Detection of plant diseases in an industrial greenhouse: Development, Validation & Exploitation

* Laboratoire Recherche Informatique Maisonneuve (LRIMa), Montreal, Canada
† Esiee, France
Yassine Lakhdari†*, Enric Soldevila*, Jihene Rezgui*

***Abstract*—The effective detection of plant diseases is crucial for the optimal management of agricultural systems. In this paper, we present our contributions in the context of detecting plant diseases in an industrial greenhouse [1], focusing specifically on tomatoes. Our main objectives are to develop and validate a detection system using the YOLOv8 model and to explore its potential for practical application in a real-world setting. To facilitate our research, we introduce a novel dataset comprising images of tomato leaves affected by various diseases. This dataset serves as a valuable resource for training and evaluating our detection model. We employ the YOLOv8 architecture, a state-of-the-art object detection framework, and experiment with different parameters to assess its performance in accurately detecting diseased areas on tomato leaves. Through extensive experimentation, we compare the performance of the YOLOv8 model using various parameters, such as different training strategies, data augmentation techniques, and hyperparameter configurations. The results provide insights into the optimal settings for achieving high detection accuracy and robustness. Furthermore, we demonstrate the practical utility of our developed model by conducting a real-life implementation within an industrial greenhouse. This exemplifies the integration of our detection system into an operational environment, showcasing its potential to assist greenhouse operators in early disease detection, monitoring, and decision-making processes. Our preliminary findings demonstrate promising disease detection capabilities on tomato leaves inside greenhouses, achieving an mAP50 score of 0.8 using our best model. Although there is room for improvement, these initial results indicate significant potential.**

***Keywords*—** Machine learning, Convolutional Neural Networks, Plant Disease Detection, Tomato Disease Detection.

I. Introduction

The accurate recognition and assessment of plant diseases play a crucial role in the agriculture industry. If not treated fast enough, diseases will spread and further contaminate surrounding plants, resulting in more agricultural losses. The best solution is proactive treatment, which requires detecting diseases before they become parasites to other plants. Over the years, extensive research has been conducted to tackle this issue by training advanced Artificial Intelligence (AI) models capable of detecting plant diseases with an accuracy greater than 95% and by creating datasets with sometimes over 20 thousand images of plant diseases. We explored the contributions of studies that have employed AI, such as in the study [2] for plant disease detection. These works have leveraged advanced techniques such as machine learning [3], and computer vision [4] to extract meaningful features from plant images and accurately determine their diseases and detect leaves.

Many of these datasets use images of leaves that have been placed in a confined space with similar backgrounds, which may not be ideal if we wish to automate the disease detection process by placing cameras in a greenhouse. With our camera station in place in a real greenhouse, we aim to provide a comprehensive performance review on tomato leaf disease detection and how existing datasets, in this case, the Plant Village dataset, perform using images in a real greenhouse environment without any prior isolation of the plant. For better results, we will train our models using two distinct models; YOLOv8 and Faster R-CNN. We also used our camera station to augment the existing dataset and gather more training data for the Plant Village dataset. Our motivation is to fully automate a greenhouse that is able to take photos and detect tomato diseases in real-time without the need for any human interaction except for the treatment of the disease.

Our contributions in this paper can be summarized as follows: (1) We augmented the Plant Village dataset by adding our own labelled images with roboflow in order to detect leaves and classify them inside a real greenhouse; (2) We wrote an algorithm capable of accurate leaves detection and cropping in order to test and use our disease detection model; (3) We introduced performance comparison with different parameters inside the model YOLOv8; (4) We compared the performance of the model YOLOv8 to the model Faster R-CNN; (5) We conducted our tests of leaf detection inside a real greenhouse using an automated photo station.

Section II gives an overview of related work. Section III explains the dataset we used and how we gathered additional data. Section IV explains the models we chose to detect diseases. Section V shows our results. Finally, Section VI concludes the paper.

II. Related Works

In this section, we discuss relevant literature that studied and contributed to the issue related to tomato disease recognition. We can divide our related work into two sections. The disease detection, the leaf detection and the tomato maturity assessment.

