Signal, Image and Video Processing (2022) 16:841–847
https://doi.org/10.1007/s11760-021-02024-y

ORIGINAL PAPER

# Leaf-based disease detection in bell pepper plant using YOLO v5

**Midhun P. Mathew**[1] · **Therese Yamuna Mahesh**[1]

Received: 17 May 2021 / Revised: 12 August 2021 / Accepted: 5 September 2021 / Published online: 27 September 2021


**Abstract**
In the era of twenty-first century, artificial intelligence plays a vital role in the day to day life of human beings. Now days it has been used for many application such as medical, communication, object detection, object identification, object tracking. This paper is focused on the identification of diseases in bell pepper plant in large fields using deep learning approach. Bell Pepper farmers, in general do not notice if their plants are infected with bacterial spot disease. The spread of the disease usually causes a decrease in the yield. The solution is to detect if bacterial spot disease is present in the bell pepper plant at an early stage. We do some random sampling of few pictures from different parts of the farm. YOLOv5 is used for detecting the bacterial spot disease in bell pepper plant from the symptoms seen on the leaves. With YOLO v5 we are able to detect even a small spot of disease with considerable speed and accuracy. It takes the full image in a single instant and predicts bounding boxes and class probability. The input to the model is random picture from the farm by using a mobile phone. By viewing the output of the program, farmers can find out whether bacterial spot disease has in any way affected the plants in their farm. The proposed model is very useful for framers, as they can identify the plant diseases as soon as it appears and thus, do proper measures to prevent the spread of the disease. The motive of this paper is to come up with a method of detecting the bacterial spot disease in bell pepper plant from pictures taken from the farm.

**Keywords** Deep learning · YOLOv5 · Labelimg · Artificial intelligence · Object detection · Plant diseases identification · PyTorch

## 1 Introduction

In this paper, we are focusing on the detection of diseases in the bell pepper plant using a deep learning approach such as YOLOv5. Figure 1 shows a bell pepper farm from which we can see that the detection of diseases in plants from the field is a herculean task. Manual identification is difficult as it would take a lot of time to inspect the plants in the entire farm. Also accuracy of such prediction is low and it takes a long time to find out the diseased leaves. So there should be a model which helps us to identify the diseases with more dependency and accuracy. Our paper describes a model which can help the framers to identify the disease with low additional costs. The main disease that affects the bell pepper plant is the bacterial spot. It is a disease that mainly affects the leaf of bell pepper. Figure 2a shows a healthy leaf, and Fig. 2b shows a leaf infected by bacterial spot disease. The paper is organized into three sections. In Sect. 1, the pre-processing operations are done on image dataset and the training and testing datasets are prepared. Section 2 deals with Training and testing the data. Section 3 is about identifying the healthy and diseased part in the randomly selected leaf that is used as a test image.

✉ Therese Yamuna Mahesh
thereseyamunamahesh@gmail.com

[1] Department of Electronics and Communication Engineering, Amaljyothi College of Engineering, Kanjirapally, Kerala, India

## 2 Literature review

Object detection is defined as the process of identification of a particular object in the image or video. Nowadays, the object detection techniques are widely used in computer vision. There are different methods used for object detection in deep learning and machine learning.

Disease detection in plant leaves can be seen as an object detection problem [1]. Several detection methods were studied from literature like SIFT [2], Haar [3], HOG [4] and finally convolutional features[5]. After extracting the features, localizers [6] or classifiers [7, 8] were used to identify objects in the feature space. Many applications were imple-

**Fig. 1** Bell Pepper farm


**Fig. 2** **a** Healthy leaf **b** Diseased leaf

mented using the YOLO algorithm [1, 9–11], YOLOv2 [12–15]. In our paper, we are focusing only on a deep learning method using YOLOv5.

