Efficient deep learning-based approach for detecting citrus fruits

Ghazif Adeem[1], Omar Abdulkader[2], Muhammad Jawad Ikram[2], Muhammad Aqib[1,3*], Yaser Hafeez[1], Muhammad Naveed Tahir[4], Shoaib Rashid Saleem[5], Yingkuan Wang[6], Yubin Lan[7]

(1. *University Institute of Information Technology, PMAS-Arid Agriculture University Rawalpindi, Rawalpindi 46300, Pakistan*;
2. *Faculty of Computer Studies, Arab Open University, Riyadh, Saudi Arabia*;
3. *National Center of Industrial Biotechnology, PMAS-Arid Agriculture University Rawalpindi, Rawalpindi 46300, Pakistan*;
4. *Department of Agronomy, PMAS-Arid Agriculture University Rawalpindi, Rawalpindi 46000, Pakistan*;
5. *Department of Farm Machinery and Precision Engineering, Faculty of Agricultural Engineering and Technology, PMAS-Arid Agriculture University Rawalpindi, Rawalpindi 46000, Pakistan*;
6. *Academy of Agricultural Planning and Engineering, Ministry of Agriculture and Rural Affairs of PRC, Beijing 100125, China*;
7. *National Center for International Collaboration Research on Precision Agriculture Aviation Pesticides Spraying Technology, South China Agricultural University, Guangzhou 510642, China*)

Abstract: Yield monitoring is crucial for the agricultural sector, as it can be used to inform decisions on harvesting, storage, and transportation. Traditionally, several statistical methods and visual inspection techniques are employed to get an early estimate of the final yield of citrus, with the downside of being inaccurate, costly, and time-consuming. In recent years, there have been a lot of advancements in the fields of Artificial Intelligence (AI) and computer vision, providing opportunities to automate plenty of things in different domains, including agriculture. This research proposes a deep learning-based framework that leverages multiple Convolutional Neural Networks (CNN) to efficiently and effectively operate in real-world environments, using field data to provide accurate, improved yield estimates. A high-quality dataset, consisting of citrus tree images, is obtained from orchards at the university research farm Koont and the National Agriculture Research Center (NARC). Afterwards, the CNN-based models are trained thoroughly with various configurations and data augmentation techniques. All the models are rigorously tested and evaluated on the basis of a number of performance metrics. Experiments have shown that YOLOv8m performs with the highest mean average precision, reaching up to 90% with an inference time of a few milliseconds, making it worthy to be deployed for fruit detection, counting, and yield estimation tasks.

Keywords: deep learning, convolutional neural networks, object detection, sensor, yield monitoring, fruit counting
DOI: 10.25165/j.ijabe.20261903.9702

Citation: Adeem G, Abdulkader O, Ikram M J, Aqib M, Hafeez Y, Tahir M N, et al. Efficient deep learning-based approach for detecting citrus fruits. Int J Agric & Biol Eng, 2026; 19(3): 267–280.

1 Introduction

Crop yield monitoring is to properly keep on tracking the state of crops in various weather conditions, monitoring diseases for the purpose of gaining an insight into the final yield of the crops. Early on, yield monitoring and yield estimation give the farmers the benefit of improved and effective decision-making. Yield estimation can give farmers the ability to schedule their activities, such as logistics, storage of crops, and sales strategies[1,2]. The information acquired through the crop yield monitoring and estimations is critical for ensuring the security of food and agricultural production, and for properly maintaining agricultural development[3-5].

Researchers also report that crop yield monitoring and estimation at the regional scale during drought conditions help farmers in making well-informed and effective decisions, as well as guiding agronomical management[6,7]. Yield monitoring and estimation are the core tasks from the perspective of management of crops and in terms of marketing. Based on the results of yield estimation, better decisions with regard to the period of harvesting, crop disease prevention strategies, and follow-up for practices of cultivation can be carried out by farmers. Throughout the world, a huge number of citrus fruits are being cultivated, harvested, and produced. As stated by FAOSTAT (United Nations Food and Agriculture Organization), citrus production is nearly 157.98×10^6 t, out of which the number of oranges is more than half of the production[8]. Economies produce a large amount of revenue from these crops. The huge production amount is to be properly monitored in various weather conditions and against various pests and diseases to make sure the yield is profitable. Figure 1 shows citrus worldwide production by the end of the year 2021.

Citrus fruits have been the area of research for many researchers for tasks including disease detection, fruit severity detection, yield monitoring and estimation, and many more[9,10]. Many researchers have carried out a number of studies[11-13] on yield estimation for a variety of crops, including cotton, citrus, wheat, and soybean, respectively.

Received date: 2025-01-23 **Accepted date:** 2025-09-25
Biographies: Ghazif Adeem, Research Scholar, research interest: deep learning, object detection, digital agriculture, Email: ghazifadeem@gmail.com; **Omar Abdulkader**, PhD, research interest: cybersecurity, IoT, M2M, artificial intelligent, blockchain, Email: o.abdulkader@arabou.edu.sa; **Muhammad Jawad Ikram**, PhD, research interest: energy-aware algorithms, HPC, GPU computing, AI, federated learning, Email: m.ikram@arabou.edu.sa; **Yaser Hafeez**, Professor, research interest: AI, Email: yasir@uaar.edu.pk; **Muhammad Naveed Tahir**, Associate Professor, research interest: precision agriculture, Email: naveed@uaar.edu.pk; **Shoaib Rashid Saleem**, Assistant Professor, research interest: precision agriculture, Email: shoaibrashid@uaar.edu.pk; **Yingkuan Wang**, Professor, research interest: agricultural informatization, Email: wangyk@agri.gov.cn; **Yubin Lan**, Professor, research interest: precision agricultural aviation, low-altitude economy, Email: ylan@scau.edu.cn.
***Corresponding author: Muhammad Aqib**, Assistant Professor, research interest: AI, deep learning. UIIT, PMAS-Arid Agriculture University Rawalpindi, Pakistan, Email: aqib.qazi@uaar.edu.pk.

The overview of the approaches with regard to yield monitoring and estimation is shown in Figure 2.

This research has the following objectives:

1) To identify and analyze the appropriate yield monitoring techniques and deep learning-based approaches;

2) To collect real data from fields for making estimations in real environments;

3) To propose a deep learning-based yield estimation model that can be used for yield prediction in real environments with high accuracy;

4) To develop a mobile application for the Android platform utilizing the best model for the detection and yield estimation of citrus fruits.

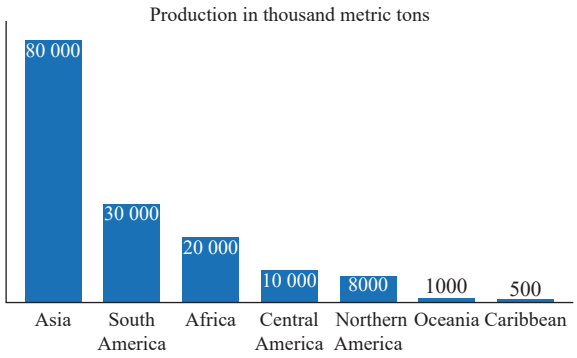

Figure 1 Worldwide citrus production by the end of 2021

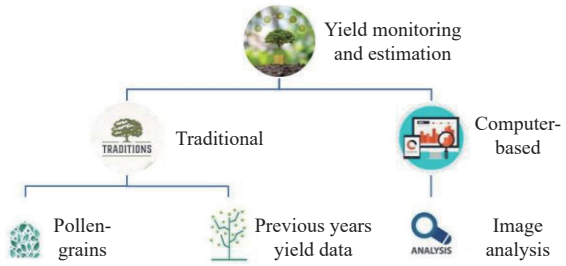

Figure 2 Yield monitoring and estimation techniques

2 Background

This section gives a brief introduction to tools and deep learning models that are used in this research.

2.1 Tools and technologies

For training deep learning model(s) in a supervised learning fashion, where both the inputs and outputs (labels) are already available, the model is assigned to perform a mapping of input to output targets. In case of images, labeling and annotation must be done on each input image using bounding boxes (one of the ways of annotation) for each target class. Afterwards, these annotated images are passed to the network, and it is fitted on this training data to make generalizations on real-world data because of this training.

2.1.1 Label Image (LabelImg)

For annotation and labeling, LabelImg software has been used. It is a free, open-source solution for this purpose. It comes as a pip (Python dependency manager) package and can be easily installed. For data annotation and labeling, version 1.8.6 has been used on the Linux operating system. Linux provides it via the pip package manager. Pip package manager pulls it from the official Python repository of packages named PyPi. After downloading and installing it via pip, a virtual environment for Python is set up, and then LabelImg is utilized. Its interface can be seen in Figure 3.

It supports several labeling formats for various object detection models. Formats include YOLO, PascalVOC, and createML. Each format results in either an XML or a txt file containing the coordinates of all bounding boxes (classes). The process of labeling and annotation is fairly easy. The alternative to this open-source library can be YOLO-mark, which is designed specifically with YOLO object detection models by providing labeling in text file format. Steps include:

1) Open the directory containing training images.

2) Select labeling format, i.e., YOLO, PascalVOC, etc.

3) Select the save directory.

4) Perform annotation and assign labels.

