Comparing Deep Learning Object detection Methods for Real Time Cow Detection

Gichuki Wambui Martha
Computing Department*: School of Computing & Information Technology*
Jomo Kenyatta University of Agriculture & Technology
Nairobi, Kenya
mgichuki@jkuat.ac.ke

Prof Ronald Waweru Mwangi, Ph. D
Computing Department*: School of Computing & Information Technology*
Jomo Kenyatta University of Agriculture & Technology
Nairobi, Kenya
waweru_mwangi@icsit.jkuat.ac.ke

Assoc. Prof Supavadee Aramvith, Ph.D
Department of Electrical Engineering: Faculty of Engineering
Chulalongkorn University
Bangkok, Thailand
supavadee.a@chula.ac.th

Dr. Richard Rimiru Ph. D
Computing Department*: School of Computing & Information Technology*
Jomo Kenyatta University of Agriculture & Technology
Nairobi, Kenya
rimiru@jkuat.ac.ke

Abstract

Deep learning algorithms particularly Convolutional Neural Networks (CNNs) are the state-of-the-art techniques for object detection, classification, segmentation and behaviour classification. These algorithms have extensive application across various domains including agriculture. However, cow identification in dairy farming still relies on methods like direct visual monitoring which are time consuming, costly and inaccurate; or use of invasive contact devices such as sensors which can cause discomfort during attachment or removal. This research compared three deep learning object detection models i.e. YOLOv5, YOLOv7 and YOLOv8, which were selected based on their performance in object detection tasks. We generated cow images from videos captured from a housed dairy cattle barn. The dataset had 11,828 cow images, augmented to depict different illumination conditions and using makesense AI tool, we annotated the images in YOLO format, trained and validated the three models to visualize cow bounding boxes. Our approach demonstrates efficiency of the YOLOv8 model, achieving an accuracy of 94.7% and 93% before and after data augmentation respectively. YOLOv8 baseline model was finetuned using the Ray Tune library achieving a mAP@0.50 score of 92.9%. This research makes a significant contribution in the future research direction of the YOLO algorithm and highlights the practical implementation of deep learning models for cow detection, applicable in livestock management.

Keywords— Convolutional Neural Networks, YOLO, Object Detection

I. Introduction

In computer vision, object detection is a crucial task that entails identifying object instances among image categories. Researchers are diligently working towards achieving optimal accuracy and speed in object detection models and remarkable breakthroughs have been made. Deep learning methods are categorized as one-stage detectors which include You Only Look Once (YOLO) and Single Shot Detectors (SSD) and two-stage detectors such as Region-based Convolutional Neural Networks (RCNN) among others. Fig. 1. Shows some of these methods and according to [20], two-stage detectors outperform one-stage detectors in terms of accuracy but with time trade-off. Generally, YOLO is adopted due to faster inference and not detection accuracy [20].

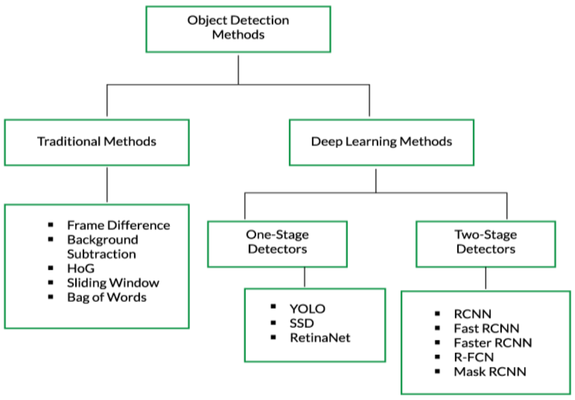


Fig. 1: Object detection methods

1) *Introduction to YOLO*

YOLO algorithm was published in Computer Vision and Pattern Recognition conference (CVPR) 2016 by [14] for real time object detection. According to [17], the algorithm detects objects with a single network pass by combining the two tasks in image analysis into one i.e. object classification and bounding box display [5]. Some of the traditional approaches run several passes using sliding windows followed by classifiers that run multiple times per image. Other advanced approaches divide the object detection task into two tasks: i.e. detecting possible regions with objects (region proposals) and running a classifier on them.

Several computer vision models have utilized YOLO algorithm; these include video surveillance, autonomous driving, animal behaviour, drones and hospital applications as elaborated by [6]. Based on detection speed and accuracy, YOLO algorithm has performed remarkably well compared to other algorithms in real time object detection. YOLO has registered speeds of 45 - 155 Frames per Second (FPS) with fast YOLO doubling the mean Average Precision (mAP) of other algorithms [5], [14].

