Food Object Detection Using Custom-Trained YOLOv8 with Roboflow Integration

Rajesh Kumar S.
Department of Computer Science and Engineering, Cambridge Institute of Technology, Bengaluru, India | Visvesvaraya Technological University, Belagavi, Karnataka, India
rajeshkumar.d1405@gmail.com (corresponding author)

Josephine Prem Kumar
Department of Computer Science and Engineering, Cambridge Institute of Technology, Bengaluru, India | Visvesvaraya Technological University, Belagavi, Karnataka, India
d_prem_k@yahoo.com

Received: 15 September 2025 | Revised: 6 October 2025 and 27 October 2025 | Accepted: 29 October 2025

 DOI: https://doi.org/10.48084/etasr.14810

ABSTRACT

This study presents a complete, real-time food detection pipeline integrating the YOLOv8 object detection framework and the Roboflow dataset management platform. A dataset with a total of 1,800 high resolution images of six food categories (egg, chicken, carrot, apple, bread, and burger) was developed by using Roboflow for annotation and augmentation, and it was trained through YOLOv8 in a PyTorch (Ultralytics) framework. Strong performance was reflected in experimental results, with a mean Average Precision (mAP@0.5) of above 92% and real-time processing speed of more than 25 FPS on NVIDIA T4 GPUs. Improvements of 5% in precision and 7% in recall were observed for the proposed approach compared to SSD and YOLOv3 baselines. Thus, the findings validated that the YOLOv8-Roboflow pipeline is a flexible and effective option for modern food detection applications in health monitoring and smart kitchen systems.

Keywords-PyTorch (Ultralytics) framework; deep learning; food object detection; health applications; real-time detection; Roboflow; YOLOv8

I. INTRODUCTION

The evolution of intelligent food monitoring systems can be attributed to the combined progress of computer vision and nutrition science, which led to the automation of dietary assessment and caloric computation through advanced object detection techniques. The You Only Look Once (YOLO) algorithm stands out for its real-time performance, while tools like Roboflow support data collection and the development of Machine Learning (ML) pipeline. However, issues, involving high intra-class variability, occlusion, fluctuating lighting, and visually similar items, complicate the detection of accurate food.

The traditional manual food logging methods are slow and prone to errors, delaying effective health monitoring. Previous research has already investigated several deep learning-based object detection models, such as Faster R-CNN, SSD, and earlier YOLO versions [1, 2]. The YOLOv8 architecture is recognized for its speed and accuracy, making it suitable for applications where rapid visual recognition is necessary [3]. Nevertheless, the success of these models is significantly influenced by the quality and quantity of the training data. This is where Roboflow contributes, by streamlining dataset annotation, preprocessing, and exportation into YOLO-compatible formats, thereby closing the gap between dataset preparation and model training [4]. This study proposes a full pipeline that integrates the YOLOv8 object detection algorithm and Roboflow's dataset management platform, in order to detect and classify different food items including egg, chicken, carrot, apple, bread, and burger. The proposed system is designed to operate effectively in real-time settings and across various imaging conditions. The novelty of this research lies in its comprehensive incorporation of YOLOv8 and Roboflow. While both tools have been applied in prior studies, this work demonstrates a practical, low-latency framework that is not only accurate but also replicable and extensible for real-world applications setting a foundation for future enhancements in dietary informatics and healthcare technology.

II. LITERATURE REVIEW

Authors in [5], applied YOLOv5, ResNet50, and VGG16 models to detect cotton plant leaf diseases from garden images, with an accuracy of 98.51%. Their method outperformed previous ones, but specific disease types could not be detected. In [6], a drone-based YOLOv5 and Deep SORT solution was proposed for counting tomato fruits and flowers. The accuracy for red tomatoes reached 85% but the performance for flowers was lower. In fruit detection, YOLOv3 and YOLOv5 were compared for apple recognition, with YOLOv5 reducing false