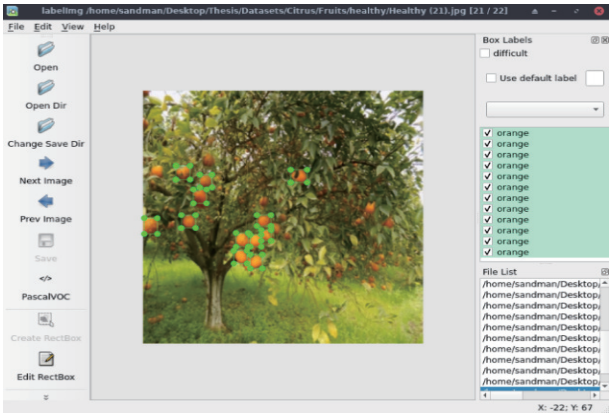

Figure 3 An overview of LabelImg interface

2.1.2 You Only Look Once (YOLO)

YOLO is the current state-of-the-art object detection model, even with capabilities of performing detection in real time with high mean average precision (mAP). Right now, YOLOv7 and YOLOv8 are in head-to-head competition with other object detection models, including Cascaded R-CNN, DETR, Vision Transformers, and RetinaNet, to name a few. The general architecture diagram for YOLO is given in Figure 4. YOLOv7 comes in various architectures, some of which are shown in Figure 5.

The YOLO model takes an image as input and then passes it through a simple Deep Convolutional Neural Network (DCNN) for detecting objects of interest in the image. The first few convolutional layers of YOLO are pre-trained using ImageNet by utilizing an average pooling and a Fully Connected (FC) layer. Then, YOLO's final FC layer predicts coordinates of bounding boxes as well as class probabilities[14].

YOLO breaks an input image into a grid where the number is. An object is detected if the center of the target class falls into a grid cell, making that grid responsible for detection. Then, each grid predicts a confidence score for each of those bounding boxes as well as the count of actual bounding boxes. Confidence score is defined as the model's certainty on how accurate the predicted bounding box is and the likelihood of it having an object in that box. YOLO tries to predict multiple bounding boxes for each grid cell. It assigns one predictor to be responsible for detecting objects based on the highest IoU threshold in comparison to the actual target or ground truth. Then, with each training cycle, results get better and better, allowing each predictor to give improved results for various image sizes or objects (classes), while improving recall. YOLO utilizes a key technique, i.e., Non-Maximum Suppression (NMS), in the post-processing phase to improve the efficiency and accuracy of detection of target objects in given input images. In an object detection setting, an object in a single image can have multiple predicted bounding boxes over it, making the boxes

overlap, although the object in each box is the same. NMS identifies and removes such redundant bounding boxes and outputs a single bounding box over the target object.

YOLOv7 introduces a new loss function, namely "focal loss", over the traditional cross-entropy function, which is less effective at detecting small objects[15]. Focal loss function overcomes this limitation by down-weighting losses of well-classified targets and focuses on hard and complex targets to be detected. Both YOLOv7 and YOLOv8 improve detection accuracy and result in inference times of milliseconds, making them real-time capable.

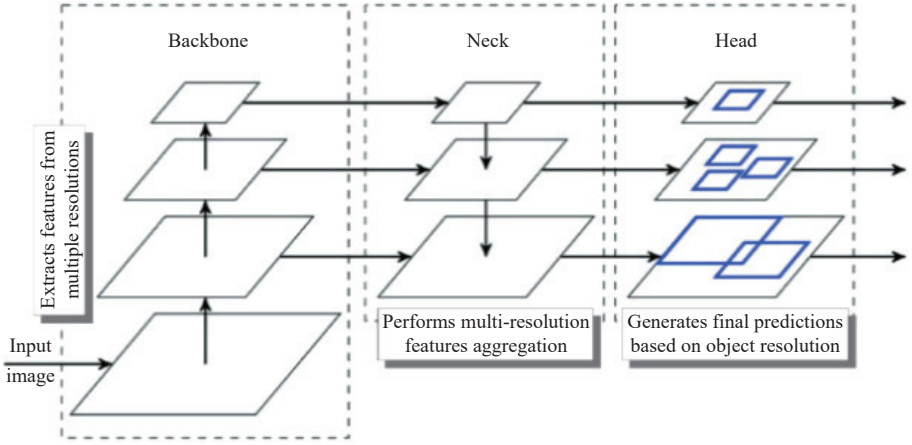

Figure 4 General YOLO architecture for object detection

	YOLOv7	YOLOv7x	YOLOv7-tiny
Stage 0	3×3/1 Conv, 32	3×3/1 Conv, 40	
Stage 1	3×3/2 Conv, 64 3×3/1 Conv, 64	3×3/2 Conv, 80 3×3/1 Conv, 80	3×3/2 Conv, 32
Stage 2	3×3/2 Conv, 128 2-4 ELAN, 256	3×3/2 Conv, 160 2-6 ELAN, 320	3×3/2 Conv, 64 1-2 ELAN, 64
Stage 3	/2 Down, 256 2-4 ELAN, 512	/2 Down, 320 2-6 ELAN, 640	/2 MaxPool 1-2 ELAN, 128
Stage 4	/2 Down, 512 2-4 ELAN, 1024	/2 Down, 640 2-6 ELAN, 1280	/2 MaxPool 1-2 ELAN, 256
Stage 5	/2 Down, 1024 2-4 ELAN, 1024	/2 Down, 1280 2-6 ELAN, 1280	/2 MaxPool 1-2 ELAN, 512
Stage 5	CSPSPP, 512	CSPSPP, 640	CSPSPP, 256
Stage 4	*2 Up, 512 1-4 ELAN, 256	*2 Up, 640 2-6 ELAN, 320	*2 Up, 256 1-2 ELAN, 128
Stage 3	*2 Up, 256 1-4 ELAN, 128	*2 Up, 320 2-6 ELAN, 160	*2 Up, 128 1-2 ELAN, 64
Stage 4	/2 Down, 512 1-4 ELAN, 256	/2 Down, 640 2-6 ELAN, 320	/2 Down, 256 1-2 ELAN, 128
Stage 5	/2 Down, 1024 1-4 ELAN, 512	/2 Down, 1280 2-6 ELAN, 640	/2 Down, 512 1-2 ELAN, 256

Figure 5 Various YOLOv7 architectures

2.1.3 Faster R-CNN

Another commonly used deep learning-based object detection model is Faster R-CNN. It improves over traditional R-CNN and Fast R-CNN by means of overcoming their limitations. It introduces the concept of Region Proposal Network (RPN), which is basically a convolutional neural network behind the scenes, generating proposals for various aspect ratios and scales. It does this with the help of an attention mechanism that allows the backbone of Fast R-CNN to tell where to look for objects in that given input image. Previously, the concept of the pyramid of images (i.e., images having different scales and ratios) and the pyramid of filters (i.e., different-sized filters over the same image) was used in object detection scenarios.

Faster R-CNN proposes concepts of anchor boxes. An anchor box is basically a reference box for a particular set of aspect ratios and image scale. For a single region, multiple such anchor boxes of various aspects and scales exist, also called the pyramid of anchor boxes. Each region maps to a particular set of anchor boxes having scale and aspect ratio, thus detecting objects in a variety of scales and aspect ratios. All these computations are shared between the Fast R-CNN backbone and the Region Proposal Network (RPN), hence cutting the computational time. Figure 6 depicts the general structure for Faster R-CNN. In simple terms, RPN produces region proposals for an image. Out of all these proposals, a fixed-size feature vector is extracted from them using a pooling layer of Region of Interest (ROI). Carrying these feature vectors to Fast R-CNN, classification is done. Fast R-CNN then results in the detected objects' class score and their bounding boxes over the target classes.

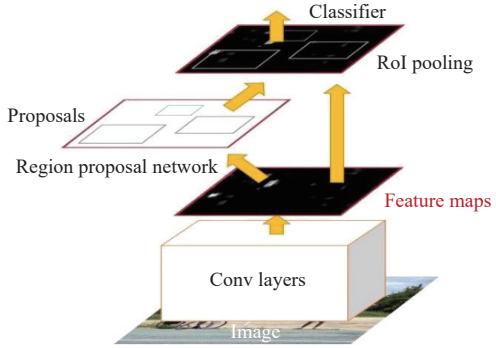

Figure 6 General overview of Faster R-CNN

2.1.4 Detectron2 Library

To quickly set up and get started with deep learning object detection and segmentation projects, a team at Facebook came up with a library known as Detectron. It provides a handsome amount of object detection and segmentation algorithms. Detectron2 is an improved and upgraded version with capabilities to apply Vision Transformers, Cascaded R-CNN, Panoptic segmentation, and much more. All the models provided in Detectron2 are exportable to various other production-ready formats, such as Caffe2 or Torch Script. The Facebook team worked quite extensively to provide models to be trained much faster[16]. Some of the models provided by Detectron2 include: 1) Faster R-CNN; 2) RetinaNet; 3) RPN & Fast R-CNN; 4) Mask R-CNN; 5) Cascade R-CNN.

Faster R-CNN in this research has been applied using the Detectron2 library as the backend.

3 Related works

In this section, previously conducted studies in yield monitoring and estimation are discussed to show the need for an improved yield monitoring approach that can work with real data in real environments[17,18]. These studies worked on the fruit detection and load (yield) estimation for orange fruits. They applied the detection methods based on deep learning techniques like You Only Look Once (YOLO) and its various versions for the detection and counting of oranges in an orchard. Different image datasets for oranges were created under different illumination conditions out of 100 sample orange trees. Out of all the tested YOLO versions, YOLO-v4 resulted in the highest orange detection rate of 92%, which led the researchers to apply it for yield estimations, which ended up showing tree yield. The real data samples were very few and were not properly labeled and annotated, making them not very effective to be deployed in real environments to work with real data. The YOLO version 4 is not suitable for detecting small-sized fruits as it suffers from accuracy issues in such cases. There are other recent approaches that could have been adapted to achieve even more improved results.