There are many YOLO variants such as YOLOv1, YOLO9000v2, YOLOv3, YOLOv4, YOLOv5, YOLOv6, YOLOv7 and the relatively new YOLOv8 by the time of writing this paper, which is currently revolutionizing computer vision tasks with incredible features of high inference speed, accurate real time object detections and image segmentation at pixel level.

The main contribution of this paper is a comprehensive comparison of three YOLO variants, namely YOLOv5, YOLOv7 and YOLOv8 in object detection. The three models were selected due to their wide acceptance and superior performance compared to their earlier counterparts According to [13], there are substantial detection improvements offered by YOLOv6 but the variant is not scalable and is hard to train when compared to YOLOv5 and YOLOv7. For this reason, YOLOv6 was excluded in this study. YOLOv7 represents a relatively recent version and as we write this paper, YOLOv8 is the most up-to-date YOLO variant. The small versions of the three variants were chosen for faster processing. The outcome of this research will provide valuable insights into the ongoing research direction of one-stage object detectors particularly YOLO algorithm.

2) Background of YOLO Algorithm

Looking at how YOLO has evolved, it is clear that researchers will continue to refine YOLO architectures. The enhancements have resulted into significant improvements of detection accuracy with some results exceeding those of two-stage detectors [20]. The authors of YOLO [14], [15] and [16] have reframed object detection problem as a regression problem and not a classification problem. Two important CNN features help address the problem of accurate recognition of numerous objects and their precise location in object detection; these are parameter sharing and multiple filters [20]. To detect an object, YOLO algorithm divides the image/frame into grid cells, which predict bounding boxes marked with their position and dimensions, probability of an object in the grid and class probabilities [20]. Anchor boxes are predefined orientations of a 2D box expected from an object.

Fig. 2 shows a simple a 3x3 grid YOLO model with a single class prediction from three classes. This generates an eight-vector value (*pc,bx,by,bh,bw,c1…cm*) where:- *pc* is the probability that the predicted bounding box

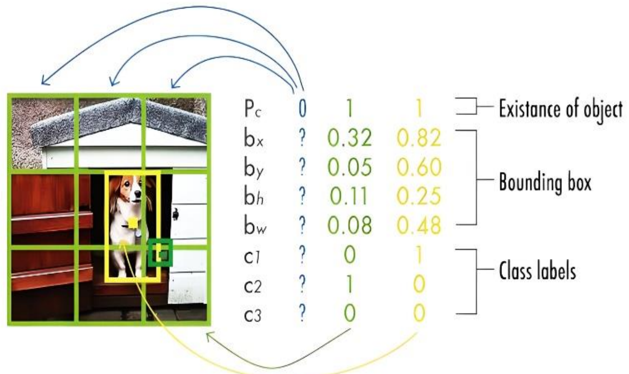


Fig. 2: YOLO output prediction [17]

contains an object belonging to one of the categories $c_1...c_n$; *bx,by, bh and bw* are the predicted bounding box dimensions and $c_1...c_n$ is the *n* different objects in the dataset (e.g. cow, person, dog or house). These can either have a value of zero (no object detected) or one (if an object is detected).

To detect an object using YOLO, one pass on an image is executed to generate bounding box predictions by undertaking the following steps: -

a) Create anchor boxes representing location, shape and size of objects to be predicted

b) Generate a vector output with *pc, bx, by, bh, bw, c1…cn* information for each *SxS* grid cell

c) Remove any predicted bounding box(es) whose *pc* is smaller than a defined threshold (no object present).

d) Perform Non-Max-Suppression (NMS) for each object category "c" in [c_1….c_n], (bounding box with highest *pc*)

3) Intersection over Union (IoU)

A common attribute in detection models is Intersection over Union (IoU), which calculates localization accuracy and error range between ground truth and predictions. Union is the total area covered by the bounding boxes while intersection is their overlap area. To compute the overlap ratio to total area, we divide the intersection by the union as shown in (1), which is an estimated margin between the original bounding box (ground truth) and the predicted bounding box.