positive and false negative rates [7]. A similar application was utilized in [8], with YOLOv5 used to distinguish toxic mushrooms. A mAP value of 0.77 was calculated, offering a public safety tool with generalizability concerns. Authors in [9] used YOLOv8-CSP and Deep SORT to count pears in real time on mobile devices, achieving over 98% accuracy, though detection flickering and lighting variability posed limitations. In [10], the optimization of YOLOv8 and YOLOv8-tiny was conducted with image tiling for loose fruit detection in oil palm plantations. Their integration with a robotic system indicated a boost in precision up to 92.6%, albeit with potential computational overhead. Additionally, in [11], an improved version of YOLOv8 was evaluated, with dilated convolutions and focal loss to enhance small object detection, offering up to 16% performance gain. However, the computational complexity increased. Applications of YOLO in flower [12] and fish detection also exhibited encouraging results. YOLOv8-tiny and ZED 3D data enabled real-time flower localization. For marine biodiversity, YOLOv9 surpassed earlier versions in fish classification [13]. Moreover, research into real-time field detection, such as YOLO-Tomato, improved YOLOv5 for apple graspability, rice seedling counting via YOLOv8/9/10 and citrus, kiwifruit and strawberry detection further validated the capabilities of YOLO variants across diverse domains and use cases [14-19].

III. PROPOSED METHODOLOGY

A combination of the YOLOv8 model and the Roboflow platform was developed to construct a strong deep learning pipeline for finding multiple types of food items, including eggs, chicken, carrots, apples, bread, and burgers.

A. System Design Overview

The proposed design used the Roboflow platform to make data preparation easier and connected it to the YOLOv8 detection algorithm, which run on the PyTorch (Ultralytics) framework. Figure 1 illustrates the overall system architecture, including the steps from getting images and adding annotations to training the model and receiving output from the inference. Initially, a selection of food images was collected, and then Roboflow was used to create tags for them. The YOLOv8 model was subsequently trained on the annotated dataset using the PyTorch (Ultralytics) framework. It learns how to find food and put it into groups using bounding boxes and confidence scores. After that, the model was tested on a different validation set to see if it can handle food photos it has not seen before.

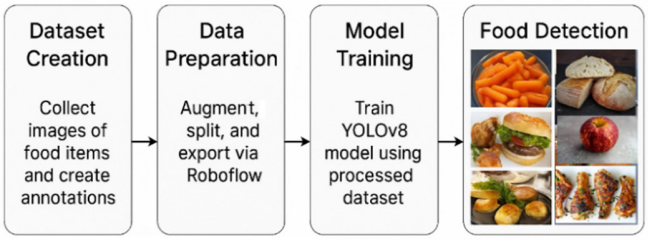

Fig. 1. Workflow diagram for YOLOv8-Based food detection with Roboflow integration.

B. Dataset Preparation and Annotation

In this research, the dataset consisted of 1,800 high-resolution food pictures that were sorted into six distinct groups: egg, chicken, carrot, apple, bread, and burger, with each of them containing 300 images [20]. Image annotations were performed using the Roboflow annotation tool, where bounding boxes with class labels were drawn around each food item. Roboflow was also used for automatic image orientation normalization, resizing all images to 416 × 416 pixels to comply with the YOLO input requirements, as well as data augmentation including random rotations, contrast, and brightness adjustments. The dataset was then exported in the YOLOv8 PyTorch (Ultralytics) format, including images/folder, labels/folder, data.yaml, and train.txt files, and was divided into three subsets:

- Training: 70% (1,260 images)
- Validation: 15% (270 images)
- Test: 15% (270 images)

C. YOLOv8 Detection Mechanism

YOLOv8 is a one-stage object detector that formulates detection as a regression task to automatically generate bounding boxes and class probabilities from full images. The model splits the input image into an S × S grid, and every grid cell predicts B bounding boxes and their corresponding class probabilities. Each predicted bounding box consists of center coordinates (x, y), width and height: (w, h), objectness score (P_{obj}), and class probabilities: $P(\text{Class}_i \mid \text{Object})$. The total confidence score is given by:

$$Confidence = P_{obj} \times P(Class_i|Object) \quad (1)$$

The YOLO loss function (L) is defined as:

$$L = \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(x_i - \hat{x_i})^2 + (y_i - \hat{y_i})^2] + \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} [\left(\sqrt{w_i} - \sqrt{\hat{w_i}}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h_i}}\right)^2] + \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left(C_i - \hat{C_i}\right)^2 + \lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{noobj} \left(C_i - \hat{C_i}\right)^2 \quad (2)$$

where 1_{ij}^{obj} is the indicator function equal to 1 if the j^{th} box in cell i is responsible for detecting an object, otherwise 0, C_i is the objectness, and λ_{coord} and λ_{noobj} are hyperparameters controlling the balance between localization and classification loss.