A. Disease detection

Most of the studies that contributed to plant disease detection, such as the tomato disease detection, used the Plant Village dataset [5], a Kaggle repository. The study [3] used it to achieve the best possible accuracy by comparing it with different types of models, such as a random forest algorithm or a neural network, achieving an accuracy of over 95%. Overall, most studies that used this dataset achieved an accuracy of 95%. Additionally, one paper [6] summarized all the methods used for disease detection inside plants by using AI or image processing. This allowed us to make comparisons between each process. In this paper, we focused on classifying tomato leaves. Thanks to this study, we could see that the fastest and easiest way to do it was using a neural network. Moreover, the paper [2] summarized models used throughout the years to detect plant diseases. We can learn that the model that got most papers on this subject was at this time AlexNet. However, two models are already implemented inside the model Faster R-CNN:

- ResNet50
- MobileNet

Since several studies also used ResNet50, we concluded that it was better to use ResNet50 than MobileNet.

The study [7] created an application called PlantifyAI used to determine, with image recognition, the disease of the tomato leaf and describe how to fix it. Similarly, in the long term, we want to create a web service accessible by farmers inside the website ALIVEcode.ca that automatically notifies them if a disease is detected with our camera station in order for them to treat it efficiently.

B. Leaf detection

In agriculture, many studies use an R-CNN or a YOLO model to detect plant leaves. In order to create our model, we used these studies as references. The study [9] used an image detection algorithm in order to compare the performance of leaf counting from a non-augmented dataset and an augmented dataset. This dataset contained strawberry leaves instead of tomato leaves, but the goal is still the same. In their research, they used an older version of YOLO, YOLOv3, the algorithm Faster R-CNN, the SSD algorithm, and finally CenterNet. Faster R-CNN achieved the highest performance in both situations and was able to improve its accuracy from 75% to almost 91%. Furthermore, the study [4] used the model MaskRCNN in order to count the number of leaves, then proceeded with the labelling using the neural network ResNet50. Most of the studies used an advanced classification model. There are two main neural networks that are used: MobileNet and ResNet50. As stated previously, since most studies used ResNet50 instead of MobileNet, we opted for ResNet50.

C. Tomato maturity assessment

Maturity assessment in fruits also uses visualization models such as in this study [8] that used the mask R-CNN model in order to detect the maturity stage of tomatoes. The model performed with an R^2 score of 0.80 on average and with difficulty to detect half-ripened tomatoes. This model performs well on tomatoes because the tomato stage is colour dependent. Contrary to the leaf detection model presented in subsection II-B, this model of R-CNN is a masked version which is required for generating an output mask of the tomato.

III. Dataset

Processing the dataset is an important part of machine learning. In this section, we will discuss the methods we used to create an exploitable dataset and the transformations we used in order to augment our samples. We can describe our process into three sections: III-A being the description of our dataset, III-B its improvement with our samples, III-C the transformation used.

A. The structure of the dataset

The existing dataset was available inside roboflow. It was composed of 32 labels. The goal of our model was primarily to detect leaves and secondary to detect sick tomato leaves. To simplify our dataset, we decided to remap labels into three categories :

- tomato leaf (Fig.1)
- sick tomato leaf (Fig.2)
- other leaf (Fig.3)

Fig. 1: Tomato leaf

Fig. 2: Sick tomato leaf

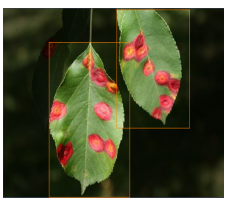

Fig. 3: Other leaf

By merging multiple labels in our dataset, we noticed class imbalance as shown in Fig.4. This could lead to an underfitting in our model. As a result, we decided to reduce the number of samples inside the label other_leaf by half as shown in Fig.5. We will present in section III-C another method we used to reduce the class imbalance.

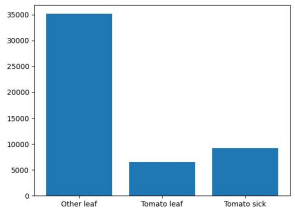

Fig. 4: Default class distribution

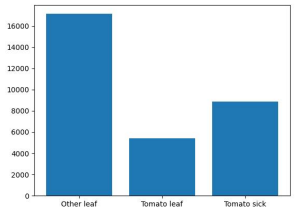

Fig. 5: Reduced class distribution

The dataset is then separated into three parts, the train set, the test set, and the validation set (See Fig.6). The train set is used to fit our model parameters, the validation set is commonly used at each epoch to check the quality performance. Finally, the train set is completely separated and is used in order to check the final performance of the model. Values used for the separation are the commonly used ones. We took 80% for the train set and approximately 10% for the two other sets as shown in Fig7.