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} \quad (1)$$

4) Variants of YOLO Algorithm

YOLO algorithm has evolved through refinements of previous models, to enhance performance by addressing previous limitations. Version enhancements mainly focus on network design changes, modification of loss functions, input resolution scaling and anchor box adaptations among others. Major YOLO variants are YOLOv1, YOLO2, YOLOv3, YOLOv4, YOLOv5, YOLOv6, YOLOv7 and YOLOv8 as summarized in Fig. 3.

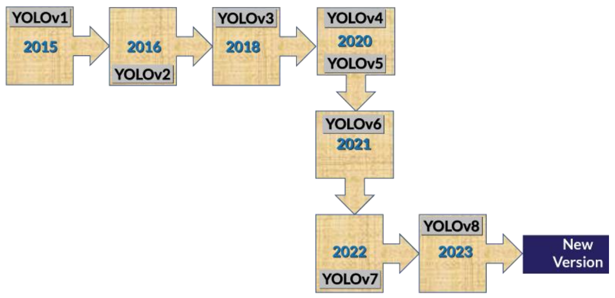


Fig. 3: YOLO Versions and Timelines

YOLOv1 uses softmax function and the architecture was improved to create YOLOv2 by adding a batch normalization layer that achieved higher resolution classification, accuracy and efficiency [20]. Both YOLOv1 and YOLOv2 were trained on PASCAL VOC 2007/2012 datasets, which has 20 object categories with 303-4087 image range per category [10]. YOLOv3 up to YOLOv8 utilizes the Microsoft Common Objects in Context (MS COCO) dataset, created after ImageNet dataset was critiqued for having objects said to be large and well centered [10]. The COCO dataset has 80 object categories with a wide range of scales including small objects and [10] indicates that the dataset may be replaced by other advanced datasets in future.

In their comparative study, [12], indicate that YOLOv3 uses a logistic function to compute class probabilities and Darknet53 to extract features from the input images resulting in better detection. YOLOv3 also uses two-class entropy loss for each category to minimize computational loss of softmax functions. YOLOv4 uses CSPDarknet53 backbone and has reported higher detection accuracy with minimum hardware requirements while compared to YOLOv3.

a) YOLOv5

According to [12], replacing three layers in the YOLOv3 algorithm and adding the focus layer brought about YOLOv5 algorithm. This greatly increased forward and backward propagation while reducing the Compute Unified Device Architecture (CUDA) memory requirements. YOLOv5 was published a month after YOLOv4 was released with high speed and remarkable smaller size compared to YOLOv4 [13]. YOLOv5 was selected in this study due to high performance levels reported in terms of speed and detection accuracy [11], [12], [13], [17], [21]. YOLOv5 model is similar to YOLOv3 and YOLOv4 models which generate three different feature map outputs to achieve multi scale predictions [12]. As shown in Fig. 4, the major improvement in YOLOv5 is the use of the focus structure and CSPDarknet53 as the backbone, thus eliminating the problem of gradient information in YOLOv3 and YOLOv4. This increased the model detection accuracy while reducing the network parameters and inference speed. In a bid to boost information flow and enhance localization accuracy, SPP and PANet were used as the neck in YOLOv5[12]. It also uses YOLOv3 head with Generalized Intersection over Union (GIoU) loss. YOLOv5 has been evaluated on the COCO dataset achieving an AP of 50.7% with an image size of 640 pixels [17].

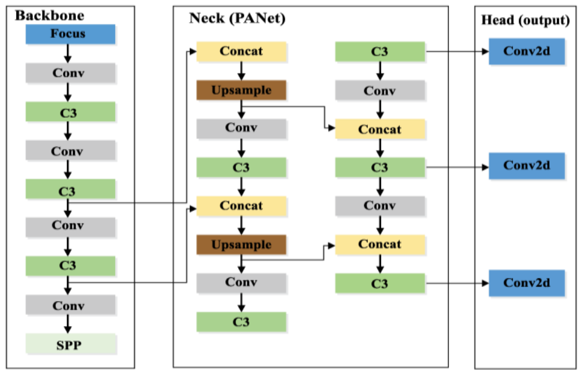


Fig. 4: YOLOv5 Architecture [12]

YOLOv6 uses RepVGG style structure, EfficientRep backbone, Rep-Path Aggregation Network (PAN) anchor free paradigm, Simpler Optimal Transport Assignment (SimOTA) algorithm and Scale-Sensitive IoU (SIoU) bounding box regression loss function.

