

Research Paper

Citrus Disease & Pest Detection Using Yolo Models with Self - Attention Yolov8

Ajmeera Veeranna

Master's Student, Department of electronics and communication engineering

jawaharlal nehru technological university hyderabad university
college of engineering jagtial, telangana, 505 501Email: veerannaajmeera18@gmail.com

Saddala praveen kumar

Associate professor,

Department of electronics and communication Engineering
jawaharlal nehru technological university hyderabad university
college of engineering jagtial, telangana, 505 501Email: praveenkumar_00019@yahoo.com

Abstract— Many of the diseases of citrus are equally devastating and early detection is essential for timely orchard management. In this work, a citrus disease detection and guiding pipeline, CitraYOLO, is presented, which covers comparative object-detection studies, the introduction of CitraYOLO's unique lightweight self-attention YOLOv8 network, and a Flask web application. The experimental part showcases Faster R-CNN and seven YOLO family variants: YOLOv3, YOLOv5s6, YOLOv5x6, YOLOv8n, YOLOv11, YOLOv26 and a special Light SA-YOLOv8 model. The proposed detector incorporates Fast_C2f, BRA and ASF straight into the YOLO pipeline. Structured citrus dataset, consisting of 4602 training images and 657 validation images across five classes: Melanose, black spot, canker, greening and healthy, were used for training. Models are compared to each other by precision, recall, mAP@0.5, F1-confidence curve, precision-recall and confusion matrix. According to the comparison results, all the metrics of the top three models (YOLOv5x6, YOLOv11, YOLOv3) are complementary. Based on the comparison results, YOLOv5x6 has the highest precision (0.774), YOLOv11 has the highest recall (0.895) and YOLOv3 has the highest mAP (0.829). A modified version of SA-YOLOv8, which is a lightweight efficient model with a balanced architecture, outperforms YOLOv8 with 0.756 precision, 0.782 recall and 0.800 mAP. The chosen model is incorporated with Flask for real world deployment and SQLite based user authentication, image upload, confidence display and local disease description, cause, remedy and prevention counsel.

Keywords— Citrus disease detection, self-attention, YOLOv8, object detection, deep learning, precision agriculture, Flask, disease guidance."

I. INTRODUCTION

Late disease diagnosis can cause several orchard diseases of citrus to reduce yield, quality and productivity. Under field conditions symptoms such as black lesions, corky spots, mottled yellowing and surface imperfections are often difficult to separate as their visual appearance is dependent upon illumination, scale, orientation, disease stage, background clutter and occlusion. Manual scouting is still important but is time consuming, subjective and difficult to scale up to large orchards. This led to the development of automated illness detection systems from images that can

provide quick, preliminary screening and aid in more consistent monitoring.

In the field of plant-health image analysis, deep learning has removed the need for handcrafted colour, texture and shape descriptors and has introduced learned hierarchical representations. Two-stage detectors like Faster R-CNN allow for the accurate region proposal and localisation. The one-stage YOLO detectors combine localisation and classification tasks into a single inference pipeline, and are suitable for real-time applications in agriculture [13]–[17]. The detection of tiny lesions, contextual attention, multi-scale feature fusion, and lightweight deployment have been paid more and more attention in recent citrus research [1]–[8].

The aim here is to build CitraYOLO as a similar and operational framework, but not just a notebook-only model. The data set procedure is used for 8 detector configurations that are examined. Moreover, a customised Light SA-YOLOv8 architecture is proposed, which consists of Fast_C2f for fast feature extraction, BRA for selective global-context interaction, and ASF for adaptive multi-scale spatial feature fusion. The work also correlates the experimental evaluation with the application developed using Flask, that enables signup, signin, upload of images, prediction of disease, reporting the confidence, and retrieval of disease description, cause, cures and prevention measures by following rules.

The main contribution is thus tri-fold. The first one is the large-scale cross model comparison of the two-stage detector and multiple versions of YOLO. Second, it demonstrates the integration of attention and fusion modules in the source level with a lightweight detector based on YOLOv8. Third, it provides an end-to-end user workflow, bringing model output to life by presenting meaningful information and assessment graphs through the application interface, making this information readily available for disease management.