Joseph Redmon, in the paper "You Only Look Once: Unified, Real-Time Object Detection" [16] explains about the importance of YOLO (you only look once) in the object detection scenario. YOLO can be defined as single neural network that can predict multiple boxes and class probability for the boxes. While training, the YOLO network takes the entire image. In his paper he mentions about the advantages and challenges of the YOLO algorithm. He compares YOLO with other traditional deep learning algorithms. While comparing, he mentions that YOLO is fast because of treating the object detection as simple regression problem. There is no need for a complex pipeline. He also mentions that the base network has a speed of 45 FPS(frames per second). Faster versions can have the speeds of more than 150 FPS. Mean average precision is twice more than other commonly used detection methods. Second advantage is that YOLO uses the entire image during training and testing, so, it encodes contextual information about the classes and appearance. Back ground errors as compared to other traditional deep learning algorithm like Faster R-CNN (Regions with CNN) [26] is less than half. There are few disadvantages also. It can detect the image faster but it lacks the precision. Secondly the accuracy is lagging behind the state-of-the-art detection systems. Third disadvantage is that the model learns to predict bounding box from given data. So when a general image or images with different aspect ratio or if the configuration of image changes, then prediction becomes difficult.

Joseph Redmon in the paper "YOLO9000: Better, Faster, Stronger" [17] comes up with another model which can overcome the defects seen in the previous model. YOLO9000 which is also known as YOLOV2, helps to overcome problems of YOLO such as error analysis in localization, relatively low recall. To overcome the above problems of YOLO, YOLOV2 introduced batch normalization, high-resolution convolution with anchor box, dimension cluster, direct location prediction, multi scale training and fine-grained features. This makes YOLOV2 better. Darknet-19 is used in YOLOV2. Hierarchical classification, Joint classification and detection, makes YOLOV2 more stronger.

Joseph Redmon, in the paper "YOLOV3: An Incremental Improvement" [18] comes up with another YOLO model which is better than YOLOV2. While comparing with traditional methods YOLOV3 has an accuracy which is three times, as compared to traditional networks. But still some of the feature does not work like, anchor box x, y, offset predictions, linear x, y predictions instead of logistic, focal loss, dual IOU (Intersection over Union) thresholds and ground truth assignment.

The next version of YOLO is implemented by Alexey Bochkovskiy in the paper "YOLOv4: Optimal Speed and Accuracy of Object Detection" [19]. This new version increases the speed of the object detection. It also increases the YOLOv3 mean average precision and frames per second by 10% and 12%, respectively. The author also says that YOLOv4 is suitable for single GPU training, this being done by introducing new method for data augmentation, optimal hyper parameter and genetic algorithms.

Within a few months after the publication of YOLOv4 there comes YOLOv5 by Ultralytics LLC team. Guanhao Yang, in the paper "Face Mask Recognition System with YOLOv5 Based on Image Recognition" [20], mentions about the YOLOv5. In our paper, we have developed an application for the detection of diseases in the bell pepper plant using YOLOv5.

## 3 Methodology

In this section we are explaining about the training and.

testing models. Figure 3 shows the block diagram of training and testing of our model.

### 3.1 Data set collection and labelling

The major difficulties that we faced while doing the project is the collection of data set. Forming a primary dataset using the

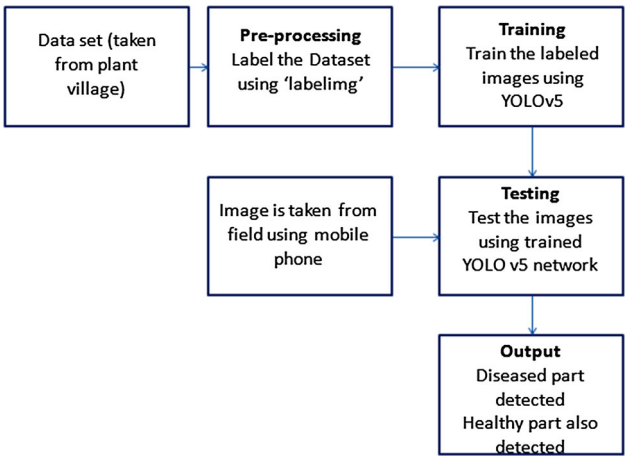


**Fig. 3** Proposed Block Diagram

**Table 1** Two classes used for identification

| Class no | Class |
|---|---|
| 0 | Healthy part |
| 1 | Bacterial spot |

**Table 2** Number of image for test/train

| Class | Training samples | Testing samples | Total samples |
|---|---|---|---|
| Healthy part | 1500 | 500 | 2000 |
| Bacterial spot | 1500 | 500 | 2000 |