b) YOLOv7

Without using other datasets or pre-trained weights YOLOv7 is trained entirely on the COCO dataset from scratch [18] as a real-time object detector. The algorithm performs better than previous variants with high accuracy of 56.8% and 30 FPS on Graphics Processing Units (GPU) V100. Like YOLOv5, YOLOv7 uses Extended Efficient Layer Aggregation Networks (E-ELAN) with expansion, shuffling and cardinality merging for better network learning abilities while preserving the initial gradient path [18]. YOLOv7 reduced parameters and computation cost greatly, which improved object detection accuracy and inference speed [13], [18].

c) YOLOv8

This variant was published in January 2023 by the YOLOv5 developers (Ultralytics), as an anchor-free model with a faster NMS and fewer box predictions. YOLOv8 has been evaluated on the COCO dataset achieving an AP of 53.9% for 640-pixel image sizes and 280 FPS speed on NVIDIA A100 and TensorRT. According to [1], YOLOv8 is designed to detect and perform instance segmentation on multiple objects in images and videos [1]. YOLOv8 architecture is similar to that of YOLOv5 consisting of a backbone, head and neck but YOLOv8 uses darknet-53 backbone network which is more accurate and faster compared to YOLOv7 [1]. YOLOv8 has a larger feature map, a feature pyramid network that identifies different object sizes and an enhanced CNN which boosts precision and detection speed [1].

According to [13], YOLOv6 accurately infers single images while YOLOv5 and YOLOv7 infer multiple images. [13] infers that, despite the great detection improvements in YOLOv6, the variant is not easy to train and lacks scalability when compared to YOLOv5 and YOLOv7. Informed by these findings, the experiments in this research were conducted using YOLOv5, YOLOv7 and YOLOv8 to visualize bounding boxes of predicted cow objects using TensorFlow Object Detection API.

II. RELATED WORK

To recognize cow behavior in real time, [5] implemented a YOLOv3 object detection model after gathering video and image data sets using several cameras installed at a livestock farm. To extract more features as required in object detection, they used an extra training layer and utilized the mish activation function with a smoother curve. Their model achieved 97.8% accuracy, 98.5% precision and 97.2% recall values.

A cattle detection and localization Faster R-CNN algorithm was implemented by [3]. Using Unmanned Aerial Vehicles (UAVs) in an indoor barn set up, cows were monitored and video data was gathered. The videos were split into 40-frame streams, with a training/testing ratio of 9:1 and the model achieved a 98.13% detection accuracy.

A Faster R-CNN dairy goats detection model from surveillance videos was implemented by [19]. Foreground segmentation was utilized to reduce static background impact in surveillance videos and to detect incomplete goats such as those standing on the edge. Background subtraction was used to detect moving objects and dimensionality reduction was done through pooling schemes achieving a 92.57% accuracy better than Faster R-CNN. One limitation to their study was insufficient labelled data and they recommend use of transfer learning.

A YOLOv4 fast object detector in production systems to optimize parallel computations was designed by [2], who used a Tesla V100 GPU for the COCO dataset training. They achieved 43.5% (AP50:95) and 65.7% (AP50) results.

A comparison on the performances of YOLOv3, YOLOv4, and YOLOv5l to detect safe landing locations for UAVs that failed while airborne was carried out by [12]. The models were trained on the Dataset for Object deTection in Aerial Images (DOTA) with 11,268 satellite and aerial images captured from Google earth. Their YOLOv5l model performed optimally with a mAP of 63.3% while YOLOv3 and YOLOv4 achieved 60.7% and 46% respectively.

A YOLOv5 model to detect heavy vehicles during winter in real time was developed by [7] and trained on colab. They developed a YOLOv5 notebook using Roboflow.ai trained on pre-trained COCO weights. After about 150 epochs, the model started overfitting and from their results, front cabin components of heavy vehicles were detected better than the rear components.

A prompt method to detect exotic plant seeds is proposed by [21], after building a database of 3000 seed images from 12 invasive plants. To improve the YOLOv5 algorithm, a channel attention mechanism was added and their experiments revealed an improvement in classification and detection accuracy with a slight increase in parameters. Their YOLOv5 Efficient Channel Attention Network (ECA-Net) achieved a 93.96% precision, 90.1% Recall, 82.77% mAP@0.5:0.95 and 91.67% mAP@0.5 prediction results.

A TensorFlow object detection API, utilizing SSD and Faster R-CNN with Inception V2 object detection structure libraries for feature extraction was developed by [4]. The models were trained to predict cow object bounding boxes on image frames achieving detection confidence scores of 90% and 50% respectively and the objects differing in colour were hard to detect as per the results.