Another similar study with regards to yield estimation was done for melons using images acquired from unmanned aerial vehicles[19,20]. They worked on identification and yield estimation of melons from top-view colored images captured from a digital camera from an unmanned aerial vehicle. The detection procedure was based on RetinaNet DCNN, alongside transfer learning for training for the detection of small objects in High Definition (HD) images. The difference between the actual and estimated yield was 3% in melon yield estimation. Deep learning approaches have also been employed in a number of disease detection and classification scenarios[21–24].

The research in Reference[14] has come up with three machine learning based models for the task of citrus yield prediction based on aerial vision. The authors captured two images of each tree, from a total of 48 trees, using a DJI Phantom 4 Pro drone. The first model is built to detect fruits (fruits per tree) using YOLOv3, and then this output is given as input to model-2 and model-3 for the purpose of yield prediction. The second model utilizes a bunch of machine learning models, namely Partial Least Squares Regression (PLSR), Random Forest Regression (RFR), Linear Regression (LR), and Gradient Boosting Regression (GBR) on the one-sided fruit count provided by YOLOv3. Similarly, the third model incorporates the fruit count of both sides (front and back) provided by YOLOv3 and provides yield estimation on this basis. All the models were evaluated using Mean Absolute Percentage Error (MAPE). Model-3 utilizing LR resulted in lower yield estimation variability.

Lin et al.[21] carried out research on the flowering rate of litchi fruits. Initially, high-quality images of litchi at different flowering stages are captured by means of a DJI Mavic 2 Pro UAV. Nearly 280 images of litchi during the flowering period are captured and stored in JPG format. To feed the data to the neural network, i.e., YOLOv4, preprocessing steps such as enhancement and cropping are applied. Noisy and useless images are deleted, and the remaining useful images are enhanced using augmentation techniques such as vertical and horizontal flipping and rotation to provide the model with complex patterns to overcome overfitting issues. The dataset is augmented to come up with 1200 images in total, which were then split into three sets, i.e., training (720 images), validation (240 images), and testing (240 images). Afterwards, the dataset is labeled using the LabelImg tool and passed into YOLOv4 for detection. The highest achieved mAP was 85% with 0.073 s of detection time. Gavahi et al.[25] proposed a hybrid technique, namely Deep Yield, by the combination of Convolutional LSTM and 3DCNN structures for robust crop yield forecasting. The yield data of county-based soybean statistics from 2003-2009 and multiple variables like day and nighttime land surface temperatures were used for model training with Adam optimizer with a high-end Graphical Processing Unit (GPU) for heavy load lifting. The input images were 4D tensors having dimensions such as Height, Width, Band, and Time. The model's Root Mean Square Error (RMSE) is compared with other approaches, showing that its RMSE is lower than that of the compared models. For instance, the RMSE of the Deep Yield is 4.85 against 3DCNN, which is 5.97. The real dataset was not involved, due to which it is not adaptable to work in real environments, and the dataset that was used was not up to date.

Mendez et al.[26] approached the orange number counting and load (size) estimation using 3D Laser scanning using 3D Light Detection and Ranging (LIDAR) models. The orange trees, both pruned and unpruned (a total of 24 trees), were modeled with LIDAR, with their color captures that are used for segmentation and fruit detection using a clustering algorithm like K-means. The yield estimation difference between the estimated and actual yield is only 2.5% by means of applying regression. The 3D LIDAR scanning approach is time-consuming but gives pretty good estimation results. The K-means algorithm resulted in very low accuracy due to an insufficient number of images in the dataset. Redmon et al.[14] proposed a new algorithm based on image processing for detecting and counting citrus fruits. A total of 133 high-resolution images of citrus trees were captured by the iPhone XR digital camera. The proposed algorithm is made simple with steps like image acquisition, converting images into different color spaces, applying the Hough Transformation algorithm, and then fruit counting and yield estimation mappings. The results showed the detection accuracy for the algorithm to be 91.7%, and for yield estimation, it reached up to 94.7%. The results were compared to a few other existing approaches. The number of real dataset samples is not sufficient, which can make this approach suffer while working with a large number of real datasets in real environments.

The use of Regional Convolutional Neural Network (R-CNN) for automated detection of apple fruits, in addition to making yield maps using Ortho mosaic maps taken from UAV images, was also done[5,27]. The results obtained were compared to the apples counted by the agro-technician, and a regression coefficient of 0.86 was achieved, with a mean absolute error of 10.35 and root mean square error of 13.56%. The experimentation was conducted on Google Colab. The results could have been much improved by using Fast Recurrent Convolutional Neural Network or Faster Recurrent Neural Network versions of Recurrent CNN. The experimentation was conducted only in a laboratory-based environment without any deployment in real environments to evaluate the performance of the proposed approach.

Xia et al.[28] worked on yield estimation for both citrus and apple orchards. Video sequences of both apple and citrus orchards were captured using an iPhone 8 with 1080×1920 resolution. Following that, a total of 240 frames were extracted for apple and citrus orchards, each having 120 frames. Sequences were divided into two equal parts for both fruits for training and evaluation purposes. Subsequently, each fruit in the sequences was labeled and annotated using a self-developed labeling tool. The CenterNet

model is trained to detect apples and citrus fruits. CenterNet gave fruit count, which is checked to remove duplicate fruits by comparing their similarity by means of another model utilizing the Kuhn-Munkres optimization algorithm. The proposed model achieves an mAP of 93% but performs slower detection.

4 Methodology

For detecting citrus and estimating yield through images, You Only Look Once (YOLO) has been used for experimentation. Improved and better citrus detection results are most promising for yield estimation. YOLO models are current state-of-the-art object detection approaches with higher mean average precision and the capability to perform under real-time circumstances. Figure 7 illustrates the methodology followed for this research. The proposed methodology consists mainly of five steps, i.e., data acquisition; preprocessing, which involves data cleaning, data augmentation, and data labeling; training of deep learning models; validation; and making predictions.

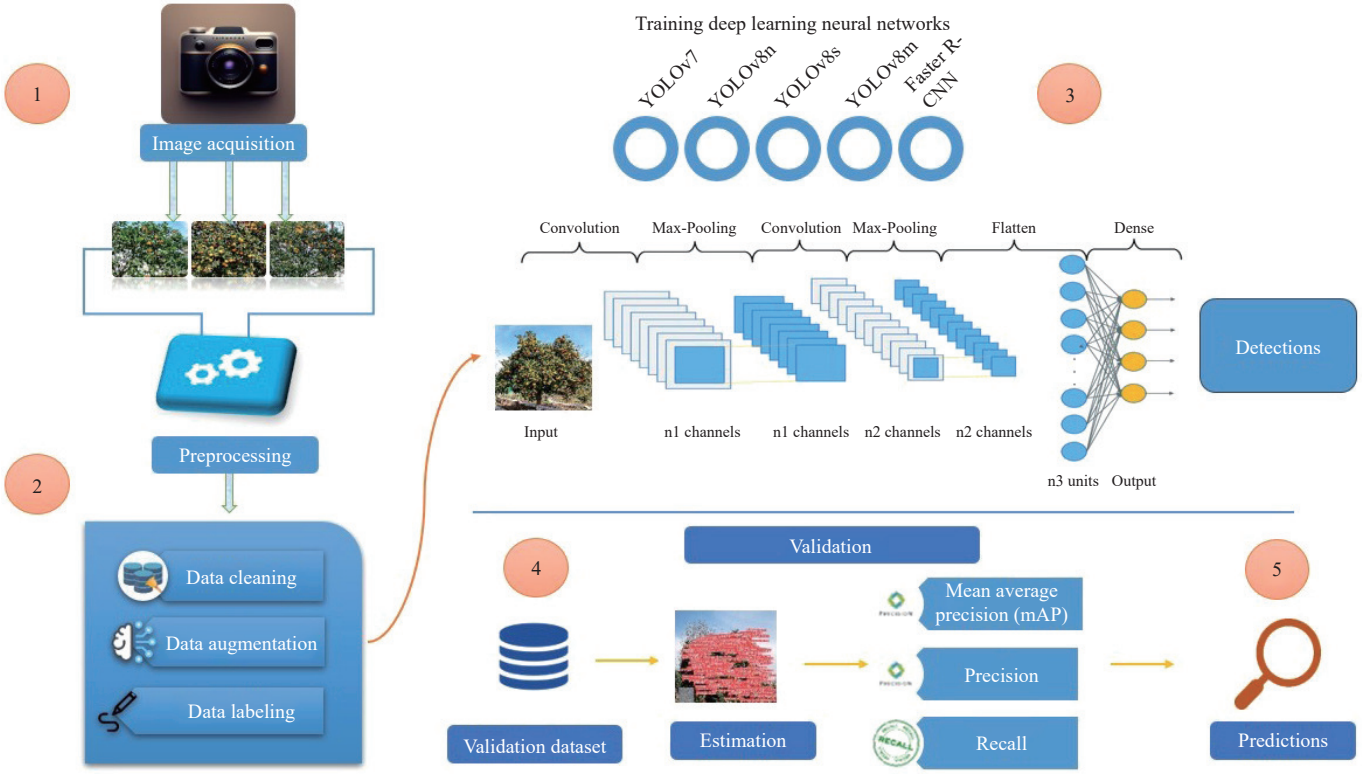

Figure 7 An overview of the proposed methodology for citrus detection

4.1 Data acquisition

For object detection, proper yield monitoring, and yield estimation, high-quality images are to be captured. For conducting our experimentation, high-quality citrus tree images from several different locations, including the university research farm Koont and the National Agriculture Research Center (NARC), are captured using a Samsung A42 5 GB with a 48MP camera. Each image is of resolution 2992×2992 pixels. Near real-time 50 citrus trees of a variety of citrus varieties are considered for research purposes. Images are captured under different illumination conditions, as they have an impact on detection accuracy. Images captured in sunny and proper sunlight tend to be detected with higher accuracy, while the other way around, accuracy dwindles. Images are stored in JPG format. Some of the sample images are shown in Figure 8.