camera and taking the picture is a time consuming and difficult task. So we used the secondary dataset which is available in Kaggle site. After downloading the dataset (plant village), the next process was to label the data set. In order to label the image, we used labelimg tool. To install 'labelimg' we used 'pip install labelimg' in python. After the successful installation, we started to do image labelling (drawing the bounding box and labelling the class). After successfully labelling the image we get the output as text file and a class file. Table 1 represent the classes of our project i.e. (the class file).

The text file consists of 5 decimal values, the first value represents the class of the bounding box, next is centre x then centre y, followed by the width and the height. Centre x and centre y is defined as the centre point of the bounding box [16]. These values are normalized in the range of 0 and 1. This is done by dividing the values by width and height of the image. This is done because it is easy to predict the values in the range of 0 and 1 for the network, rather than a random number. Number of lines in the text file depends on the number of bounding boxes drawn. The labelling is done for both testing and training datasets. Table 2 represents the number of images taken for training and testing for each class.

## 3.2 Training Labelled Images using YOLOv5

YOLOv5 [20] is the latest version of YOLO, which has high accuracy in detection and inference speed. YOLOv5 has a weight file which is of small size, 90% smaller than YOLOv4. Thus it can be used in embedded device for real-time detection. Compared to other YOLO versions, YOLOv5 has high detection accuracy, light weight characteristic and fast detection time. In the case of plant disease detection, accuracy and efficiency is very important. YOLOv5 architecture helps to improve the detection of diseases in bell pepper plant. The architecture consists of four different models such as YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x. The difference in the four architectures is in the feature extraction module and the convolutional kernel of the network. Other difference is the size of the model and the amount of model parameters which is different for the four different architectures.

### 3.2.1 Training a custom YOLOv5 model

The training of a custom YOLOv5 model consists of the following steps [20, 21]

1. Setting up the YOLO environment
   Clone the repository of YOLOv5 from GitHub. YOLOv5 runs on Torch, and this can be easily run using Kaggle or Colab. This will create a folder called 'YOLOv5 on the machine. This folder will contain the specific YOLO directory structure and the pre-trained weights for the model are also included.
2. Setting up the data and the directory structure
   A folder called data is set up, at the same level as YOLO folder. Inside the data folder, create a new folder for Images and Labels. Inside Images and Labels create folders for Train and Test. Inside the folder data/labels/train and in the data/labels/test, the labels have to be uploaded. The name of the labels file has to be the same as the image file, but with ".txt" extension. The bounding boxes have to be listed as one bounding box per line. The following are listed in the bounding box. The first value represents the class number. The value is always zero if there is only one class. In second position is the standardized center pixel of the bounding box in terms of width. The third position represents the standardized center pixel of the bounding box in terms of height. Then comes the standardized width of the bounding box followed by the standardized height of the bounding box.
   To standardize, we divide the number of pixels by the total number of pixels of the image. So the standardized value will be (0.2, 0.3, 0.5, 0.6) for a bounding box on pixel (20, 30) with a width of $50 \times 60$ on a picture of size (100, 100). The number of bounding boxes in one image

