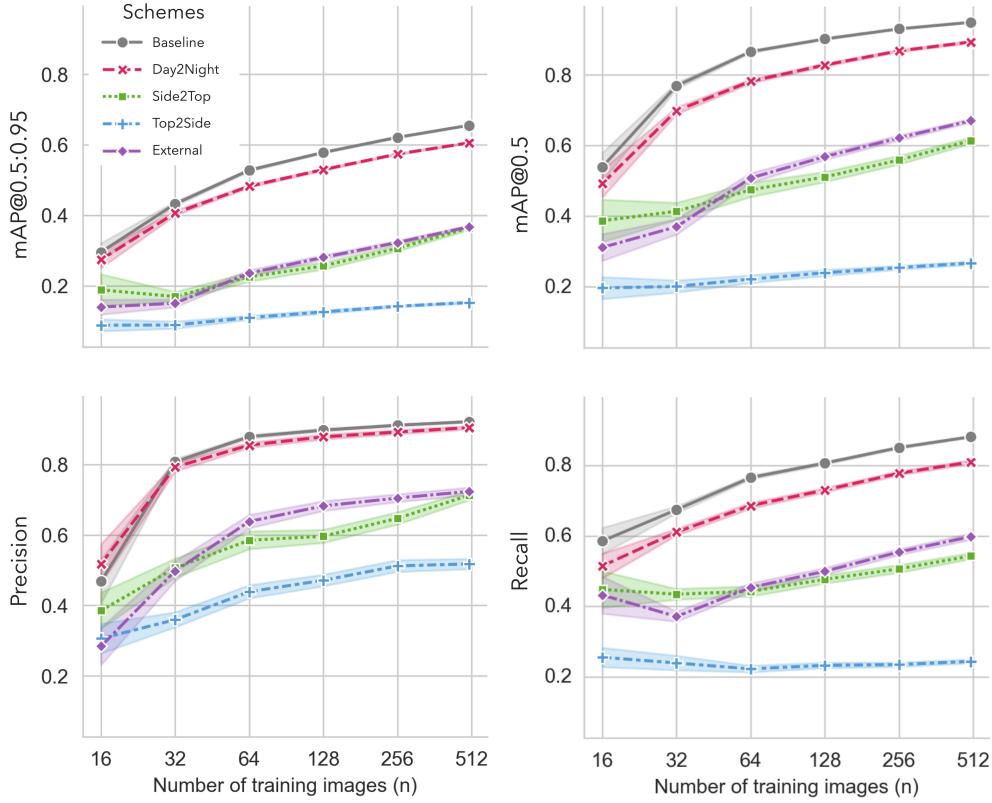


Figure 4: Examples of the annotated images.