Figure 8 Sample images of collected citrus trees

4.2 Data preprocessing

In this section, details of data preprocessing steps are given. These steps include data cleaning, data augmentation, and data labeling.

4.2.1 Data cleaning

To perform training of deep learning-based models, data must be preprocessed, and any corrupt, blurred, or noisy images must be removed. Otherwise, such data can cause trained models to lose robustness and can result in poor detection accuracy. So, all the images in the dataset that are noisy, blurred, or corrupted are deleted to make sure that the training phase does not suffer from poor performance.

4.2.2 Data augmentation

A lot of data is needed for training and testing any object detection model (or any deep learning model). The higher the amount of data, the better the results, i.e., detection accuracy. One hundred high-quality images are augmented to increase the dataset to 1000 images. There are many different strategies applied for performing data augmentations such as scaling, rotation, flipping, pixel multiplication, Gaussian blur, and additive Gaussian noise. This is done using open-source utilities, i.e., imgaug, albumentations, and augmentor, due to the availability of a huge variety of augmentation techniques. Figure 9 portrays some of the augmented samples. Some of the augmentation techniques with a brief introduction are as follows.

Rotation: In this technique, images are rotated to both left and right sides with a probability of 0.5%. Probability in this case means

a configuration setting for the libraries that are used to generate augmented images.

Figure 9 Augmentation with patches, rotation, and brightness adjustment

It simply means how much chance there is for applying a particular transform to each image. A few minor edge cases can be detected using the rotation augmentation technique.

Flipping: Images are augmented using both horizontal and vertical flipping techniques with a probability of 0.5%. Third configuration with a random flip is also applied, which results in the image being automatically generated in both randomly rotated and randomly flipped positions.

Zooming: Another augmentation transform applied is zooming, with an occurrence probability of 0.5%, with a minimum of 1.1 and a maximum of 1.5 of zooming. This helps to cope with cases in which input images are captured from far away. Once a model is trained on augmented data with zooming transforms applied, it can perform slightly better in such scenarios.

Brightness and contrast adjustment: In the case of object detection scenarios, illumination conditions can have a huge impact on detection accuracy. Lower or dim lighting conditions can cause detection accuracy to drop. Similarly, too much bright light can cause models to either miss detection completely or misidentify the objects. Therefore, to have improved accuracy in such cases and environments, brightness and contrast adjustment are done. Transforms, i.e., random brightness and random contrast, are applied with settings of minimum factor of brightness from 0.3 to maximum factor of 0.6 and minimum factor of 0.2 and maximum factor of 0.8 for contrast adjustment, respectively. Both are applied with a probability of 0.5%. A model trained on such augmented data can result in better detection accuracy for both low and bright light as well as contrast conditions.

Random patches: A technique that can help when dealing with half-shown fruits[29] in images, either by means of capturing or occlusion, is to apply random patches. This technique drops random patches on the image over various locations. Patches can be placed over half of the fruits, and the other half of the fruit is exposed to deep learning models. Then, models are trained to recognize such cases, even if they are half-covered, yet they will be considered as fruit. So, this technique is helpful in covering edge cases like half-shown and occluded fruits.

4.2.3 Summarizing augmentation

If models are not exposed to bright and low lighting conditions, they can either result in poor detection accuracy or completely miss the target objects in input images. Such a case is depicted in Figure 10.

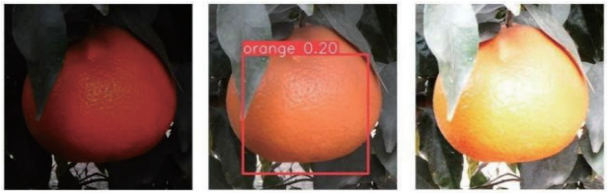

Figure 10 Missed detections without data augmentation

One can clearly see that the detection model has completely failed to detect citrus in both bright and low lighting conditions. After training models on the augmented dataset, the following results can be achieved, as shown in Figure 11.

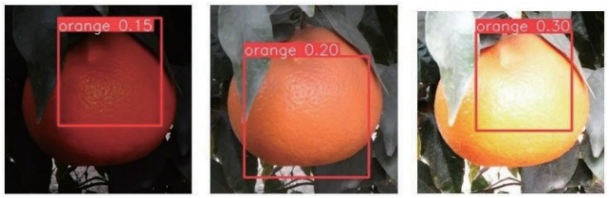

Figure 11 Citrus detected in various lighting conditions after augmentation

This proves that augmentation is useful for object detection models in such scenarios to compensate for both bright and low illumination conditions. The difference in bounding boxes and confidence scores can be improved with proper training on different numbers of epochs.

4.2.4 Data labeling and annotation

The object detection model first needs to be trained on data that is properly labeled and annotated with the target class (the class that needs to be detected during evaluation). In this research, there is one label class named “orange”. The annotations are done in the form of bounding boxes around all the citrus fruits in images. Several tools can be deployed for this task, such as YOLOmark, LabelBox, VoTT, LabelImg, etc. Citrus fruits in images are labeled and annotated using LabelImg.

LabelImg is utilized to label and annotate the images of citrus trees bearing fruit. For YOLO models, the output format is set to YOLO, which results in bounding box coordinates being stored in a .txt file. For Faster R-CNN, the output format can be set to Pascal VOC, which results in coordinates of fruits being stored in an .xml file. COCO format can also be used. Its results are stored in a json format. Figure 12 depicts some of the samples labeled and annotated using LabelImg.

Figure 12 Citrus unlabeled (left) and labeled (right)

Figure 12 shows the labeling of citrus in YOLO format. Coordinates are stored in .txt file shown below in Figure 13. Each row consists of five values. The first value represents the class ID, which is 0 in this case, for representing only citrus. The next two values are for x and y coordinates, respectively. The remaining two values are for width and height, respectively.

4.3 Training of models

After collecting, preprocessing, and augmenting data, models such as YOLOv7 (standard variant), YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium), and Faster R-CNN are trained and evaluated. Before diving into configurations of applied models, the experimental setup is discussed next.

4.4 Validation of models

Right after the training of models, a validation dataset is used to test their performance in detection and time taken to perform object

detection of citrus fruits. A number of different performance metrics are utilized.

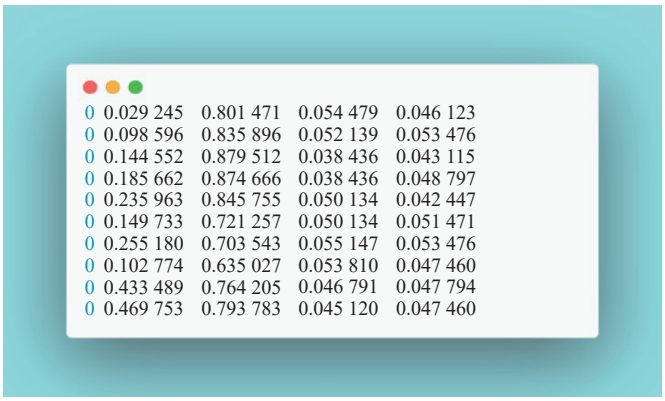

Figure 13 Results of labeling stored inside the text file

4.5 Making predictions

Afterwards, trained models can be used in real environments for object detection of citrus fruits and their counting for the estimation aspect.

5 Experiments and results

In this section, all the details of the experiments and results based on the performance metrics for the proposed Convolutional Neural Network model are discussed.

5.1 Experimental setup

All the experiments for citrus object detection, counting, and yield estimation are performed on a high-performance machine with a dedicated discrete NVIDIA GPU, as GPUs significantly accelerate deep learning workloads[30]. Details of the machine used for performing object detection experiments are listed in Table 1.

Table 1 Specifications of the machine used for experiments

System	Specifications
CPU	Core i5 10th 2.90 GHz
RAM	32 GB DDR4 Memory
GPU	8 GB NVidia Quadro 4000 RTX

5.2 Experimental configurations

This section covers details about YOLOv7, YOLOv8, and Faster R-CNN configurations that are utilized for detection and estimation.

5.2.1 Training of YOLOv7 and YOLOv8 variants

Preprocessed images are passed to YOLO variants for training. All models are applied within their Python-based virtual environments. As each of the applied models differs in terms of required libraries and their versions, there exists a chance of dependency mismatch issues. To avoid such issues, virtual environments are created using the virtualenv tool for experimentation. All the YOLO variants are trained from lower to higher values of epochs and number of images. YOLO experiments are performed with a mixed configuration having change either in epochs or the number of training and validation images. Learning rate of 0.01 is used with the Adam optimizer. The standard variant of YOLOv7 has a size of 72.0 MB with 37 million parameters. YOLOv8n has a size of 6.2 MB with 3.2 million parameters. YOLOv8s variant has a size of 21.5 MB with 11.2 million parameters. The third variant is YOLOv8m with 49.7 MB in size, having 25.9 million parameters.