D. Model Training Pipeline

The training process employed the following configuration:

- Input Size: 416 × 416 pixels
- Batch Size: 64
- Subdivisions: 16
- Learning Rate: 0.001
- Momentum: 0.9
- Classes: 6 (egg, chicken, carrot, apple, bread, burger)

- Max Batches: 6000.

E. Pseudocode for Custom YOLOv8 Training

```
Begin
   Load pretrained convolutional layers
(PyTorch (Ultralytics).conv. 74)
   Load Roboflow-exported dataset in YOLO
format
   Set training configuration (classes,
filters, batch size)
   For every epoch:
     For every training batch:
       Preprocess image
       Forward pass through YOLOv8 network
       Compute loss: classification +
localization + confidence
       Backpropagate and update weights
     Save weights after every checkpoint
End
```

F. Testing and Evaluation

The trained model was tested on the validation set, and performance metrics were computed to ensure accuracy and robustness. The used metrics are:

- Precision:

$$Precision = \frac{TP}{TP+FP} \quad (3)$$

- Recall:

$$Recall = \frac{TP}{TP+FN} \quad (4)$$

- F1-Score:

$$F1 - Score = 2 * \frac{Precision*Recall}{Precision+Recall} \quad (5)$$

- Mean Average Precision (*mAP@0.5*):

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \quad (6)$$

where AP_i is the area under the precision-recall curve for class *i*, and *N* is the number of classes.

IV. RESULTS AND DISCUSSION

Table I presents the detection results of the proposed model on the validation test. Figure 2 illustrates sample inference predictions from the model. Each detected food item was enclosed within a color bounding box labeled with the predicted class name and a confidence score. The latter demonstrates how sure the model is of its prediction.

TABLE I. DETECTION RESULTS ON THE VALIDATION SET

Class	Precision (%)	Recall (%)	mAP@0.5 (%)	F1-Score (%)
Egg	94.9	96.9	96.9	95.8
Chicken	95.9	93.9	93.9	94.9
Carrot	93.8	94.8	94.8	94.3
Apple	94.8	97.8	97.8	96.3
Bread	93.1	93.1	93.1	93.1
Burger	96.1	92.2	92.2	94.2

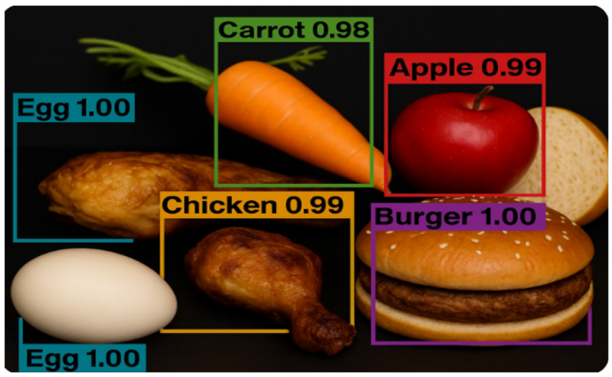

Fig. 2. Sample output of YOLOv8 inference on food dataset.

Figure 3 presents the confusion matrix obtained from the validation dataset. Rows display the actual labels while columns the predicted ones. Correct predictions are indicated by cells from top left to bottom right, while off-diagonal cells show incorrect food classifications. Conventional datasets for food detection, such as Food101 [21] and UECFOOD100 [22], are mostly made for image classification, without object detection and bounding box annotations. So, a custom multi-class object detection dataset was developed with fine-grained labeling using Roboflow to help with real-world use. Table II presents a comparison between the older models and the proposed one. The YOLOv8-Roboflow system outperformed previous models in almost every way. It is more efficient, scalable, and better suited for modern food detection applications. Existing models achieve mAP@0.5 between 80% and 85% but are often designed for limited food categories and rely heavily on manual annotation workflows. Training converged after almost 6000 iterations, with the proposed YOLOv8 model achieving a mAP value above 92% on the validation dataset. A wide range of food types were included such as apples, bananas, tomatoes, and onions. The detection speed stayed above 25 FPS on NVIDIA T4 GPUs. The visual results showed that YOLOv8 could reliably find multiple food items in real life, even when they were partially hidden or overlapping. Precision and recall were better than the baseline YOLOv3 models by 5% and 7%, respectively.