A deep learning CNN task assistant model was implemented by [11], using YOLOv5s and YOLOv5m networks to recognize parts of an automobile. In order to access google colab virtual machine Tesla P100 GPU for model training, they used a laptop computer. A total of 582 car engine images were used to demonstrate how YOLOv5 models can detect small car parts with high accuracy in real time video streams.

The YOLOv7 object detection network was added to a Deep SORT tracking algorithm by [22] to develop a visual object tracking model. To evaluate their model, they used several parameters to evaluate the models and YOLOv7 model achieved higher scores compared to YOLOv5.

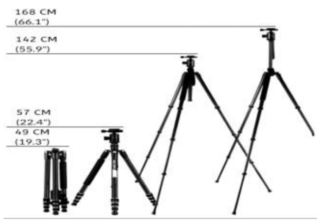
To detect helmet wearing violations in real time from video frames, [1] developed a robust YOLOv8 model with few annotations on their data (few-shot sampling). Their experiments achieved a mAP score of 58.61% on experimental validation data.

To detect small pests early, [8] trained Yolov3, YOLOv3-Tiny, YOLOv4, YOLOv4-Tiny, YOLOv6, and YOLOv8 models using 9,875 pest images taken under different illumination conditions and annotated in YOLO format. YOLOv8 model was executed for real time pest detection in Android application having the best mAP of 84.7%.

III. MATERIALS AND METHODS

1) Data Collection and Description

A dataset containing 5904 cow images was generated from five videos captured using a ground video camera (Canon EOS 750D DSLR) with a focal length of 200mm. A portable tripod monopod (Kingjoy VT-880 2-in-1) whose specifications are shown in Fig. 5 with various height adjustments was used to mount the camera in different positions of the cowshed for clear side views.



Load Capacity	22.05lb/10kg
Maximum Working Height	66.1″ (168cm)
Maximum Height without Center Column	55.9″ (142cm)
Minimum Working Height	22.4″ (57cm)
Folded Length	19.3″ (49cm)

Fig. 5: Camera Tripod Specifications

The observed scene was a small-scale commercial dairy farm (Gracer farm) 25 km North East of Nairobi city between 26th June and 28th July, 2021. There were eighteen (18) dairy cows housed in the roofed cowshed and Fig. 6 shows the barn layout which was approximately 16m by 10m.

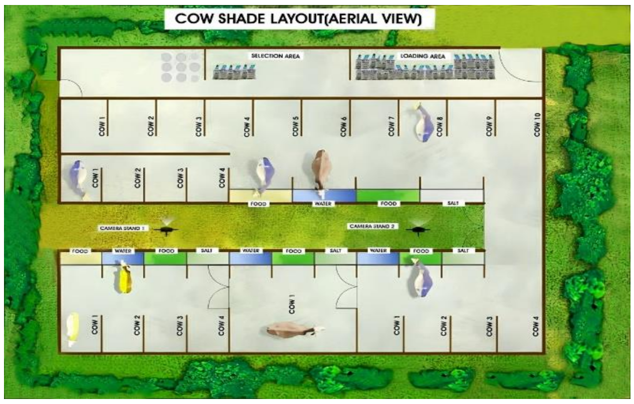


Fig.6: Cow shed layout (Aerial View)

Data augmentation to depict different illumination conditions generated 11,828 images and image resolution was downsized from 1920x1080 pixels to 640x640 pixels for faster processing since high resolution images require more computational resources. The images were used as input to train the models at a ratio of 80:20 for training and validation datasets i.e. 9834 (training images) and 1994(testing images). A 379-frame video stream was used for real time detection.

2) Data Preprocessing and Algorithm Deployment Work flow for Cow Detection

To annotate the images, rectangular shapes of the 11,828 cow images were extracted using the makesense AI tool, and the annotations were exported in YOLO format. Fig. 7 shows an image sample taken from the dataset and the same image after annotation.