5.2.2 Training of Faster R-CNN using Detectron2

To train Faster R-CNN using the Detectron2 library, first of all, it is installed via Python's pip package manager. Labeling and annotation are done in Pascal VOC format, resulting in .xml files. Afterwards, a script is used to generate json files for both training and testing sets. At first, Faster R-CNN is trained without data augmentation and evaluated. After that, Faster R-CNN is given another try with an augmented dataset to see the difference and impact on detection accuracy and performance.

The instances of the dataset are registered with Detectron2 using library calls. Consequently, class labels are registered within Detectron2. Following that, a baseline pre-trained model is utilized for transfer learning for the citrus detection task at hand. To read and load the dataset, two service workers are used, helping in the quick loading of the dataset. Learning rate of 0.001 25 is set with Adam optimizer. Initial experiments are done with a minimum number of iterations, batches, and data images.

5.3 Performance metrics

For assessment of detection accuracy, metrics such as precision, recall, and mAP are considered. Formulas are given below. Precision can be defined as a model's measure of quality. For instance, if a model is able to classify a total of 100 samples of the positive class, and 70 of them actually belonged to the positive class, and the remaining ones were incorrectly predicted as positive too, then in such a case, its precision will be 70%.

The formula for precision is as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{TN}} \tag{1}$$

Recall can be defined as a model's measure of quantity. In other words, it is the ratio of actual positives (class samples) that are identified by the used model. For instance, for a dataset of 100 samples for the positive category, if 60 out of 100 are identified as positive samples, then the recall will be 60%. Recall can be calculated by the following formula:

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

The mean Average Precision (mAP) is calculated by calculating the average precision of each class and then averaging it over the number of classes.

$$\text{mAP} = 1/N \sum_{i=1}^{n} \text{AP} \tag{3}$$

The mAP considers trade-offs between recall and precision and considers False Negatives (FN) and False Positives (FP), making it a suitable metric to be used for object detection models.

5.4 Results

This section covers the outcomes of all YOLOv7, YOLOv8 variants, and Faster R-CNN. All the models are compared and evaluated in terms of detection accuracy and inference time.

5.4.1 YOLOv7: Augmentation impact on detections

An initial YOLOv7 experiment is conducted with epochs set to 40 cycles with 400 images for training and 100 images for validation purposes. No data augmentation is involved at this stage of the experiments. The detection results of this experiment are shown in Figure 14.

The detection graphs are depicted in Figure 15.

The resulting precision is 63%, recall is 62%, and mAP is 62% at 50% IoU threshold. Continuing on the second experiment with YOLOv7, the number of epochs has been increased to 50 training cycles with an augmented dataset having 3200 images for training and 800 images for validation. Improvement in the results of detection is depicted in Figure 16.

Figure 14 YOLOv7 detections without augmentation

Figure 16 illustrates that YOLOv7 has shown quite a lot of improvement in detection results with 50 training cycles and an augmented dataset. Graphs for precision, recall, and mAP are shown in Figure 17.

In this second experiment, precision is reported up to 80% with recall up to 77%, and mAP has achieved a value of 80% at 50% threshold. The mean average precision with an IoU range of 50%-95% has reached up to 38%. The noticeable factor in these graphs is that they are not stable and have a lot of fluctuations. The model has struggled a lot to achieve a prediction accuracy (precision) of 80%.

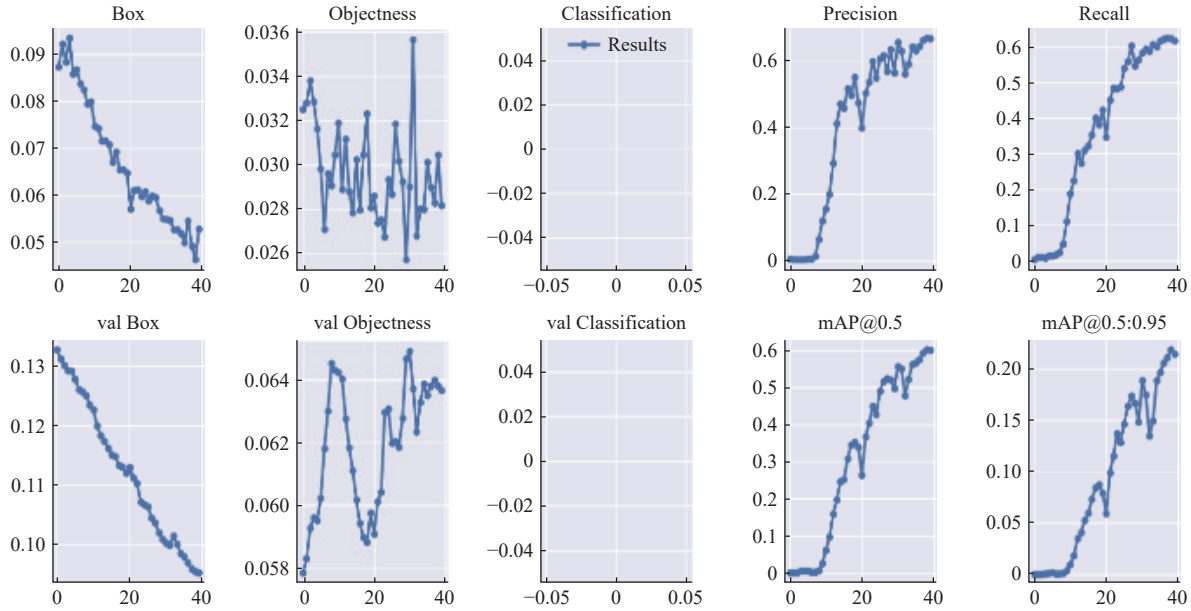

Figure 15 YOLOv7 metrics graphs without augmentation

Figure 16 YOLOv7 detections with augmentation

In all the YOLOv7 conducted experiments, the model has been able to achieve a maximum mAP of 80%. The model has struggled quite a lot to maintain this performance. Results are listed in Table 2.

Table 2 YOLOv7 summary of experiments

Parameter	YOLOv7	
Epochs	40	50
Augmented	No	Yes
Train/Test images	400/100	3200/800
Recall	62%	77%
Precision	63%	80%
mAP	62%	80%

Experimentation shows that the approximate inference time for YOLOv7 in these detections ranged from 10 to 15 ms.

5.4.2 YOLOv8: Augmentation impact on detections

After YOLOv7, another set of improved variants of YOLOv8 is utilized for the aspect of detection of fruits. Experiments with YOLOv8 variants are carried out by starting with the YOLOv8-nano variant. It is a model with a small footprint of model size and number of parameters.

An initial YOLOv8-nano experiment is performed with training images of 400, and the results are evaluated on 100 validation images. The number of training cycles for the initial experiment is set to 40 epochs. The experiment is conducted without data augmentation techniques applied. Detection results are given in Figure 18.

Performance graphs containing precision, recall, and mean average precision at different thresholds of IoU are shown in Figure 19.

Figure 19 depicts the detection results of YOLOv8n without augmentation in terms of precision, recall, and mAP. YOLOv8n has been able to achieve a precision of 75%, a recall of 78%, and an mAP of approximately 80.6% at a 50% IoU threshold. The mean average precision at IoU>50% is close to 50%. The results obtained are more remarkable than the YOLOv7 variant.

The second YOLOv8n experiment is conducted with an augmented dataset containing 3200 images in the training set and 800 images in the validation set, with 50 training cycles of 50 epochs. Results of detection are shown in Figure 20.

Performance graphs representing precision, recall, and mean average precision at various thresholds are shown in Figure 21.

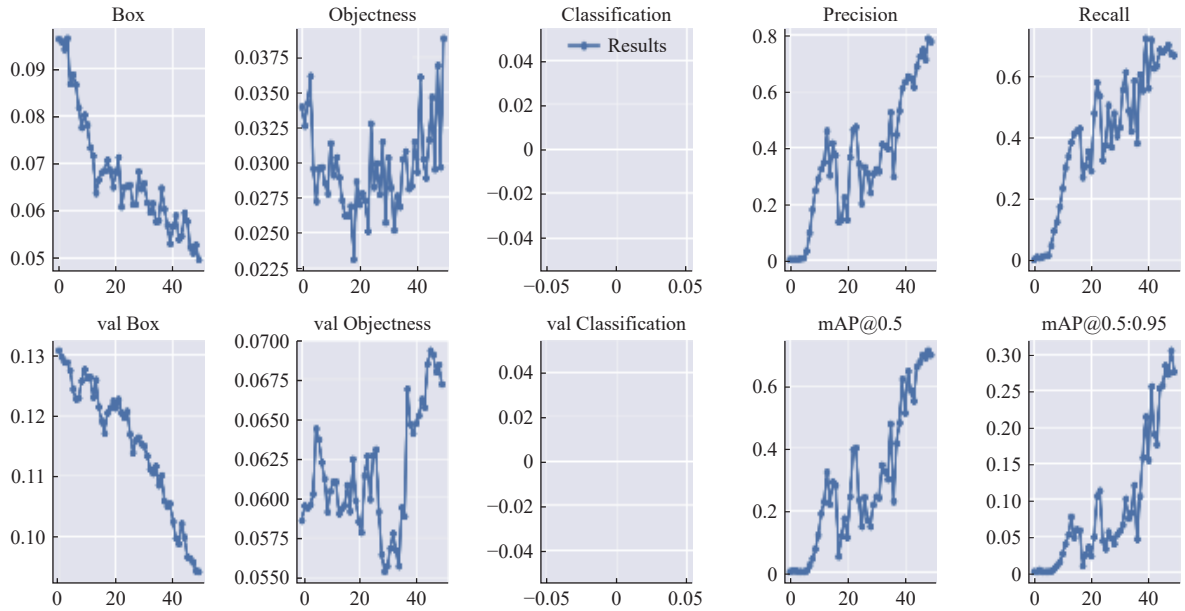

Figure 17 YOLOv7 metrics graphs with augmentation

Figure 18 YOLOv8n detections without augmentation

With the augmented dataset, YOLOv8n has reported approximately 7% improvement in mean average precision at 50% threshold, 8% improvement in recall, and 7% better precision. Inference time of YOLOv8n ranges from 7 to 12 ms on the tested images.