II. RELATED WORK

Citrus vision studies have progressed over the past few years from heavy duty classification to light weight detection systems for the orchard environment. However, to overcome

this occlusion, Lin et al. [1] proposed AG-YOLO which integrates the local feature analysis and contextual information for fast citrus fruit detection. Their findings reveal the benefits of using context fusion when local cues are occluded by overlapping targets and dense foliage. To adaptively fuse and optimise the location in the small-target feature extraction, Wu et al. [2] proposed the SAW-YOLO which is oriented to citrus pest detection in multi-scale. Although the study examined pest, it is relevant to minor disease lesions, as there are scale and background issues.

As for the detection of Huanglongbing, Xie et al. [3] tried to solve the problem using a YOLO-based approach, marking a transition from laboratory-based categorisation to field-based illness monitoring. In order to detect citrus leaf diseases, Zheng et al. [4] enhanced the citrus detection model YOLOv8 by emphasizing on slight symptom and feature representation. Feng et al. [5] suggested a lightweight detector YOLO-Citrus based on YOLOv11 for the complicated background of agriculture. For the detection of citrus pest and diseases, Dai et al. [6] developed a YOLOv11-RDTNet that carefully balances the feature representation and efficiency.

A number of 2025 studies went on to look more at building efficiency. Guo et al. [7] proposed a dilated reparameterized feature enhancement and shared-parameter head for citrus disease diagnosis. Xue and Wang [8] proposed PEW-YOLO, a compact backbone network, attention mechanism with efficiency and localisation optimisation. Goyal and Lakhwani [9] focused on deep feature learning and augmentation for recognition of the diseases in citrus leaf and fruit. Dinesh and Lakshmanan [10] used multiclass semantic segmentation to localise the pixels and analyse them by their severity. Multi-kernel convolution and channel-spatial attention [11] have also been investigated for attention-based classification, and multi-disease diagnosis models built with YOLOv8 are still proving the importance of the task's quality and fusion of features [12].

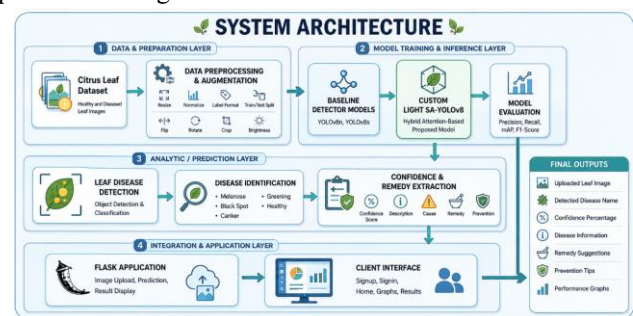
New architectures are accompanied with the earlier field/deployment studies. Yolov5s-BC was used for citrus psyllid detection by Lyu et al. [22] and a YOLO convolutional network for automatic identification of Huanglongbing by Qiu et al. [21] and Dhiman et al. [23] fused deep learning with edge computing for citrus-fruit disease diagnosis. Recent works for disease detection and mapping in orchards have demonstrated the benefits of using mobile edge AI systems as presented by Da Silva et al. [24]. More general reviews of deep learning for plant health monitoring [25]–[27] highlight the critical issues of data quality, domain shift, interpretability and efficient deployment for such systems, which are further refined by recent orchard-specific detectors like YOLOv8-Scm [28] and YOLO-MECD [29] that focus on lightweight, context-aware recognition in complex natural environments.

There are two shortcomings identified in the literature that are of practical concern. Secondly, very few studies come with a lot of cross-generation comparisons that help to understand the metric trade-offs, given that most studies only

deal with one modified detector. Secondly, the experimental effort ends when the model is assessed and does not combine in one deployable workflow user identification, image upload, confidence display, disease explanations, remedy recommendations, prevention information, and diagnostic graphs. CitraYOLO addresses these drawbacks by incorporating comparative evaluation, an exclusive hybrid attention detector, and a robust application layer.

III. MATERIALS AND METHODS

The proposed approach is iterative, in the sense that dataset preparation, common preprocessing, training baseline detector, designing the attention-model to the user's needs, model evaluation, selection of the weights and web deployment take place step by step. The experiments in YOLO are defined in the same way as YOLO-family experiments. A custom DataSet Wrapper converts YOLO-format annotations to Faster R-CNN. The overall method is presented in Fig. 1.



“Fig. 1. Overall CitraYOLO system architecture.”