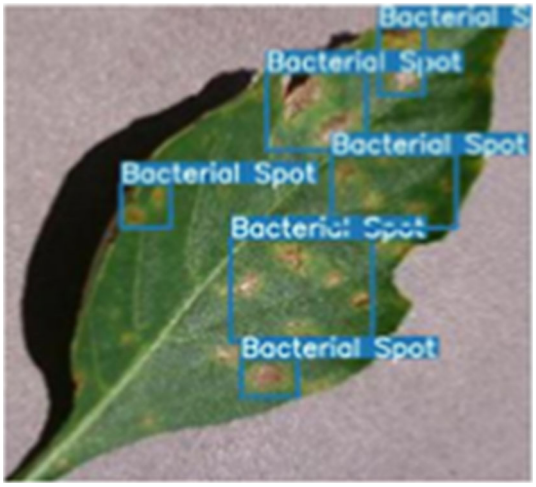


**Fig. 4** Bacterial spot detected from test dataset


**Fig. 5** Bacterial spot and healthy part

is represented by the number of lines of the label file and the number of images is represented by the number of label files.

3. Setting up the YAML configuration files

To start training a YOLOv5 model you need two YAML files. The first YAML specifies the location of the training data, the location of the test data, the number of classes, i.e. the types of the objects that are to be detected and the names of the objects corresponding to those classes. The second YAML file includes parameters, anchor boxes, YOLOv5 backbone and YOLOv5 head.

4. Training the model

Training the model is done by executing the train.py command from the notebook. Hyper parameters like image size, number of epochs and batch size can be specified. A subfolder in YOLOv5 will contain the weights of this model. Detection of the diseases on the leaf can be done by using the detect.py command.

After the training process, a subfolder is created in YOLOv5. The path of the subfolder can be expressed as YOLOv5/run/train/ exp.no./weights/ last.pt and size of the weights is only 14.4 MB if we are using YOLOv5s. yaml file.size of the weight file will be changed according to the yaml file that we use. Following figures will be generated after the training is done. Figures 4 and 5 show the result of test data from the test dataset, and Fig. 6 shows the output of a random infected leaf.

With each training batch, YOLOv5 passes training data through a data loader, which augments data online. The data loader makes three kinds of augmentations: scaling, colour space adjustments, and mosaic augmentation. The most novel of these being mosaic data augmentation, which combines four images into four tiles of random ratio. Mosaic augmentation is useful for the bell pepper disease detection dataset, helping the model to address the 'small object problem' where small symptoms of the bacterial spot are detected, Fig. 6 shows bacterial spot and healthy part detected from Google image.

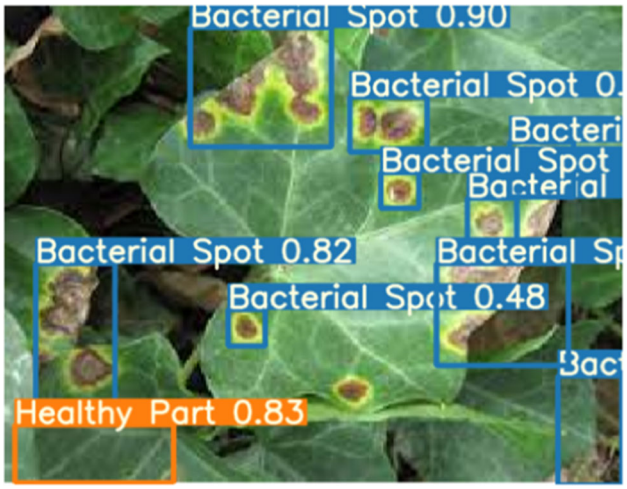


**Fig. 6** Bacterial Spot and healthy part detected from Google image

## 4 Results—performance measures

The graphs shown in Fig. 7 show the plot of mAP(mean average precision), Precision and Recall for train data consisting of 1500 samples [22].