The next model in the line of experiments is the YOLOv8s variant. It has a larger model size in terms of both size and parameters.

The first experiment utilizing YOLOv8s is performed with a similar configuration to the previous experiment of YOLOv8n without augmentation. The reason for experimenting with similar configurations is to have a better comparative analysis at the end of this dissertation work. So, the experiment is performed without augmentation first, on 40 epochs of training cycles with 400 training and 100 validation images on a resolution of 2992×2992. Previous experiments were also performed at a similar resolution. Detection results of YOLOv8s are shown in Figure 22.

Figure 23 illustrates that the baseline YOLOv8s model has been able to achieve a precision of up to 76%, while reported recall and mAP are 84% and 86.6%, respectively. This reported mAP is at 50% IoU threshold. This model performs head-to-head in competition with YOLOv8n with augmented data, although it falls short in precision by 6%.

Results such as precision, recall, and mean average precision are given in Figure 23.

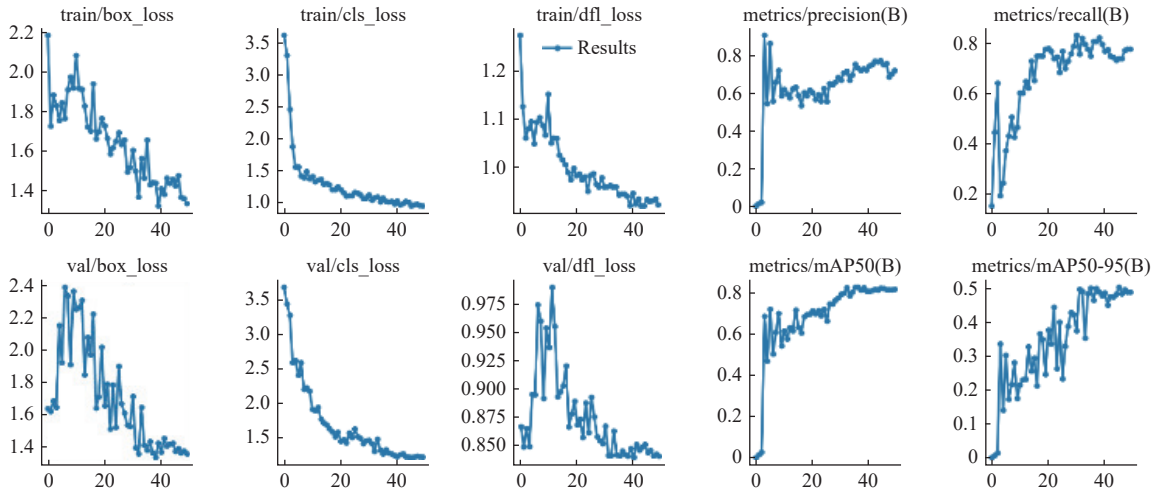

Figure 19 YOLOv8n metrics graphs without augmentation

Figure 20 YOLOv8n detections with augmentation

The second YOLOv8s experiment is conducted on a dataset with augmentation applied to see how much improved performance it attains. Detection results of YOLOv8s with augmentation are depicted in Figure 24.

Mean average precision alongside the results of precision and recall is given in Figure 25.

Figure 25 illustrates that augmentation has certainly improved the metrics of YOLOv8s. After augmentation, the model has reported a precision of almost 85.0%, a recall of 84.0%, and an mAP of 88.2% at 50% IoU threshold. Another key factor, other than these metrics, is the time taken for giving results of detection, i.e., inference time. YOLOv8s has an approximate inference time of 16-20 ms on tested images.

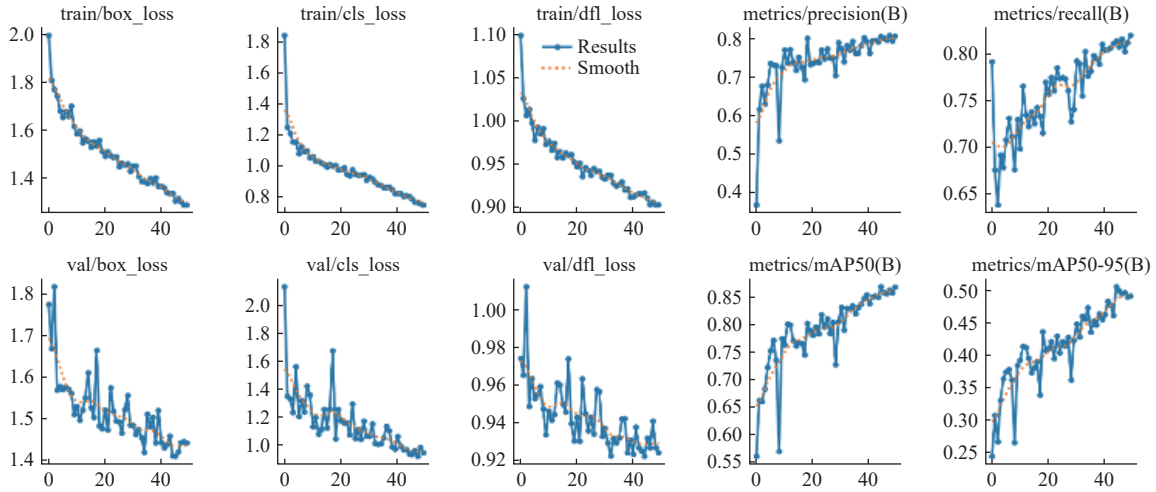

Figure 21 YOLOv8n metrics graphs with augmentation

Figure 22 YOLOv8s detections without augmentation

After applying the YOLOv8s variant, another variant has been applied for performing the object detection task. YOLOv8m is the next model in the line of experiments.

YOLOv8m is applied with similar configurations of 40 epochs of training cycles with 400 images in the training set and 100 images in the validation set without any involvement of data augmentation. The following are the detection results of YOLOv8m as shown in Figure 26.

Results of performance metrics, i.e., precision, recall, and mAP, are shown in Figure 27.

Figure 27 portrays graphs for precision, recall, and mean average precision with reported values of 75%, 83%, and 88.5%, respectively. Experimentation has shown that this model performs competitively with both YOLOv8n and YOLOv8s trained with an augmented dataset.

After the initial YOLOv8m experiment without augmentation, the next one is performed with an augmented dataset containing 3200 training images and 800 validation images with 50 epochs of training cycles. Results are given in Figure 28.

Results of performance metrics are given in Figure 29.

Figure 29 depicts that YOLOv8m with the augmented dataset has shown a slight improvement of approximately 1% in mean average precision at 50% IoU threshold. However, augmentation has certainly helped to gain a better precision of 85% and recall of 85%. It can be concluded that augmentation stabilizes the model's performance. In some cases, it improves results a lot, while in other cases, it does not improve very much but just provides stability and robustness to detecting edge cases. Inference time for YOLOv8m experiments has been shown to be approximately 30 to 40 ms on tested images.

A summary of different YOLOv8 models with and without augmentation is listed in Table 3.

5.4.3 Faster R-CNN: Augmentation impact on detection

After applying a single-stage detector, i.e., YOLO object detection models, we opted for a two-stage detector, such as Faster R-CNN. Single-stage detectors, when given an input image, perform feature extraction and then directly apply classification and localization. However, two-stage detectors work a bit differently. When an input image is given to two-stage detectors, they perform feature extraction, extract object proposals, and then they perform classification and localization. Faster R-CNN has been applied using the backend of Detectron2.

Two experiments of Faster R-CNN have been done, i.e., with and without data augmentation.

The initial experiment is carried out on a similar number of images, 400 training images and 100 validation images, as used for the above YOLO-based experiments. The number of training cycles is set to 40 epochs. Detection results are shown in Figure 30.

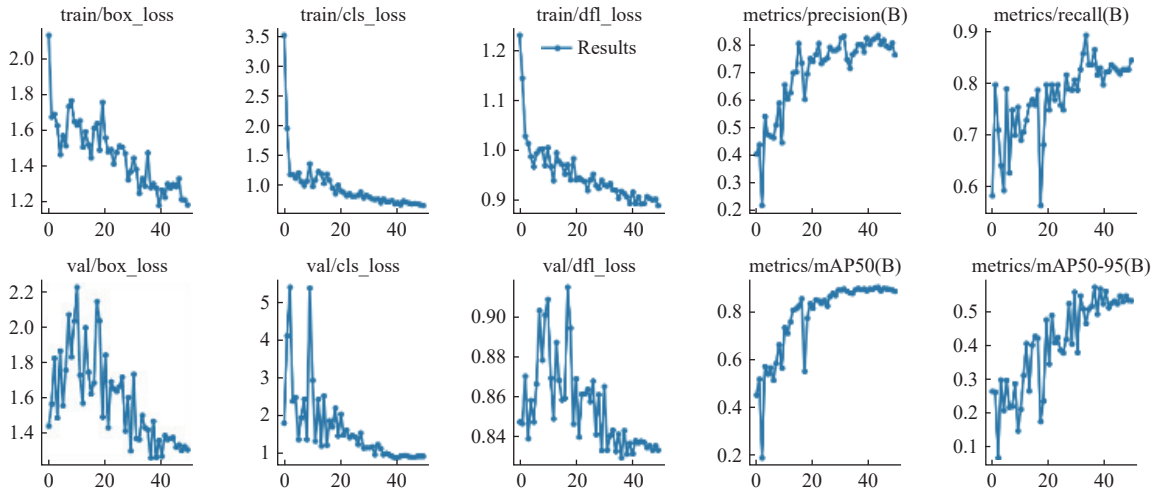

Figure 23 YOLOv8s metrics graphs without augmentation

Figure 24 YOLOv8s detections with augmentation

The initial experiment of Faster R-CNN has been able to reach up to a mean average precision of 74.7%, which is a bit low for proper fruit detection and counting resulting in yield estimation. After experimenting without augmentation, this time Faster R-CNN is trained on an augmented dataset with 3200 images in the training set and 800 images in the validation set. The number of epochs is set to 50 training cycles.