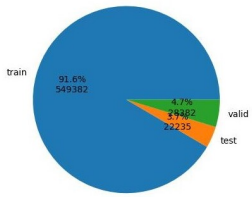

Fig. 6: Train-set-test distribution

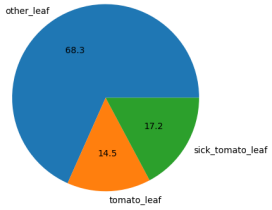

Fig. 7: Leaf distribution percentage

B. Improvement with our own dataset

Most of the studies such as [10] are done inside a controlled environment, which is not representative of an industrial greenhouse. Due to this, we improved the dataset with our own samples from the greenhouse. In this study, we only focused on samples from the early stages of the tomato plants. We took our samples with more professional cameras, such as the 64-megapixel Arducam AutoFocus, and with a more affordable one, a phone camera. The setup was taking photos automatically every 2 hours above the plantation using multiple lens focus, meaning that most of our samples are a top view of the plants with shifting resolution. The background of our images can change throughout time, between dirt and white tarpaulin. This led us to take samples with both backgrounds. We added almost 300 images to our dataset and still counting, such as in Fig.9.

Fig. 8: Example of leaves form greenhouse

We used roboflow to label our samples, such as in Fig.8.

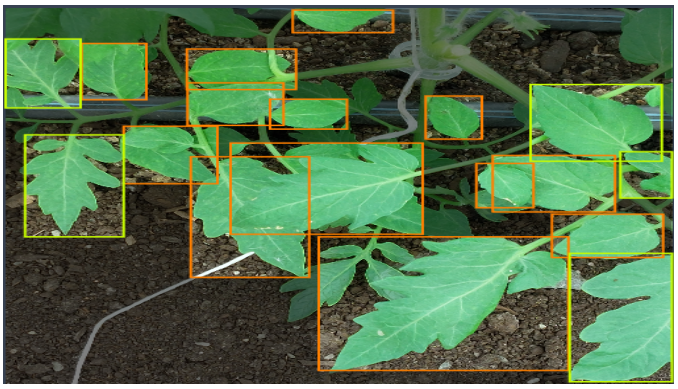

Fig. 9: Labelling of our samples

C. Image transformations

In this section, we will talk about the transformations we've done in our samples. This method was used in order to perform data augmentation, preventing our model to overfit. According to the study[11], we can improve the accuracy of the model Faster R-CNN by almost 10% by augmenting our data. We decided to take the distribution between each label and randomly transform them according to it. To train our models, we used roboflow available transformations for YOLO's models, and we performed torch transformations too.

1) Class imbalance reduction: According to [12], the ratio between labels strongly impacts the ratio of false positive and true negative predictions. The impact is significant at a ratio of 1:8. As we can see in Fig.7, the ratio between our samples is roughly the same between tomato leaf and our sick tomato leaf. However, we have a ratio of almost 1:4 between these labels and the label other leaf.

Note that this part will only concern the model Faster R-CNN and their variations. Transformations were performed thanks to the second version transform inside torchvision. This method is still in beta stage in their library. Therefore, we personalized the wrapper in order to make transformations. In order to reduce the effect of the class imbalance, we decided to prefer to transform our data according to the opposite of the probability of distribution of the class. For each sample, the probability was calculated with the equation:

$chance\_of\_transformation = \frac{1}{label\_distribution}$

2) Transformations applied: Here is a list of each possible transformation applied. Results are shown on Fig.10:

- Crop: 0% minimum zoom and 24% maximum zoom
- Rotation: Between -30° and +30°
- Shear: ± 27° Horizontal, ± 29° Vertical
- Bounding box shear: ±15° Horizontal, ±15° Vertical
- Brightness: -36% and +0%
- Blur: Up to 1.25px
- Mosaic

Crop, rotation shear, and bounding box shear are applied strictly in order to augment our data. We changed the brightness to simulate the change in luminosity inside the greenhouse, and the blur to simulate the switch of focus and resolution of our camera. Finally, the mosaic will make our model more adaptable to new images. The mosaic effect has only been applied to the model YOLOv8.