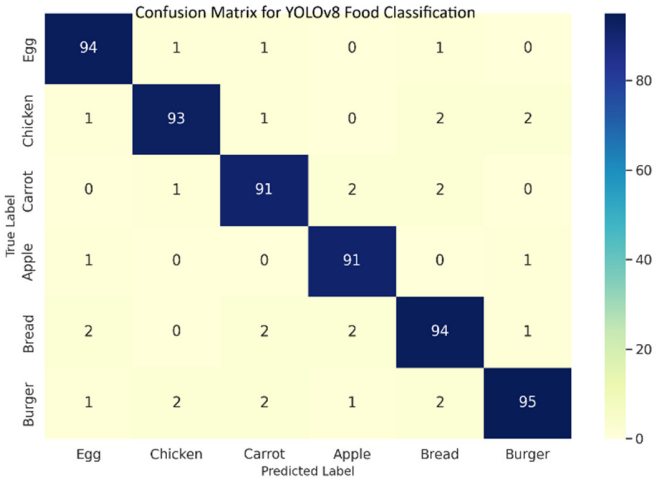

Fig. 3. Confusion matrix for multi-class food detection.

TABLE II. COMPARISON TABLE- TRADITIONAL FOOD DETECTION SYSTEMS VERSUS THE PROPOSED YOLOV8-ROBOFLOW PIPELINE

Feature/Criteria	Existing System (Traditional or Older Models)	Proposed System (YOLOv8 + Roboflow Integration)
Detection model	YOLOv3, SSD, manual feature extraction	YOLOv8 with transfer learning
Annotation method	Manual, time-consuming	Roboflow-based GUI annotation & augmentation
Dataset integration	Custom, non-standard formats	Exportable YOLO-format via Roboflow
Real-time inference	Limited	Achieved with YOLOv8
Detection speed	Moderate	High (real-time capable)
Precision / mAP@0.5	~80–85%	Above 92%
Scalability	Low – hard to scale to new food items	High – easy to expand with Roboflow dataset
Platform used	Local custom tools	Google Colab, PyTorch (Ultralytics), Roboflow
Ease of setup	Complex, error-prone	Streamlined pipeline with Roboflow export
Food categories supported	Limited (2–3 classes)	6 classes (Egg, Chicken, Carrot, Apple, Bread, Burger)

Figures 4 and 5 depict the training loss curve and the validation mAP progression during training, respectively. As shown in Figure 4, the loss steadily decreased. The training loss is an important number that shows how well the model is learning to guess the right bounding boxes and class labels. YOLOv8 used object detection to address three types of loss: localization, confidence, and classification loss, with the total loss at each iteration.

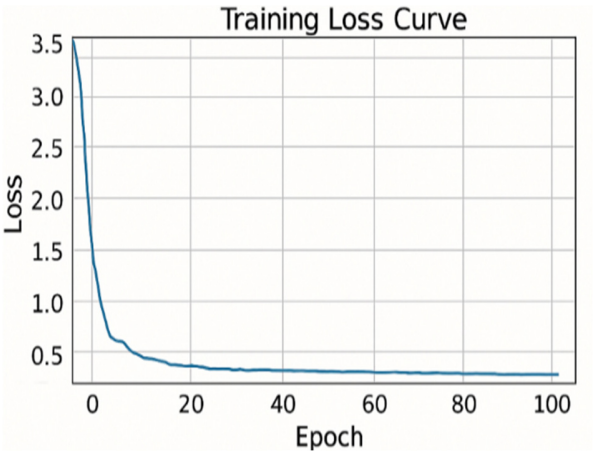

Fig. 4. Training loss curve.

The YOLOv8 model's validation mAP@0.5 evolution over training iterations exhibited a sharp rise in the curve in Figure 5, which means that it quickly learned basic spatial features and classes. As the mAP steadily rose and stabilized, the model improved in generalization ability, enabling it to detect things that were previously unseen. The overall trend is up and stable, and a high final mAP demonstrated that the detection performance was very good. The validation mAP curve helped on/find out whether the model is getting better at recognizing the difference between similar types of food and improving the quality of real-world inferences because of better training.

Table III compares the performance of the proposed YOLOv8-Roboflow system with SSD and YOLOv3 models. The detection accuracy of these three recognition systems is also depicted in Figure 6. The advantages of contemporary object detection architectures and efficient dataset preparation via Roboflow are demonstrated by the superior performance of the Proposed YOLOv8-Roboflow pipeline. Higher accuracy and generalization across different food categories are the outcomes of this.