Figure 31 shows the detection results of the second experiment utilizing Faster R-CNN. Figure 31 depicts the results of Faster R-CNN achieved with an augmented dataset. It has achieved a mean average precision of up to 83.4%. A summary of Faster R-CNN results is given in Table 4.

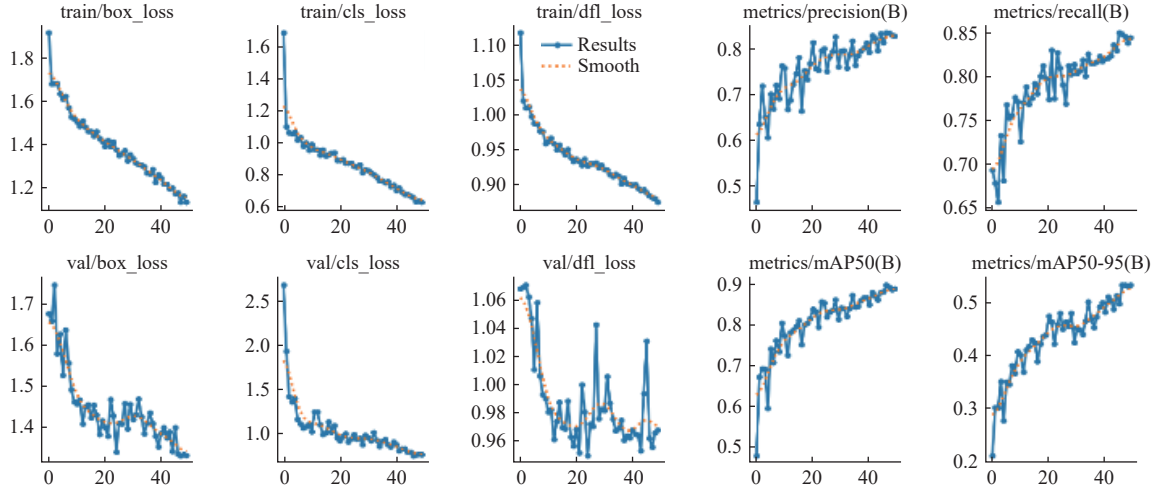

Figure 25 YOLOv8s metrics graphs with augmentation

Table 3 YOLOv8 summary of experiments

Parameter	YOLOv8n		YOLOv8s		YOLOv8m	
Epochs	40	50	40	50	40	50
Augmented	No	Yes	No	Yes	No	Yes
Train/Test images	400/100	3200/800	400/100	3200/800	400/100	3200/800
Recall	78%	86%	84%	84%	83%	85%
Precision	75%	82%	76%	85%	75%	85%
mAP	80.6%	87.1%	86.6%	88.2%	88.5%	89.6%

All the results of experiments performed are listed in Table 5.

For comparison, all the experiments are conducted on the same machine with a discrete Nvidia Quadro 4000 RTX GPU with similar configurations, such as the same number of training and validation images, number of epochs (training cycles), with and without involvement of data augmentation, and a similar learning rate with the Adam optimizer.

Table 4 Faster R-CNN summary of experiments

Parameter	Faster R-CNN	
Epochs	40	50
Augmented	No	Yes
Train/Test images	400/100	3200/800
Precision	72%	80%
Recall	70%	79%
mAP	74.7%	83.4%

Table 3 lists that YOLOv8m with an augmented dataset has shown the highest detection results with 89.6%, almost 90% mAP.

Figure 26 YOLOv8m detections without augmentation

Models performing object detection have inference time for each of their detections on the given input image. Doing experimentations, the inference time of all the applied models has been noted and is given in Table 6.

Table 6 lists that Faster R-CNN has the worst inference time out of all the utilized object detection models. The inference time of YOLOv8n is almost near to being real-time, but it offers lower mAP as compared to YOLOv8s and YOLO8m variants.

5.4.4 Detection via Android application

To provide easy citrus fruit detection, a mobile application for the Android platform has been developed. As shown by the experimentation, YOLOv8m resulted in 90% mAP with a roughly millisecond inference time difference with other test models. So, the weights of the best-performing model (given in .pt file) are first of all converted to an intermediate format called ONNX before it can be deployed to Android devices, as shown in Figure 32. ONNX is further converted to TFLite, a format suitable and highly optimized for mobile devices, focusing on both efficiency and speed. The TFLite variant of YOLOv8m weights is then finally embedded into the Android application for making the detections and estimations. The Android application requires an input image and performs citrus detection on that image, and returns the count of citrus as well as an estimated yield by means of counting and average weight.

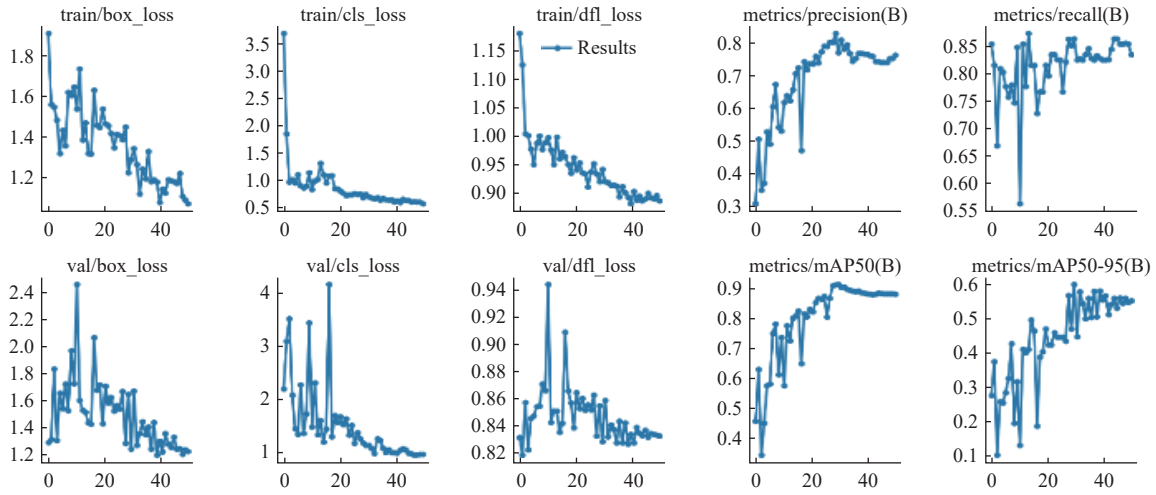

Figure 27 YOLOv8m metrics graphs without augmentation

Table 5 Results of all experiments

Parameter	YOLOv7		YOLOv8n		YOLOv8s		YOLOv8m		Faster R-CNN	
Epochs	40	50	40	50	40	50	40	50	40	50
Augmented	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes
Train\Test images	400\100	3200\800	400\100	3200\800	400\100	3200\800	400\100	3200\800	400\100	3200\800
Precision	63%	80%	75%	82%	76%	85%	75%	85%	72%	80%
Recall	62%	77%	78%	86%	84%	84%	83%	85%	70%	79%
mAP	62%	80%	80.6%	87.1%	86.6%	88.2%	88.5%	89.6%	74.7%	83.4%

Figure 28 YOLOv8m detections with augmentation

Table 6 Inference time of models

Model	Inference time
YOLOv7	10-15 ms
YOLOv8n	7-12 ms
YOLOv8s	16-20 ms
YOLOv8m	30-40 ms
Faster R-CNN	3-10 s

6 Conclusions

This research focuses on the object detection of citrus fruits on trees from the aspect of yield in terms of counting. The higher detection accuracy, the better results for estimated yield, as higher