Fig. 10: Example of transformations applied

IV. Models

Image detection and classification models are commonly used, and many models have come out during these years. In order to detect diseases in our plants, we decided to take two different types of models. YOLO's model and Faster R-CNN's model. In this section, we will first introduce these two models. Then we will explain their architecture and finally how we fine-tuned our models.

A. YOLOv8 and Faster R-CNN introduction

1) Faster R-CNN model: Faster R-CNN (Region-based Convolutional Neural Network) is an influential object detection algorithm introduced in 2014. It utilizes a two-stage approach, generating region proposals and then classifying and refining them using a CNN. Faster R-CNN has been foundational in the development of faster and more accurate object detection algorithms. It has found applications in image analysis, surveillance, and autonomous driving. Models are available on Torch, we used the series Faster R-CNN. it's a faster version of the Faster R-CNN. Models available on torch are:

- fasterrcnn_resnet50_fpn from the study [13]
- Faster R-CNN_resnet50_fpn_v2 from the study [14]
- fasterrcnn_mobilenet_v3_large_fpn
- fasterrcnn_mobilenet_v3_large_320_fp

We decided to use version 2 of the model with ResNet50 in order to detect our disease, since it was more used than mobilenet.

2) YOLOv8: YOLO has wide applications in fields like autonomous driving, surveillance, and robotics. Its real-time performance and high accuracy make it a preferred choice for object detection tasks. YOLO has revolutionized computer vision with its speed and efficiency in detecting objects. YOLOv8 is the last model in the YOLO's series. It has been released by the library ultralytics. YOLO's models have become the leader of image detection models thanks to their performance on the COCO dataset. COCO dataset is a database of over 330k images and 81 categories. It is commonly used to check a model's performance. Fig.11 shows the progression of each model of YOLO's series.

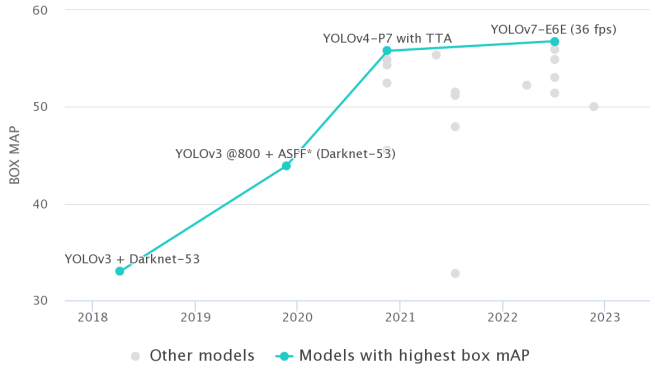

Fig. 11: Example of transformations applied

B. The model's architecture

1) Faster R-CNN's model: In this article, we decided to use faserRCNN_v2 and Faster R-CNN_v3, available on Torch. The version 2 of the model has a ResNet 50-FPN backbone [15] and version 3 has a MobileNetV3-Large FPN backbone [16]. Fig.12 shows the global architecture of Faster R-CNN. VGG layers are the architecture of convolutional layers used for high-resolution image transformation. It is the backbone. The region of interest uses a spatial pyramid layer [17]. Finally, there are two final neural networks of size 1024 and one layer of size 2 to detect whether the tomato leaf is sick or not.

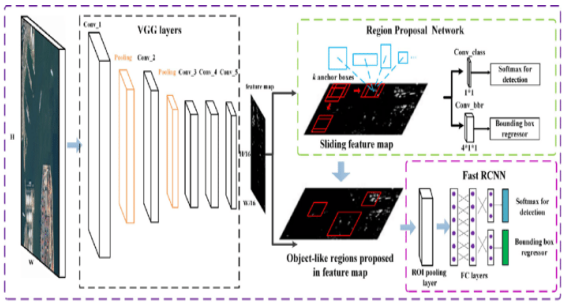

Fig. 12: Architecture of Faster R-CNN

2) YOLOv8: The article [18] explains the construction of each version of YOLO. To train our data, we used the last version of YOLOv8 available on torch. Its architecture provided by [19] is almost the same as YOLOv5. Its CSP Layer has been replaced by a C2f module in order to improve detection accuracy. The 6×6 convolutional layers backbone was changed into a 3×3 convolutional layer. We used the large scale of this model.