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

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

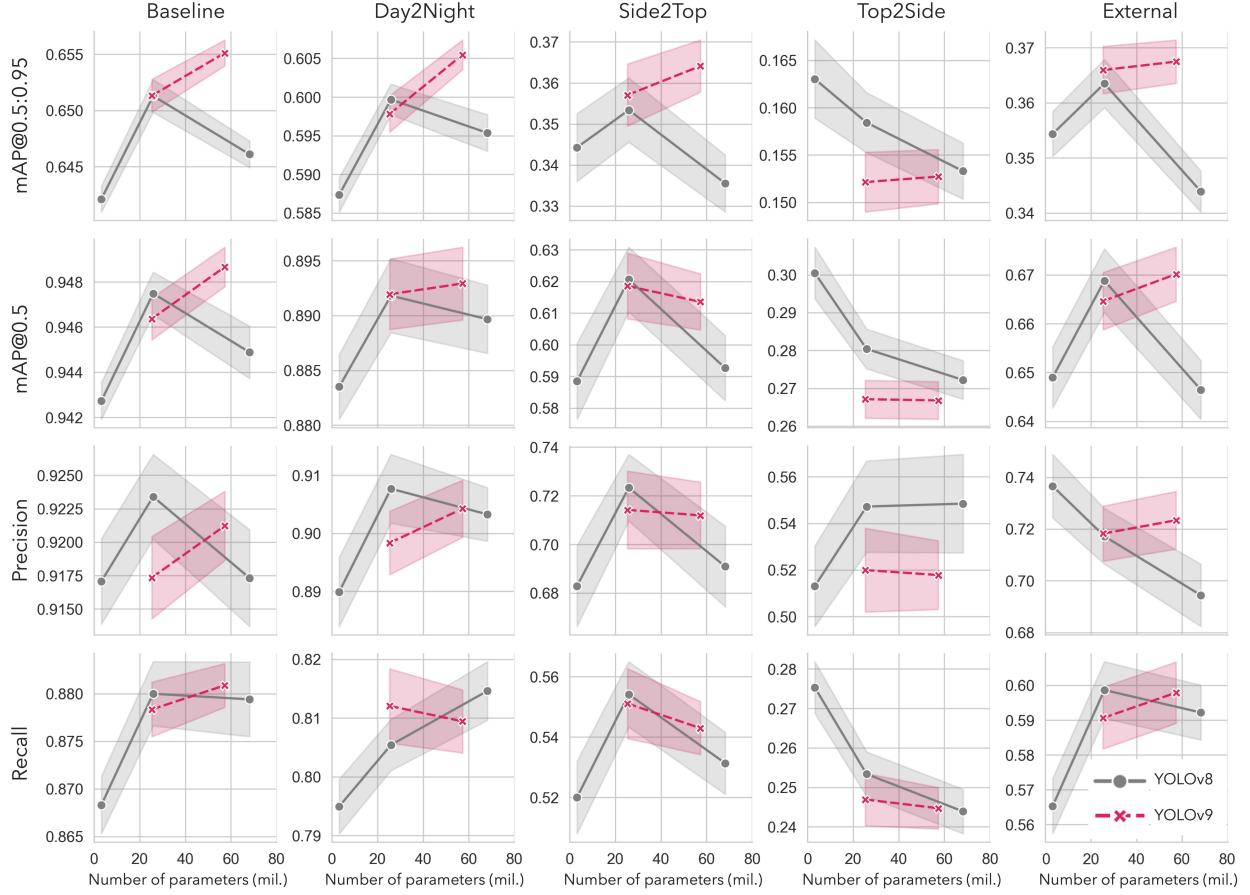


Figure 5: Examples of the annotated images.

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

234 The required computational resources can be evaluated from training time (Figure 6a), inference time (Figure 6b), and
 235 the memory storage sizes (Figure 6c). The training time was presented as a ratio of the actual training time over the
 236 baseline, which is the time required to train the YOLOv8n model with only 32 samples. The results suggested that
 237 using the largest model, YOLOv8x, that with 20 times more parameters, the training time increased by 4 to 6 times
 238 based on the training sample size. Additionally, the YOLOv9 models overall required more training time and slower
 239 inference frames per seconds (FPS) compared to the YOLOv8 models. The gap of the training time was expanded as
 240 the training samples increased. The inference time was calculated as the average FPS in a batch of 64 images. Running
 241 the models on CPU with the smallest model (i.e., YOLOv8n), was found to be slower than running the largest model
 242 (i.e., YOLOv8x) on GPU, where the FPS were 19.77 and 29.21, respectively. The importance of high FPS models was
 243 highlighted in this study, as a real-time inference usually requires a model with FPS higher than 30. Implementing these
 244 YOLO models on CPU may not meet this goal according to the result.