The graphs in Fig. 7 shows the mAP at 0.5 and mAP in the interval 0.5 to 0.95. The graphs are plotted based on the precision and recall and IoU (Intersection over Union) thresholds

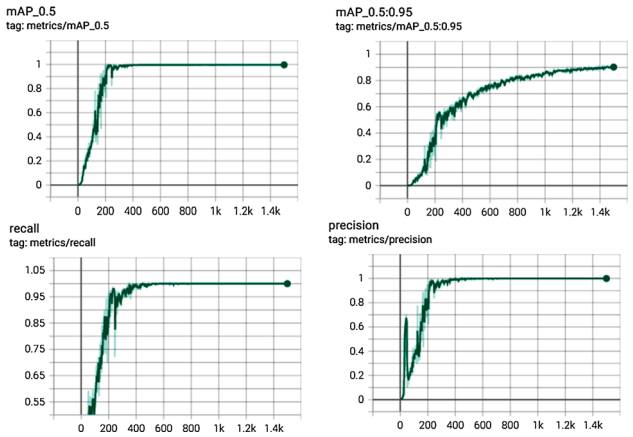


**Fig. 7** Plot of mAP, precision and recall after Training the YOLO network

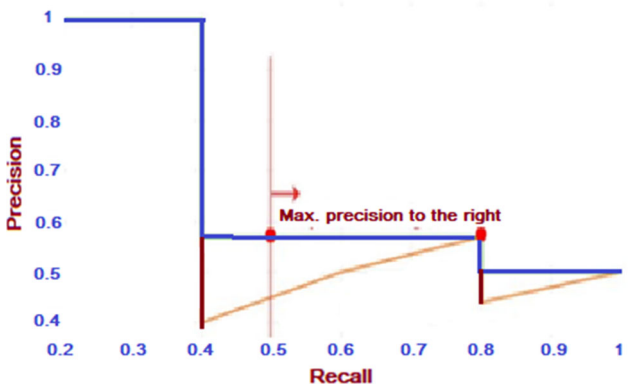


**Fig. 8** Calculation of maximum average precision [23]

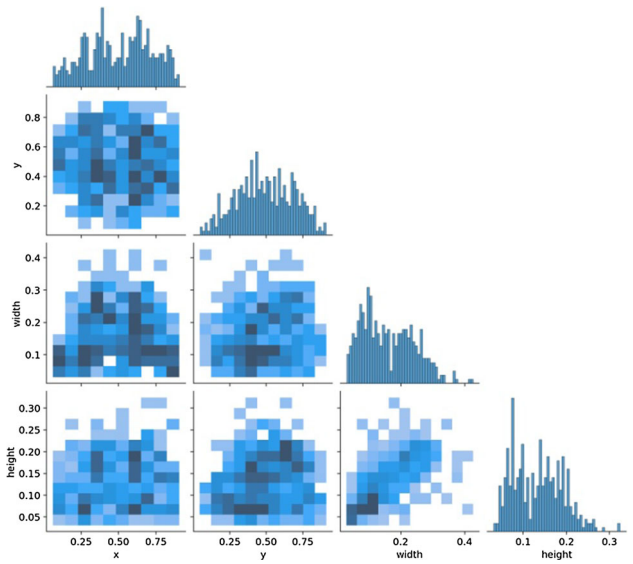


**Fig. 9** Correlogram

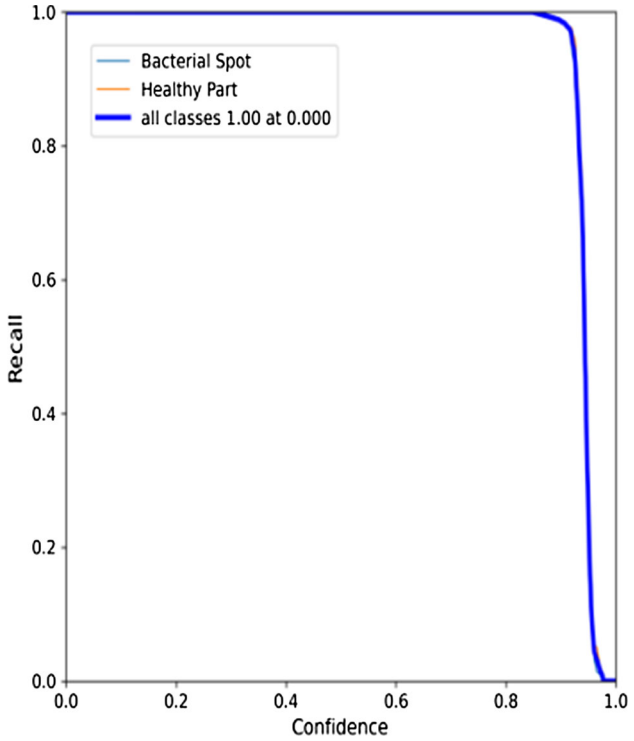


**Fig. 10** Confidence Vs R Curve

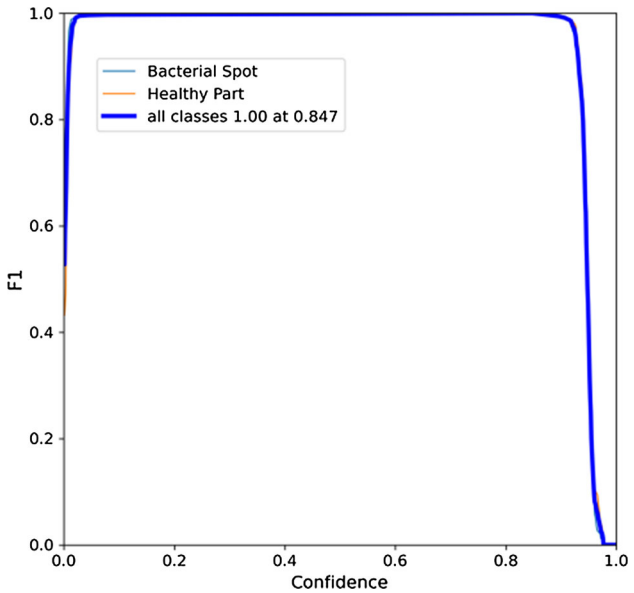


**Fig. 11** Confidence Vs F1 curve

$$\text{Precision(P)} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1}$$

$$\text{Recall(R)} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

**Table 3** Comparison of YOLOv4 and YOLOv5

| YOLO version | Batch | Inference Time (ms) | Maxval mAP@0.5 | Size of weight file ( mb) | Training time comparison (min) |
|---|---|---|---|---|---|
| YOLOv4 | 1 | 22 | .907 | 250 | 210 |
| YOLOv5s | 1 | 20 | .907 | 27 | 20 |

We replace the precision value for recall $R$ with the maximum precision for any recall $\geq$ R as shown in Fig. 8. A prediction is positive if output is greater than threshold K [22, 23]

$$\text{mAP} = \frac{1}{11}\sum_{\text{R}\in(0,0.1.0.2,.....1.0)} \text{P}_{\text{interpolation}}(\text{R}) \tag{3}$$