V. Results

In this section, we will be using 4 types of metrics.

- The box loss, the loss of the predicted boxes.
- The recall, the fraction between the predicted relevant instance between all the relevant instances also known as sensitivity.
- The precision, the fraction of relevant instances among the retrieved instances.
- The mean average precision 50 (mAP50), the average precision of object detection with an intersection of union above 50

In order to reduce the training time, we dropped the category other_leaf. We decided to test 3 different types of datasets:

- Images from the greenhouse called natural.
- Images from the greenhouse augmented called leaves_v1.
- Images from leaves and greenhouse augmented mixed.

A. Faster R-CNN

In this section, we will talk about the results of the model Faster R-CNN.

We decided to train our models, for 25 epochs. The model seems to perform its best on the dataset mixed. At the end of the training phase, we achieve:

- 0.20 of box loss on natural on Fig.13 and 50% of mAP50 on Fig.14.
- 0.22 of box loss on leaves_v1 on Fig.15 and 60% of mAP50 on Fig.16.
- 0.30 of box loss on mixed on Fig.17and 60% of mAP50 on Fig.18.

.

These results are expected because the natural version and leaves_v1 have fewer samples and fewer diversity than the mixed one. Moreover, we can see that the mAP50 of mixed is higher than the others.

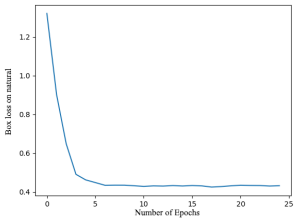

Fig. 13: Box loss of yolov8 on natural

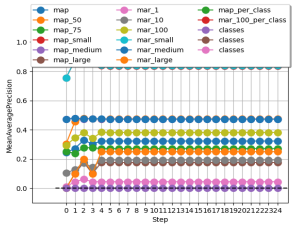

Fig. 14: Box loss of yolov8 on natural

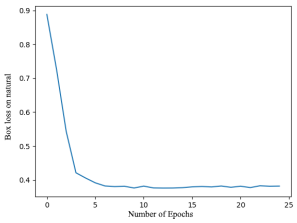

Fig. 15: Box loss of Faster R-CNN on natural

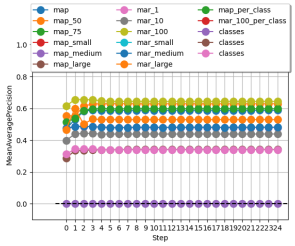

Fig. 16: Box loss of Faster R-CNN on leaves_v1

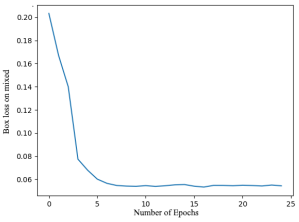

Fig. 17: Box loss of Faster R-CNN on mixed

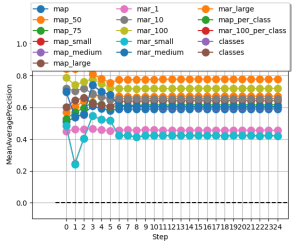

Fig. 18: Box loss of Faster R-CNN on mixed

B. YOLOv8

In this section, we will talk about the results using the model yolov8.

The Yolov8 model seems to achieve a high box loss according to Fig.19. However, it performs well at mapping, achieving a mAP50 of 60% as shown in Fig.20. Moreover, we can see that the model has a bad precision and recall as shown in Fig.21. This can be explained by the lacking quantity of sick leaves inside our dataset.