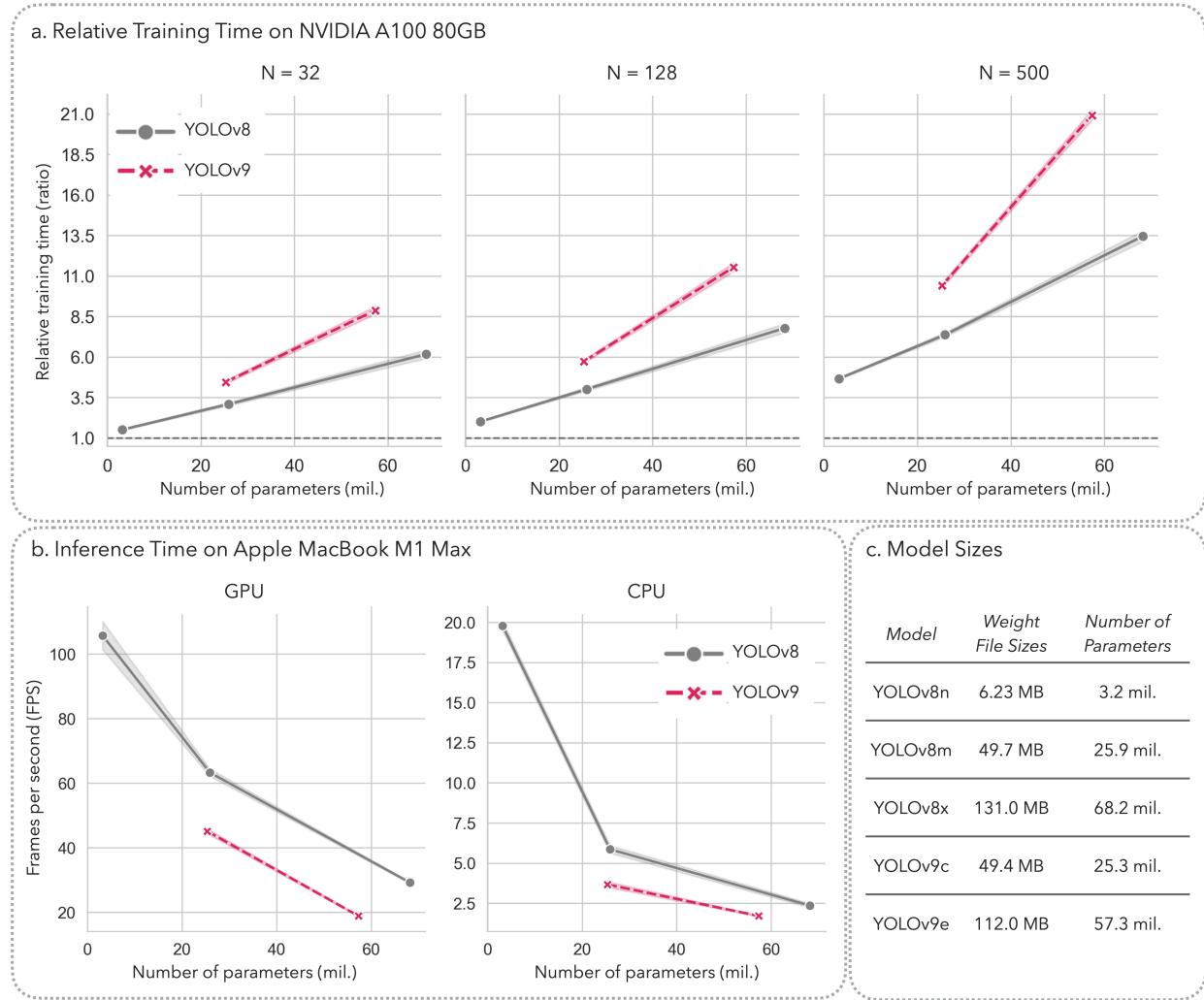


Figure 6: Examples of the annotated images.