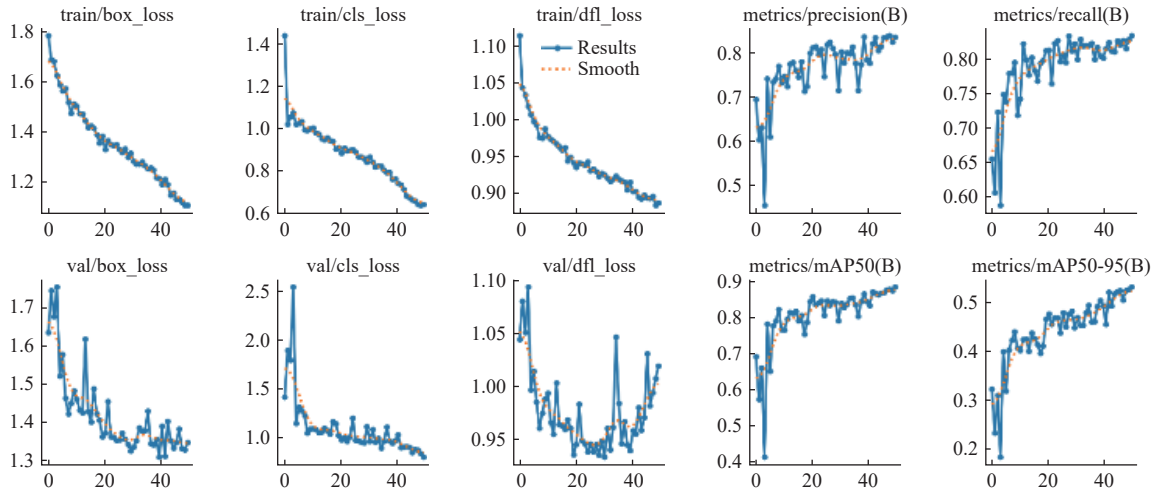

Figure 29 YOLOv8m metrics graphs with augmentation

Figure 30 Faster R-CNN detections without augmentation

Figure 31 Faster R-CNN detections with augmentation

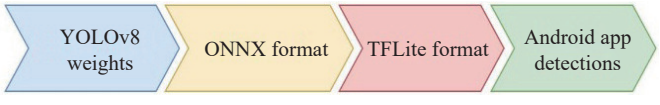

Figure 32 Android application detection workflow

accuracy means near-to-realistic detection. Better detections result in better and more accurate fruit counting, resulting in better estimates. For the purpose of yield monitoring and estimation, a dataset of citrus has been collected from two sites, i.e., Koont Farm of Arid Agriculture University and the National Agriculture Research Center (NARC). The dataset is collected using a Samsung A42 5G with a 48MP High Definition (HD) camera offering a high resolution of 2992×2992 pixels. Following data collection, preprocessing by means of data cleaning, data augmentation, data annotation, and labeling is done. Subsequently, a number of deep learning object detection models, including YOLOv7, YOLOv8 nano, YOLOv8 small, YOLOv8 medium, and Faster R-CNN, are deployed for the detection of fruits and their counting.

After comparison, it is found that Faster R-CNN performed better than YOLOv7 and YOLOv8n in some cases but suffers from a slow inference time of 3-10 s.

Out of all models, YOLOv8m has shown the highest detection results with 90% mAP with an inference time of milliseconds, making it capable of providing high-quality yield estimation results within seconds. The capabilities of the YOLOv8m model are deployed inside a mobile application for Android platform to make detections possible from handheld devices. In the future, such high-performance models can be used to detect diseases having an impact on yield, and to build real-time monitoring systems.

Acknowledgements

The authors extend their appreciation to the Arab Open University for funding this work through AOU research fund (No. AOUKSA524008).

Furthermore, the authors gratefully acknowledge the technical support provided by PSDP-funded project No. 321 “Establishment of National Center of Industrial Biotechnology for Pilot Manufacturing of Bioproducts Using Synthetic Biology and Metabolic Engineering Technologies at PMAS-Arid Agriculture University Rawalpindi”, executed through Higher Education Commission Islamabad, Pakistan.

[References]

[1] Ribeiro H, Abreu I, Cunha M. Olive crop-yield forecasting based on airborne pollen in a region where the olive groves acreage and crop system changed drastically. Aerobiologia, 2017; 33: 473–480.

[2] Dhiab A B, Mimoun M B, Oteros J, Garcia-Mozo H, Domínguez-Vilches E, Galán G, et al. Modeling olive-crop forecasting in Tunisia. Theoretical and Applied Climatology, 2017; 128: 541–549.

[3] Abdel-Hamid O, Mohamed A, Jiang H, Deng L, Penn G, Yu D. Convolutional neural networks for speech recognition. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2014; 22(10): 1533–1545.
[4] Aguilera F, Ruiz-Valenzuela L. A new aerobiological indicator to optimize the prediction of the olive crop yield in intensive farming areas of southern Spain. Agicultural and Forest Meteorology, 2019; 271: 207–213.
[5] Apolo-Apolo O E, Pérez-Ruiz M, Martínez-Guanter J, Valente J. A cloud-based environment for generating yield estimation maps from apple orchards using UAV imagery and a deep learning technique. Front. Plant Sci., 2020; 11: 1086.
[6] Zhu Y L, Wu S S, Qin M J, Fu Z Y, Gao Y, Wang Y Y, et al. A deep learning crop model for adaptive yield estimation in large areas. International Journal of Applied Earth Observation and Geoinformation, 2022; 110: 102828.
[7] Masheswari P, Raja P, Hoang V T. Intelligent yield estimation for tomato crop using SegNet with VGG19 architecture. Scientific Reports, 2022; 12: 13601.
[8] Dhiman P, Kukreja V, Manoharan P, Kaur A, Kamruzzaman M M, Dhaou I, et al. A novel deep learning model for detection of severity level of the disease in citrus fruits. Electronics, 2022; 11(3): 495.
[9] Syed-Ab-Rahman S F, Hesamian M H, Prasad M. Citrus disease detection and classification using end-to-end anchor-based deep learning model. Applied Intelligence, 2022; 52(1): 927–938.
[10] Dhiman P, Kaur A, Hamid Y, Alabdulkreem E, Elmannai H, Ababneh N. Smart disease detection system for citrus fruits using deep learning with edge computing. Sustainability, 2023; 15(5): 4576.
[11] Kang X Y, Huang C P, Zhang L F, Zhang Z, Lyu X. Downscaling solar-induced chlorophyll fluorescence for field-scale cotton yield estimation by a two-step convolutional neural network. Computers and Electronics in Agriculture, 2022; 201: 107260.
[12] Zhang X H, Toudeshki A, Ehsani R, Li H L, Zhang W F, Ma R J. Yield estimation of citrus fruit using rapid image processing in natural background. Smart Agricultural Technology, 2022; 2: 100027.
[13] Zhou X B, Kono Y, Win A, Matsui T, Tanaka T S T. Predicting within-field variability in grain yield and protein content of winter wheat using UAV-based multispectral imagery and machine learning approaches. Plant Production Science, 2021; 24(2): 137–151.
[14] Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: Unified, real-time object detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, 2016; pp.779–788. doi: 10.1109/CVPR.2016.91.
[15] Xing S L, Lee M. Classification accuracy improvement for small-size citrus pests and diseases using bridge connections in deep neural networks. Sensors, 2020; 20(17): 4992.
[16] Wu Y, Kirillov A, Massa F, Lo W Y, Girshick R. Detectron2 Releases. Available: https//github.com/facebookresearch/detectron2/releases. Accessed on [2021-03-03].
[17] Nevavuori P, Narra N, Linna P, Lipping T. Crop yield prediction using multitemporal UAV data and spatio-temporal deep learning models. Remote Sensing, 2020; 12(23): 4000.
[18] Zhang W L, Wang J Q, Liu Y X, Chen K Z, Li H B, Duan Y L, et al. Deep-learning-based in-field citrus fruit detection and tracking. Horticulture Research, 2022; 9: uhac003.
[19] Darwin B, Dharmaraj P, Prince S, Popescu D E, Hemanth D J. Recognition of bloom/yield in crop images using deep learning models for smart agriculture: A review. Agronomy, 2021; 11(4): 646.
[20] Kalantar A, Edan Y, Gur A, Klapp I. A deep learning system for single and overall weight estimation of melons using unmanned aerial vehicle images. Computers and Electronics in Agriculture, 2020; 178: 105748.
[21] Lin P Y, Li D H, Jia Y H, Chen Y Y, Huang G W, Elkhouchlaa H, et al. A novel approach for estimating the flowering rate of litchi based on deep learning and UAV images. Front. Plant Sci., 2022; 13: 966639.
[22] Gour M, Jain S, Kumar T S. Residual learning based CNN for breast cancer histopathological image classification. International Journal of Imaging Systems and Technology, 2020; 30(3): 621–635.
[23] Gour M, Jain S, Agrawal R. DeepRNNetSeg: Deep residual neural network for nuclei segmentation on breast cancer histopathological images. In: Computer Vision and Image Processing. CVIP 2019. Communications in Computer and Information Science, 2019; 1148: 243–253.
[24] Vijayakumar V, Ampatzidis Y, Costa L. Tree-level citrus yield prediction utilizing ground and aerial machine vision and machine learning. Smart Agricultural Technology, 2023; 3: 100077.
[25] Gavahi K, Abbaszadeh P, Moradkhani H. DeepYield: A combined convolutional neural network with long short-term memory for crop yield forecasting. Expert Systems with Applications, 2021; 184: 115511.
[26] Mendez V, Perez-Romero A, Sola-Guirado R, Miranda-Fuentes A, Manzano-Agugliaro F, Zapata-Sierra A, et al. In-field estimation of orange number and size by 3D laser scanning. Agronomy, 2019; 9(12): 885.
[27] Stateras D, Kalivas D. Assessment of olive tree canopy characteristics and yield forecast model using high resolution UAV imagery. Agriculture, 2020; 10(9): 385.
[28] Xia X, Chai X J, Zhang N, Zhang Z, Sun Q X, Sun T. Culling double counting in sequence images for fruit yield estimation. Agronomy, 2022; 12(2): 440.
[29] Khalid S, Oqaibi H M, Aqib M, Hafeez Y. Small pests detection in field crops using deep learning object detection. Sustainability, 2023; 15(8): 6815.
[30] Aqib M, Mehmood R, Alzahrani A, Katib I, Albeshri A, Altowaijri S M. Smarter traffic prediction using big data, in-memory computing, deep learning and GPUs. Sensors, 2019; 19(9): 2206.