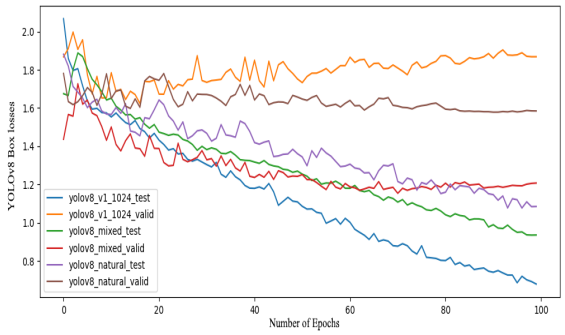

Fig. 19: train and test box loss of yolov8 models

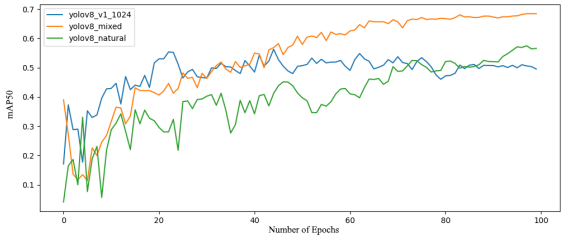

Fig. 20: mAP50 of yolov8

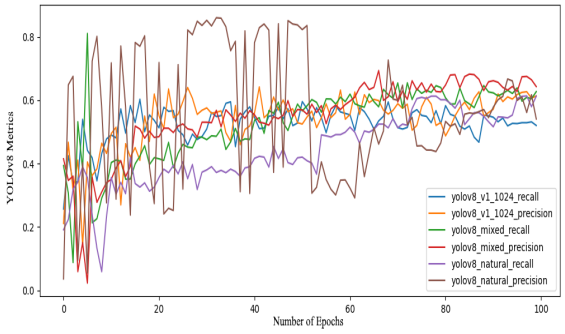

Fig. 21: Recall and precision of yolov8

C. Comparison

On Fig.22, we can see the result on mAP50 on each label. We can see that the model yolov8 on the natural dataset achieves the best performance among our models. We are aware of the lack of data on our validation set (only 4 images) on the natural dataset. Therefore, we should gather more data in order to evaluate our models. In Fig.23, we have the prediction from the model of the model yolov8 on the natural dataset.

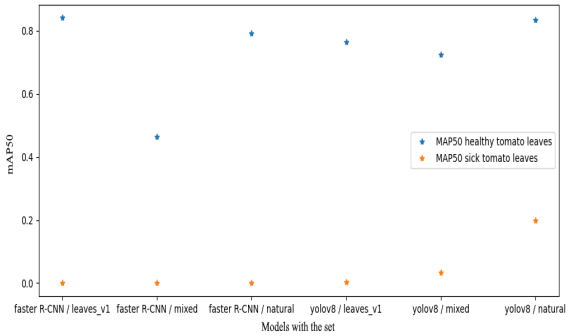

Fig. 22: mAP50 of labels on each model

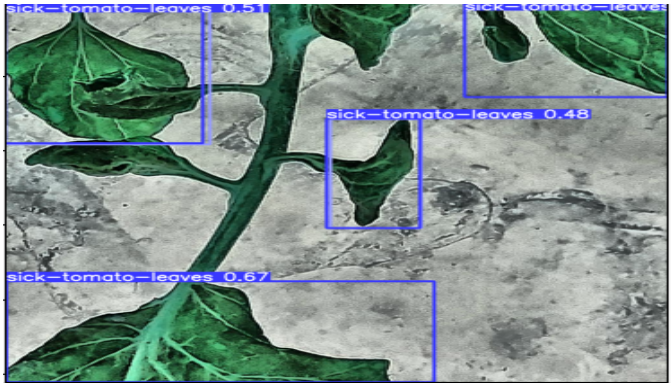

Fig. 23: Caption

VI. Conclusion and Future Works

We've seen that the detection of the leaves works well. By having a map overage of 60 % we detect most of the plants. However, we need to improve the results of the classification. Detecting leaves that are sick from the view of our camera is difficult due to the number of noise that can occur and the distance between plants. Moreover, we can't detect every plant. Further data on the greenhouse will be collected and labelled in order to increase the model's accuracy. Moreover, we will increase the input size of our models and the quality of our samples in order for the model to detect disease easily.

In future work, we want to link the greenhouse parameters such as temperature or humidity with the disease apparition. The final goal is to totally prevent the apparition of disease thanks to the data we collect there.