245 **Study 3: The advantages of fine-tuning the model is limited when the model is simple**

246 The results presented in Figure 7 indicate that the benefit of using fine-tuned initial weights is minimal for simpler
 247 models. Specifically, when employing YOLOv8n, the performance difference between the default and fine-tuned
 248 weights was insignificant when fine-tuning data from the Top-View Camera and Side-View Camera. However, as the
 249 model complexity increased, a greater number of fine-tuning samples were required for the two different initial weights
 250 to achieve similar performance. For instance, in the case of YOLOv9e, the performance gap was eliminated when
 251 the number of fine-tuning samples reached 128 and 64 for the Top-View Camera and Side-View Camera data sources,
 252 respectively. The similar trend was observed from the External camera, where a significant performance gap of more
 253 than 25% in mAP50:95 was observed for YOLOv9e when the sample size was 16. It is also noted that, although the
 254 performance gap was closed to zero from the Top-View Camera and Side-View Camera data sources, the gap was never
 255 closed for the External camera. In summary, using fine-tuned initial weights is only beneficial for models with larger
 256 parameter sizes (e.g., YOLOv8x, YOLOv9c, and YOLOv9e) and when limited training samples are available.

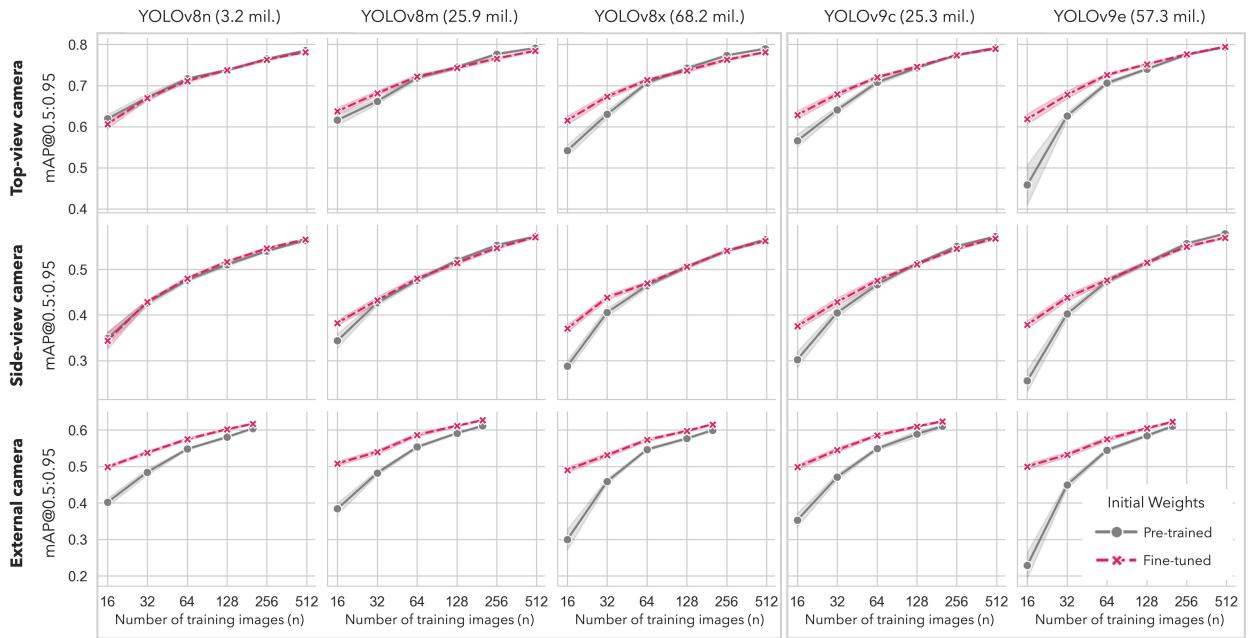


Figure 7: Examples of the annotated images.

257 **Acknowledgments**

258 The authors acknowledge Advanced Research Computing at Virginia Tech for providing computational resources and
 259 technical support that have contributed to the results reported within this paper. URL: <https://arc.vt.edu/>

260 References

- 261 [1] Dusty Greif. dgreif/ring, April 2024. original-date: 2018-10-12T22:53:01Z.

262 4 Appendix

263 Hyperparameters in Ultralytics library

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

Table 2: 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