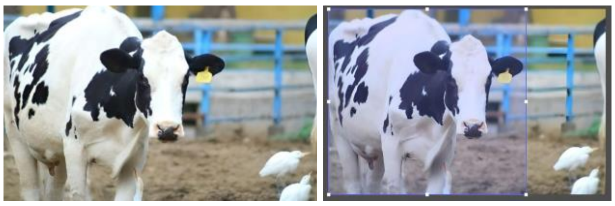


Fig. 7: Sample cow image before and after annotation

Annotated images were uploaded in a google drive folder and python 3.7 was used for platform programming using Google colab notebooks, which mainly offered CUDA Tesla T4 GPU with 15101.8MB video memory. Fig. 8 shows the cow detection work flow from data collection, input, preprocessing to output evaluation.

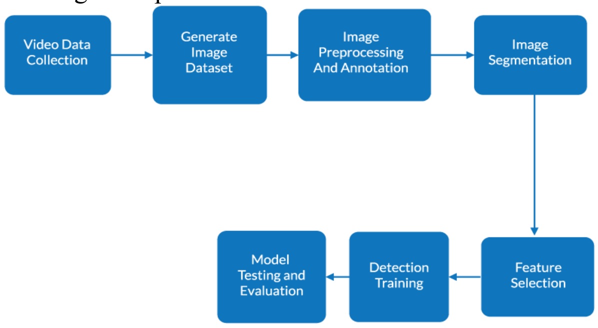


Fig. 8: Cow Detection Workflow

3) *Experiment Setup*

To run the experiments, we accessed Google colab via Google chrome web browser using a HP Notebook Intel(R) Core (TM) i5-7200U CPU @ 2.50GHz 2.70 GHz laptop with 8GB RAM. We used pre-trained COCO weights to train, validate and test the detection models and we saved the training results in Google drive for future use.

a) *Microsoft COCO Dataset*

YOLOv3 up to YOLOv8 models are trained using the COCO dataset, whereby, the AP is computed from multiple IoU values to guarantee detailed model performance evaluation and also from a common AP metric known as AP@50, computed for a single IoU threshold of 0.5[17]. As documented by [17], computing AP in COCO datasets involves:- using different confidence thresholds of the model's class predictions to generate Precision-Recall (PR) curve; calculating the AP thresholds for each of the 80 categories; incrementing IoU levels from 0.50 to 0.95 with 0.05 to compute the *AP*; computing *mAPs* from IoU thresholds across the 80 categories and computing the *general AP* (mean AP values for all IoU thresholds).

b) *Other parameters used*

Optimizers utilize back propagation to minimize loss and we chose Adaptive Moment estimation (Adam) optimizer, which according to [21], is more suited for small datasets. Random neuron drop-out prevents overfitting and a dropout rate of 0.2 of the data was used. A low learning rate with more epochs enables the model to easily reach a minimum point and converge effectively. A learning rate of 0.01 was used to run the models for 50 epochs before and after data augmentation. Warm-up epochs were set at 3, IoU at 0.5, momentum at 0.937 and data loaders at 4. We resized image input size to 640 with a 64-batch input size.

IV. MODEL PERFORMANCE EVALUATION

Popular detection model performance indicators are mAP, Precision, Recall, mAP and FPS which were computed using equations (2), (3) and (4) respectively. After predictions, the objects that are correctly identified are known as True Positives (TP) while negative objects incorrectly identified as positives are False Negatives (FN).

$$Precision = \frac{\text{TP}}{\text{TP + FP}} \tag{2}$$

$$Recall = \frac{\text{TP}}{\text{TP + FN}} \tag{3}$$

$$Accuracy = \frac{\text{TP + TN}}{\text{TP + TN + FP + FN}} \tag{4}$$

The AP value indicates the accuracy in a category and the mAP is a single mean value computed from the AP measures across all categories in the model training. The two metrics were computed using equations (5) and (6),

$$\text{AP} = \int_0^1 P(R)dR \tag{5}$$

$$\text{mAP} = \frac{\sum_{i=1}^{N}(AP)}{N} \tag{6}$$

where P, R represents the precision and recall values respectively while N is the number of classes in the dataset.

1) *Results and Discussion*

Table 1 and 2 shows results for the executions before and after data augmentation respectively.

a) *Precision*

YOLOv8s had the highest TP value of 88.5% and 91.1% for both executions

b) *Recall*

YOLOv5s outperformed YOLOv7-tiny and YOLOv8s with 89.6% and 87.5% recall values respectively for both executions.

c) *Accuracy (mAP@0.5:0.95)*

YOLOv8s outperformed YOLOv5s and YOLOv7-tiny with a mAP@0.5:0.95 of 72.9% and 71.9% respectively.

d) *Accuracy (mAP@0.5)*

YOLOv8s outperformed YOLOv5s and YOLOv7-tiny for the two executions with a mAP of 94.7%, and 93% respectively.

e) *Training and Inference Time*

YOLOv5s outperformed YOLOv7-tiny and YOLOv8s for both training and inference time for the first execution. In the second execution YOLOv8s was the fastest. The variance could be due to the different GPU resources provided by Google colab during training.