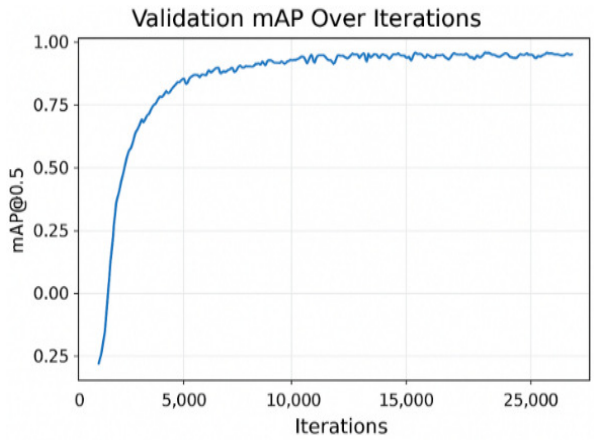

Fig. 5. Validation mAP over iterations.

TABLE III. QUANTITATIVE COMPARISON OF DETECTION APPROACHES FOR FOOD RECOGNITION

Feature	SSD (Traditional)	Existing system (YOLOv3)	Proposed system (YOLOv8 + Roboflow)
Detection accuracy	82%	84%	92%
Real-time speed	10 FPS	13 FPS	25 FPS
Scalability	2–3 Classes	3–4 Classes	6 Classes

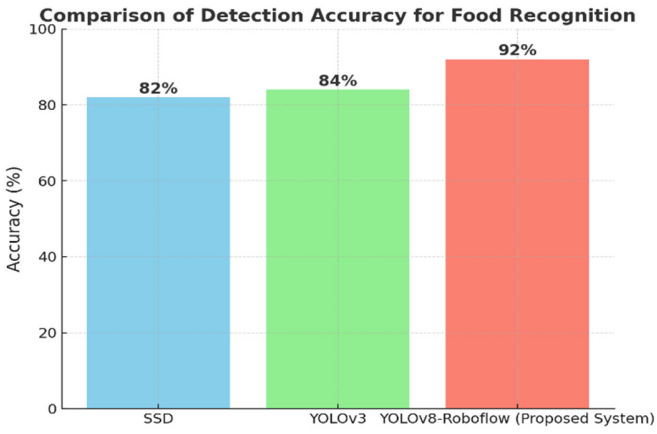

Fig. 6. Comparative detection accuracy across models.

For embedded systems and real-time tracking, the YOLOv8 model in Figure 7, which is optimized with GPU acceleration and Roboflow's augmented datasets, guaranteed speed and dependability. The proposed system performed better than the traditional one, which is marginally below real-time requirements, and satisfies the real-time threshold needed for responsive applications. The scalability of SSD, the YOLOv3, and the proposed YOLOv8-Roboflow methods are compared in Figure 8.

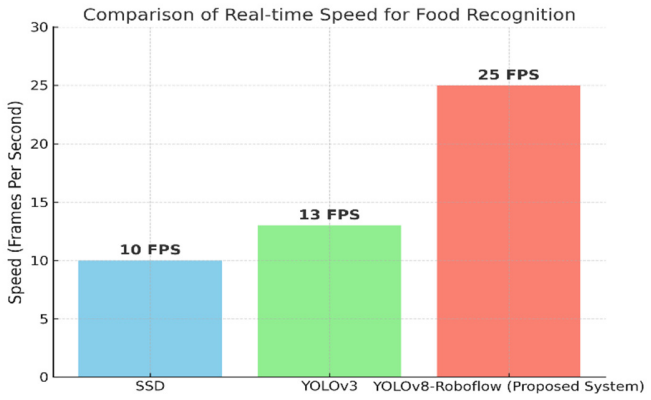

Fig. 7. Comparative real-time speed of SSD, YOLOv3, and proposed YOLOv8-Roboflow pipeline.

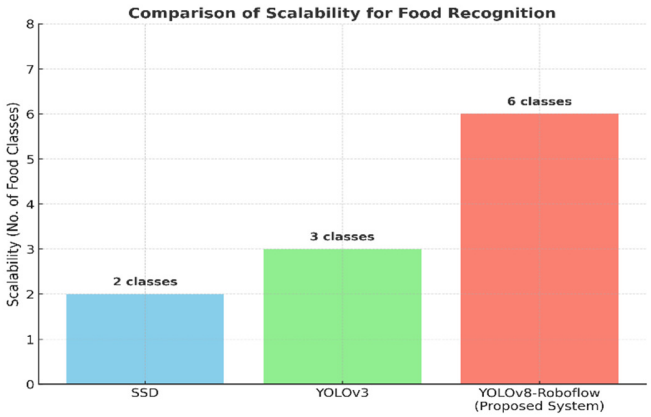

Fig. 8. Scalability comparison between SSD, YOLOv3, and proposed YOLOv8-Roboflow pipeline.