Acknowledgements

We would like to thank La CCHM and Innovations ALIVEcode inc. for supporting this research. We would also like to thank Nils Lahaye, Simon Lafrance, Adrien Lesage, Merieme Bouisri, Francis M. Gosselin, Abderrazak Mokraoui, and other members of our research lab who aided us in our greenhouse project.

References

[1] The industrial greenhouse project: https://alivecode.ca/showcase/projects/SICRO [last visited 14 July 2023]

[2] Saleem, M.H.; Potgieter, J.; Arif, K.M. Plant Disease Detection and Classification by Deep Learning. Plants 2019, 8, 468. https://doi.org/10.3390/plants8110468

[3] A. Lakshmanarao, M. R. Babu and T. S. R. Kiran, "Plant Disease Prediction and classification using Deep Learning ConvNets," 2021 International Conference on Artificial Intelligence and Machine Vision (AIMV), Gandhinagar, India, 2021, pp. 1-6, doi: 10.1109/AIMV53313.2021.9670918

[4] L. Xu, Y. Li, Y. Sun, L. Song and S. Jin, "Leaf Instance Segmentation and Counting Based on Deep Object Detection and Segmentation Networks," 2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS), Toyama, Japan, 2018, pp. 180-185, doi: 10.1109/SCIS-ISIS.2018.00038.

[5] https://www.kaggle.com/datasets/abdallahalidev/plantvillage-dataset

[6] Liu, J., Wang, X. Plant diseases and pests detection based on deep learning: a review. Plant Methods 17, 22 (2021). https://doi.org/10.1186/s13007-021-00722-9

[7] S. Shrimali, "PlantifyAI: A Novel Convolutional Neural Network Based Mobile Application for Efficient Crop Disease Detection and Treatment," 2021 2nd Asia Conference on Computers and Communications (ACCC), Singapore, 2021, pp. 6-9, doi: 10.1109/ACCC54619.2021.00008.

[8] Umme Fawzia Rahim* and Hiroshi Mineno. 2021. Highly Accurate Tomato Maturity Recognition: Combining Deep Instance Segmentation, Data Synthesis and Color Analysis. In 2021 4th Artificial Intelligence and Cloud Computing Conference (AICCC '21), December 17-19, 2021, Kyoto, Japan. ACM, New York, NY, USA, 11 Pages. https://doi.org/10.1145/3508259.3508262

[9] Rahim UF, Mineno H (2020). Data augmentation method for strawberry flower detection in non-structured environment using convolutional object detection networks. J. Agric. Crop Res. 8(11):260-271. doi: 10.33495/jacr_v8i11.20.180.

[10] M. A. M. Araneta et al., "Controlled Environment for Spinach Cultured Plant with Health Analysis using Machine Learning," 2021 IEEE 13th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM), Manila, Philippines, 2021, pp. 1-6, doi: 10.1109/HNICEM54116.2021.9732020.

[11] R. A. Harianto, Y. M. Pranoto and T. P. Gunawan, "Data Augmentation and Faster Improve Vehicle Detection and Recognition," 2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT), Surabaya, Indonesia, 2021, pp. 128-133, doi: 10.1109/EIConCIT50028.2021.9431863.

[12] A. Ahmadzadeh and R. A. Angryk, "Measuring Class-Imbalance Sensitivity of Deterministic Performance Evaluation Metrics," 2022 IEEE International Conference on Image Processing (ICIP), Bordeaux, France, 2022, pp. 51-55, doi: 10.1109/ICIP46576.2022.9897445.

[13] Li, Yanghao et al. "Benchmarking Detection Transfer Learning with Vision Transformers." ArXiv abs/2111.11429 (2021): n. pag.

[14] Li, Yanghao et al. "Benchmarking Detection Transfer Learning with Vision Transformers." ArXiv abs/2111.11429 (2021): n. pag.

[15] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[16] A. Howard et al., "Searching for MobileNetV3," 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), 2019, pp. 1314-1324, doi: 10.1109/ICCV.2019.00140.

[17] He, Kaiming, et al. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition." Lecture Notes in Computer Science, 2014, pp. 346–61. Crossref, https://doi.org/10.1007/978-3-319-10578-9_23.

[18] https://arxiv.org/abs/2304.00501

[19] https://github.com/ultralytics/ultralytics/issues/189, 2023