2) *Hyperparameter Optimization*

Using Ray Tune library, we fine-tuned our custom YOLOv8s model to align precision and generalization for adaptability to unseen data. A learning rate in the range of 1e-5 to 1e-3 and batch sizes of 8,16, 32 and 64 within 10 different configuration trials were used and evaluated on the dataset. The best configuration emerged at a learning rate of 0.0009920872863737676 and a batch size of 64. Nine trials were terminated and the total running time was 24 minutes 41 seconds. The fine-tuned model's mAP@0.50 score was 92.9% lower than the baseline 93% which could mean that the baseline model was overfitting to the training dataset potentially compromising real-world applicability. Fig. 9 shows partial prediction results after fine-tuning. Fig. 10 and Fig 11 shows partial prediction results from selected variants with YOLOv8s demonstrating image segmentation.

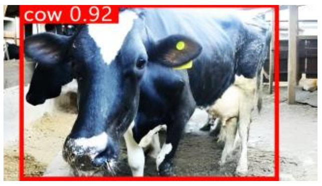


Fig. 9: Partial Prediction results after fine-tuning

TABLE I. PERFORMANCE RESULTS BEFORE DATA AUGMENTATION

Metric	Model		
	YOLOv5s	***YOLOv7-tiny***	***YOLOv8s***
Precision (%)	88.4	89.7	**88.5**
Recall (%)	**89.6**	83.5	88.5
mAP @ 0.5:0.95 (%)	71.9	64.6	**72.9**
mAP @ 0.5 (%)	94.1	91.3	**94.7**
Training time (hours)	**0.317**	4.982	2.734
Inference time (ms)	**6.8**	11.312	12.1

TABLE II: PERFORMANCE RESULTS AFTER DATA AUGMENTATION

Metric	Model		
	YOLOv5s	***YOLOv7-tiny***	***YOLOv8s***
Precision (%)	90.2	84.5	**91.1**
Recall (%)	**87.5**	78.7	86.1
mAP @ 0.5:0.95 (%)	70.5	51.7	**71.9**
mAP @ 0.5 (%)	92.9	85.9	**93**
Training time (hours)	3.470	2.736	**1.921**
Inference time (ms)	10.0	12.181	**9.7**

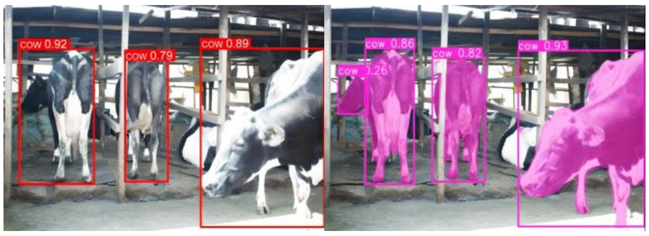


Fig. 10: Prediction and segmentation results of YOLOv8

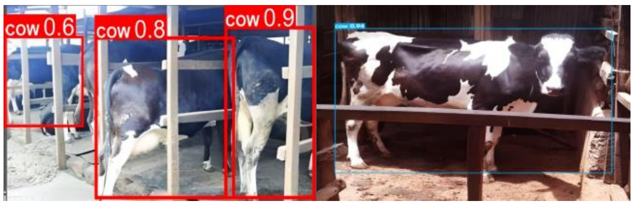


Fig. 11: Partial YOLOv5 and YOLOv7 prediction results

In addition to performing instance segmentation, YOLOv8s demonstrated a better accuracy and precision than YOLOv5s and YOLOv7-tiny. However, YOLOv5s model achieved faster training and inference time during the first execution.

V. Conclusion

This study compared the performance of three YOLO variants using four metrics (precision, recall, mAP and time). The results will benefit researchers seeking to use these models for similar detection tasks using the same metrics. These cow detection models have a promising potential application in assisting farmers manage their small-scale dairy farms. For further research, bigger datasets need to be tested on the fine-tuned model to showcase model performance, robustness and efficiency.