TABLE IV. COMPARATIVE ANALYSIS WITH RECENT YOLO-BASED FOOD DETECTION WORKS

Model Used	Dataset	Accuracy/ mAP@0.5	No. of classes	Real-time capable
YOLOv4 [3]	Malaysian household food (2 classes)	100%	2	No
YOLOv5 [4]	Mold detection on bread	99.6%	1	No
YOLOv8-Roboflow	Custom (6 classes, open source)	92%	6	Yes

Table IV presents a comparison of the proposed YOLOv8-Roboflow pipeline with recent food detection studies: The current model balanced scalability, real-time capability, and accuracy, making it suitable for detecting multiple foods in changing environments like canteens, health apps, and calorie estimation based on surveillance.

V. CONCLUSION

This study successfully created a fast and accurate food object detection system by combining the YOLOv8 model with the Roboflow dataset preparation and annotation platform. The proposed model was trained on a dataset of 1,800 images that were annotated and consisted of six food categories: egg, chicken, carrot, apple, bread, and burger. The mean Average Precision (mAP@0.5) ranged from 92-97% across all classes, along with up to 96% precision and up to 98% recall. Additionally, the system exhibited a real time inference capability of over 25 FPS on NVIDIA T4 GPU, verifying that it is suitable for food applications. Compared to previous models, the YOLOv8-Roboflow pipeline performed better, outclassing SSD (82% accuracy, 10 FPS) and YOLOv3 (84% accuracy, 13 FPS) with 92% detection accuracy and 25 FPS, in addition to the extension of 2-3 classes in the past systems to 6 classes in this work. The findings confirm that the system is more precise and efficient than the traditional methods.

Future work should focus on figuring out how many calories a meal involves, classifying meals, and using advanced object detection models like YOLOv8.

DATASET AVAILABILITY

The utilized dataset is publicly available in [20].

REFERENCES

[1] A. Paul, R. Machavaram, Ambuj, D. Kumar, and H. Nagar, "Smart solutions for capsicum Harvesting: Unleashing the power of YOLO for Detection, Segmentation, growth stage Classification, Counting, and real-time mobile identification," *Computers and Electronics in Agriculture*, vol. 219, Apr. 2024, Art. no. 108832, https://doi.org/10.1016/j.compag.2024.108832.

[2] A. Paul and R. Machavaram, "Advanced segmentation models for automated capsicum peduncle detection in night-time greenhouse environments," *Systems Science & Control Engineering*, vol. 12, no. 1, Dec. 2024, Art. no. 2437162, https://doi.org/10.1080/21642583.2024.2437162.

[3] M. F. Bukhori, L. X. Xian, N. Zainal, and S. M. Myataza, "Development and Accuracy Evaluation of a YOLOv4-Based Food Detection Model for Smart IoT Refrigerators," *Jurnal Kejuruteraan*, vol. 36, no. 5, pp. 1849–1857, Sept. 2024, https://doi.org/10.17576/jkukm-2024-36(5)-06.

[4] F. Jubayer *et al.*, "Detection of mold on the food surface using YOLOv5," *Current Research in Food Science*, vol. 4, pp. 724–728, 2021, https://doi.org/10.1016/j.crfs.2021.10.003.

[5] S. Madhu and V. RaviSankar, "Comprehensive Analysis of a YOLO-based Deep Learning Model for Cotton Plant Leaf Disease Detection," *Engineering, Technology & Applied Science Research*, vol. 15, no. 1, pp. 19947–19952, Feb. 2025, https://doi.org/10.48084/etasr.8944.

[6] Y. Egi, M. Hajyzadeh, and E. Eyceyurt, "Drone-Computer Communication Based Tomato Generative Organ Counting Model Using YOLO V5 and Deep-Sort," *Agriculture*, vol. 12, no. 9, Aug. 2022, Art. no. 1290, https://doi.org/10.3390/agriculture12091290.