TP (True Positive) = Number of correctly identified instances of Bacterial Spot and Healthy part of the leaf.

FP (False Positive) = Number of incorrectly identified instances of Bacterial Spot and healthy part of the leaf.

FN (False Negative) = Number of unidentified diseased part and healthy part.

N = Number of IoU (Intersection over union) thresholds.

K = IoU threshold,C = Class, P(k) = precision, R(k) = recall.

Exploratory data analysis is best done with the help of a correlogram. The relationship of the whole dataset can be visualized in a glimpse. The relationship of the bounding boxes for the two classes is shown in the correlogram, Fig. 9. A high quality correlogram is obtained using seaborn.

Figure 10 shows the R curve showing the confidence Vs Recall, and Fig. 11 shows the Confidence Vs F1 curve.

where

$$F1 = \frac{2}{\frac{1}{precision} + \frac{1}{Recall}} \tag{4}$$

The values in Table 3 are obtained by running the algorithm on the same GPU using the same dataset.

Figure 12 shows the comparison of the percentage of correct detection by different algorithms [7, 24, 25]. The graph shows the confidence score obtained while using R-CNN [6, 26], RFCN (Region Based Fully Convolutional Networks) [16], SSD (Single Shot Multibox Detector) [16], YOLOv3 [27], YOLO v4 [28], YOLOv5 (our method).