VI. Acknowledgments

This work is supported by JICA Project for AUN/SEED-Net CR-X project titled AI-Based Video Analytics for Smart Farming.

VII. References

[1]. Aboah, A., Wang, B., Bagci, U. & Adu-Gyamfi, Y. (2023). Real-time multi-class helmet violation detection using few shot data sampling techniques and yolov8. arXiv preprint arXiv:2304.08256.

[2]. Alexey, B., Chien-Yao, W., & Hong-Yuan, M. L. (2020). Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934, 2(7).

[3]. Andrew, W., Greatwood, C., & Burghardt, T. (2017). Visual localisation and individual identification of holstein friesian cattle via deep learning. Paper presented at the Proceedings of the IEEE International Conference on Computer Vision Workshops.

[4]. Arago, N. M., Alvarez, C. I., Mabale, A. G., Legista, C. G., Repiso, N. E., Robles, R. R. A., . . . Velasco, J. (2020). Automated estrus detection for dairy cattle through neural networks and bounding box corner analysis. Int. J. Adv. Comput. Sci. Appl, 11, 303-311.

[5]. Chae, J.-w., & Cho, H.-c. (2021). Identifying the mating posture of cattle using deep learning-based object detection with networks of various settings. Journal of Electrical Engineering & Technology, 16(3), 1685-1692.

[6]. Górriz, J. M., Ramírez, J., Ortíz, A., Martínez-Murcia, F. J., Segovia, F., Suckling, J. & Ferrández, J. M. (2020). Artificial intelligence within the interplay between natural and artificial computation: Advances in data science, trends and applications. Neurocomputing, 410, 237-270.

[7]. Kasper-Eulaers, M., Hahn, N., Berger, S., Sebulonsen, T., Myrland, Ø., & Kummervold, P. E. (2021). Detecting heavy goods vehicles in rest areas in winter conditions using YOLOv5. Algorithms, 14(4), 114.

[8]. Khalid, S., Oqaibi, H. M., Aqib, M., & Hafeez, Y. (2023). Small Pests Detection in Field Crops Using Deep Learning Object Detection. Sustainability, 15(8), 6815.

[9]. Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., ... & Wei, X. (2022). YOLOv6: A single-stage object detection framework for industrial applications. arXiv preprint arXiv:2209.02976.

[10]. Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., & Pietikäinen, M. (2020). Deep learning for generic object detection: A survey. International journal of computer vision, 128, 261-318.

[11]. Malta, A., Mendes, M., & Farinha, T. (2021). Augmented reality maintenance assistant using yolov5. Applied Sciences, 11(11), 4758.

[12]. Nepal, U., & Eslamiat, H. (2022). Comparing YOLOv3, YOLOv4 and YOLOv5 for autonomous landing spot detection in faulty UAVs. Sensors, 22(2), 464.

[13]. Olorunshola, O. E., Irhebhude, M. E., & Evwiekpaefe, A. E. (2023). A Comparative Study of YOLOv5 and YOLOv7 Object Detection Algorithms. Journal of Computing and Social Informatics, 2(1), 1-12.

[14]. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

[15]. Redmon, J., & Farhadi, A. (2017). YOLO9000: better, faster, stronger. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7263-7271).

[16]. Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804. 02767.

[17]. Terven, J., & Cordova-Esparza, D. (2023). A Comprehensive Review of YOLO: From YOLOv1 to YOLOv8 and Beyond. arXiv preprint arXiv:2304.00501.

[18]. Wang, C. Y., Bochkovskiy, A., & Liao, H. Y. M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXiv preprint arXiv:2207.02696.

[19]. Wang, D., Tang, J., Zhu, W., Li, H., Xin, J., & He, D. (2018). Dairy goat detection based on Faster R-CNN from surveillance video. Computers and Electronics in Agriculture, 154, 443-449.

[20]. Diwan, T., Anirudh, G., & Tembhurne, J. V. (2023). Object detection using YOLO: Challenges, architectural successors, datasets and applications. Multimedia Tools and Applications, 82(6), 9243-9275.

[21]. Yang, L., Yan, J., Li, H., Cao, X., Ge, B., Qi, Z., & Yan, X. (2022). Real-time classification of invasive plant seeds based on improved YOLOv5 with attention Mechanism. Diversity, 14(4), 254.

[22]. Yang, F., Zhang, X., & Liu, B. (2022). Video object tracking based on YOLOv7 and DeepSORT. arXiv preprint arXiv:2207.12202.