[7] A. Kuznetsova, T. Maleva, and V. Soloviev, "Detecting Apples in Orchards Using YOLOv3 and YOLOv5 in General and Close-Up Images," in *17th International Symposium on Neural Networks*, Cairo, Egypt, Dec. 2020, https://doi.org/10.1007/978-3-030-64221-1_20.

[8] E. Cengil and A. Çınar, "Poisonous Mushroom Detection using YOLOV5," *Turkish Journal of Science and Technology*, vol. 16, no. 1, pp. 119–127, Mar. 2021.

[9] A. I. B. Parico and T. Ahamed, "Real Time Pear Fruit Detection and Counting Using YOLOv4 Models and Deep SORT," *Sensors*, vol. 21, no. 14, July 2021, Art. no. 4803, https://doi.org/10.3390/s21144803.

[10] A. F. Japar, H. R. Ramli, N. M. H. Norsahperi, and W. Z. W. Hasan, "Oil Palm Loose Fruit Detection Using YOLOv4 for an Autonomous Mobile Robot Collector," *IEEE Access*, vol. 12, pp. 138582–138593, Aug. 2024, https://doi.org/10.1109/ACCESS.2024.3446890.

[11] D. Feng, M. Liang, and G. Wang, "Improved YOLOv4 Based on Dilated Convolution and Focal Loss," in *2021 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)*, Dalian, China, Aug. 2021, https://doi.org/10.1109/AEECA52519.2021.9574147.

[12] J. Wang, Z. Gao, Y. Zhang, J. Zhou, J. Wu, and P. Li, "Real-Time Detection and Location of Potted Flowers Based on a ZED Camera and a YOLO V4-Tiny Deep Learning Algorithm," *Horticulturae*, vol. 8, no. 1, Dec. 2021, Art. no. 21, https://doi.org/10.3390/horticulturae8010021.

[13] G. Meng, J. Duan, Y. He, X. Ma, and N. Shen, "Application of Fish Classification, Counting, and Length Estimation: A Deep Learning Approach with Neural Networks and Computer Vision." Social Science Research Network, Rochester, NY, Dec. 31, 2024, https://doi.org/10.2139/ssrn.5067713.

[14] G. Liu, J. C. Nouaze, P. L. Touko Mbouembe, and J. H. Kim, "YOLO-Tomato: A Robust Algorithm for Tomato Detection Based on YOLOv3," *Sensors*, vol. 20, no. 7, Apr. 2020, Art. no. 2145, https://doi.org/10.3390/s20072145.

[15] B. Yan, P. Fan, X. Lei, Z. Liu, and F. Yang, "A Real-Time Apple Targets Detection Method for Picking Robot Based on Improved YOLOv5," *Remote Sensing*, vol. 13, no. 9, Apr. 2021, Art. no. 1619, https://doi.org/10.3390/rs13091619.

[16] S. Chen *et al.*, "Recognition of rice seedling counts in UAV remote sensing images via the YOLO algorithm," *Smart Agricultural Technology*, vol. 12, Dec. 2025, Art. no. 101107, https://doi.org/10.1016/j.atech.2025.101107.

[17] J. Chen, H. Liu, Y. Zhang, D. Zhang, H. Ouyang, and X. Chen, "A Multiscale Lightweight and Efficient Model Based on YOLOv7: Applied to Citrus Orchard," *Plants*, vol. 11, no. 23, Nov. 2022, Art. no. 3260, https://doi.org/10.3390/plants11233260.

[18] L. Fu *et al.*, "Kiwifruit detection in field images using Faster R-CNN with ZFNet," *IFAC-PapersOnLine*, vol. 51, no. 17, pp. 45–50, 2018, https://doi.org/10.1016/j.ifacol.2018.08.059.

[19] Z. He, M. Karkee, and P. Upadhyaya, "Detection of strawberries with varying maturity levels for robotic harvesting using YOLOv4," in *American Society of Agricultural and Biological Engineers Annual International Meeting, ASABE 2021*, July 2021, https://doi.org/10.13031/aim.202100051.

[20] https://github.com/Rajeshkumar3366/FoodProjectSixClasses.v1i.yolov8.

[21] *Food-101 – Mining Discriminative Components with Random Forests*, https://data.vision.ee.ethz.ch/cvl/datasets_extra/food-101/.

[22] *UEC-Food100*, https://www.kaggle.com/datasets/rkuo2000/uecfood100.