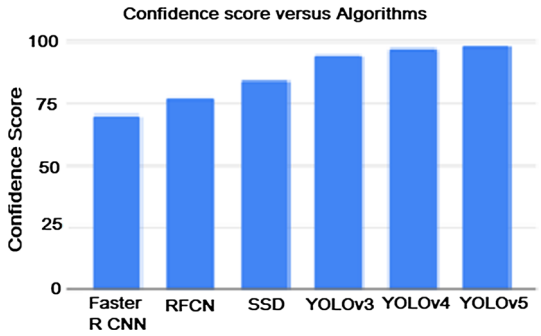


**Fig. 12** Comparison of different algorithms

## 5 Conclusion

Bacterial spot disease in the bell pepper plant is successfully detected at an early stage in our procedure. By using YOLOv5 the results obtained are more accurate compared to other models as shown in Fig. 1). As the size of the weight file is only 27mb as mentioned in Table 3, hardware implementation becomes much easier. As the comparative training time using the same GPU and dataset is only 9.5% of the previous model using YOLOv4, results are obtained faster. Future scope includes extension of disease detection to other diseases also that are likely to affect bell pepper plant. Thus the method described in this paper accomplishes faster results and better accuracy than the previous versions of YOLO and is suitable to improve yield in farms by detecting and identifying plant diseases.

**Authors' Contributions** All authors contributed to the study conception and design and both the authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

**Funding** No funding was received for conducting this study.

**Availability of data and material** Image dataset used in this research is available online.

**Code availability** Not applicable.

## Declarations

**Conflict of interest** The authors declare that they have no conflict of interest.

## References

1. Khan, S., Tufail, M., Khan, M.T., et al.: Deep learning-based identification system of weeds and crops in strawberry and pea fields for a precision agriculture sprayer. Precision Agric. (2021). https://doi.org/10.1007/s11119-021-09808-9
2. Lowe, D. G.: Object recognition from local scale-invariant features. In: The Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999, vol. 2, pp. 1150–1157. IEEE, (1999)

3. Papageorgiou, C. P., Oren, M., Poggio, T.: A general framework for object detection. In: Sixth International Conference on Computer Vision, 1998, pp. 555–562. IEEE, (1998)
4. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 1, pages 886–893. IEEE, (2005)
5. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: a deep convolutional activation feature for generic visual recognition. arXiv preprint http://arxiv.org/abs/1310.1531, (2013)
6. Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y.: Overfeat: Integrated recognition, localization and detection using convolutional networks. CoRR, http://arxiv.org/abs/1312.6229, (2013)
7. Sultana, F., Sufian, A., Dutta, P.: A review of object detection models based on convolutional neural network. In: Mandal J., Banerjee S. (eds) Intelligent Computing: Image Processing Based Applications. Advances in Intelligent Systems and Computing, vol. 1157. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-4288-6_1
8. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 580–587. IEEE, (2014)
9. Chen, B., Miao, X.: Distribution line pole detection and counting based on YOLO using UAV inspection line video. J. Electr. Eng. Technol. **15**, 441–448 (2020). https://doi.org/10.1007/s42835-019-00230-w
10. Wageeh, Y., Mohamed, H.E.D., Fadl, A., et al.: YOLO fish detection with Euclidean tracking in fish farms. J. Ambient Intell. Human Comput. **12**, 5–12 (2021). https://doi.org/10.1007/s12652-020-02847-6
11. Zhao, J., Li, C., Xu, Z., et al.: Detection of passenger flow on and off buses based on video images and YOLO algorithm. Multimed Tools Appl. (2021). https://doi.org/10.1007/s11042-021-10747-w
12. Hou, X., Zhang, Y., Hou, J.: Application of YOLO V2 in construction vehicle detection. In:Meng, H., Lei, T., Li, M., Li, K., Xiong, N., Wang, L. (eds) Advances in Natural Computation, Fuzzy Systems and Knowledge Discovery. ICNC-FSKD 2020. Advances in Intelligent Systems and Computing, vol. 1348. Springer, Cham (2021) https://doi.org/10.1007/978-3-030-70665-4_135
13. Yang, S., Bo, C., Zhang, J., Wang, M.: Vehicle logo detection based on modified YOLOv2. In: Lu, H., Yujie, L. (eds) 2nd EAI International Conference on Robotic Sensor Networks. EAI/Springer Innovations in Communication and Computing. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-17763-8_8
14. Sujee, R., Shanthosh, D., Sudharsun, L.: Fabric defect detection using YOLOv2 and YOLOv3 Tiny. In: Chandrabose, A., Furbach, U., Ghosh, A., Kumar, M. A. (eds) Computational Intelligence in Data Science. ICCIDS 2020. IFIP Advances in Information and Communication Technology, vol 578.Springer, Cham (2020). https://doi.org/10.1007/978-3-030-63467-4_15
15. Saranya, K. C., Thangavelu, A., Chidambaram, A., Arumugam, S., Govindraj, S.: Cyclist detection using tiny YOLO v2. In: Das, K., Bansal, J., Deep, K., Nagar, A., Pathipooranam, P., Naidu, R. (eds) Soft Computing for Problem Solving. Advances in Intelligent Systems and Computing, vol 1057. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-0184-5_82
16. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You Only Look Once: Unified, Real-Time Object Detection. published in http://arxiv.org/abs/2004.10934v1 [cs.CV] 09 May 2016
17. Redmon, J., Farhadi, A.: "YOLO9000: Better, Faster, Stronger" published in http://arxiv.org/abs/2004.10934v1 [cs.CV] 25 Dec 2016
18. Redmon, J., Farhadi, A.: "YOLOv3: An Incremental Improvement" published in http://arxiv.org/abs/2004.10934v1 [cs.CV]
19. Bochkovskiy, A.: "YOLOv4: Optimal Speed and Accuracy of Object Detection" published in http://arxiv.org/abs/2004.10934v1 [cs.CV] 23 Apr 2020
20. Yang, G., Feng, W., Jin, J., Lei1, Q., Li, X., Gui, G., Wang, W.: Face Mask Recognition System with YOLOV5 Based on Image Recognition. In: 2020 IEEE 6th International Conference on Computer and Communications
21. Yan, B., Fan, P., Lei, X., Liu, Z., Yang, F.: A Real-Time Apple Targets Detection Method for Picking Robot Based on Improved YOLOV5" MDPI Remote Sens. 2021, 13, 1619. https://doi.org/10.3390/rs13091619
22. Zhang, E., Zhang, Y.: Average Precision. In: LIU L., ÖZSU M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-39940-9_482(2009)
23. Towards Data Science- Evaluating Performance of an Object Detection Model https://towardsdatascience.com/evaluating-performance-of-an-object-detection-model-137a349c517b
24. Saeidi, M., Ahmadi, A.: High-performance and deep pedestrian detection based on estimation of different parts. J. Supercomput. **77**, 2033–2068 (2021). https://doi.org/10.1007/s11227-020-03345-4
25. Brahimi, M., Arsenovic, M., Laraba, S., Sladojevic, S., Boukhalfa, K., Moussaoui, A.:"Deep learning for plant diseases: detection and saliency map visualisation. In: Human and Machine Learning. Eds. J. Zhou and F. Chen (Cham, Switzerland: Springer International Publishing), pp. 93–117 (2018)
26. Girshick, R. B.: Fast R-CNN. CoRR, http://arxiv.org/abs/1504.08083, (2015)
27. Redmon, J., Farhadi, A.:Yolo V3: An incremental improvement. http://arxiv.org/abs/1804.02767 [cs], pp. 1–6 (2018)
28. P Mathew, M., Mahesh, T. Y.: "Leaf Based Disease Detection of Bell Pepper plant Using Yolo V4" Journal of Huazhong University of Science and Technology (ISSN-1671–4512) Volume-50, Issue-5