A) Dataset Collection

The dataset is structured in the YOLO detection format and has separate folders for training and validation images and labels. The training subset has 4,602 photos, with eight images of background. The reported scan contains 657 images, and no corrupt files in the validation subset. There are five classifications: Melanose, black spot, canker, greening and healthy. Unbalanced distribution of validation Melanose 1 black spot 85 canker 380 greening 81 healthy 110. This mismatch is considered in the interpretation of class-level diagnostics and is one of the factors that leads to expanding classes that are under-represented.

“Table 1. Dataset organization and validation distribution.”

Subset / Class	Images	Instances	Purpose / Note
Training	4,602	—	Model learning
Validation	657	657	Common evaluation
Class distribution	—	657	Melanose 1; Black spot 85; Canker 380; Greening 81; Healthy 110

B) Pre-Processing and Augmentation

Ultralytics training validates the image-label mappings, generates or re-uses caches, resizes and letterboxes the images to a predetermined input size and transforms the samples into tensors for training on GPUs. Most of the runs from the YOLO-family use 448×448 . In Faster R-CNN, images are loaded via Pillow, converted to RGB, turned to

tensors and normalised YOLO center coordinates are translated to absolute corner coordinates required by the TorchVision detector. The class order is preserved, as the projected model class name is utilised as the key for disease-information search in the web application later.

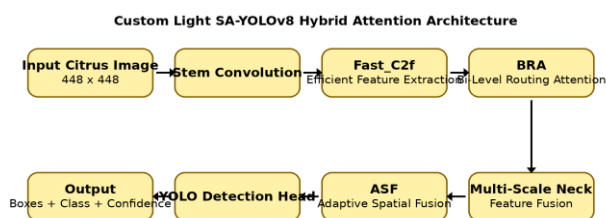
YOLO trainer is able to process geometric and photometric augmentation. The recorded configurations employ mosaic augmentation with a close_mosaic parameter, disabling mosaic during the last epochs, to enable the detector to learn from composite scenes early and then adjust to natural picture distributions near convergence. Depending on the trainer configuration, scale, translation, flipping and color-space variation can also be implemented. Augmentation is used sparingly to ensure that the disease boxes are valid, and that the appearance of small lesions is not lost.

C) Baseline Detector Training

Faster RCNN is using ResNet-50 FPN backbone with a new ROI predictor which predicts 6 outputs: background and 5 citrus labels. The experiment is repeated with the Adam optimiser and learning rate 0.0001 for 10 epochs. The studies of the YOLO-family include YOLOv3 Tiny, YOLOv5s6, YOLOv5x6, YOLOv8n, YOLOv11 and YOLOv26 and typically run 50 epochs on input images of 448-pixel resolution. The comparison is done based on multiple model scales and detector generations to uncover precision-recall trade-offs and not just because of what came off as being the most accurate out of accuracy intuition.

D) Custom Light SA-YOLOv8

The custom detector gets added to an adapted ultralytics source tree. Three Python layer modules: Fast_C2f, BRA, and ASF are registered in the package initialisation and model-construction code, allowing for a dedicated YAML architecture file to resolve them during network parsing. According to the project log, the end network has 139 layers, approximately 2.91 million parameters and approximately 7.4 GFLOPs. Fast_C2f allows for feature extraction with efficiency, BRA allows for multi-level routing attention to selectively engage with context, and ASF allows for multi-scale spatial data integration ahead of the YOLO detection head.



“Fig. 2. Custom Light SA-YOLOv8 hybrid attention architecture.”

E) Training, Evaluation and Web Integration

There is a common results structure that contains precision, recall and mAP for all 8 detectors. The mAP@0.5 for Faster R-CNN is calculated with TorchMetrics, and the true positive, false positive and false negative for Precision and Recall are calculated by a matching technique based on

IoU. YOLO metrics are taken from the last validation outputs. F1-confidence curves, precision-recall curves and confusion matrices are maintained for analysis of confidence thresholds and class-level confusion in addition to a summary score.

The Flask application is built and calls in best.pt on application start up and provides routes for landing, signup, signin, prediction and the visualisation of the model. Missing or empty choices in uploaded files are checked, uploaded files are saved to a static upload directory and then sent to the detector. If the detection is not output, the interface reports "No Disease Detected" with 0% confidence. If not, the class name and confidence are retrieved and the matching local knowledge item gives a description, likely cause, recommendations for cure and advice for prevention. SQLite is used for account registration and signin while disease information is left in separate structure in python for modular maintenance.

IV. EXPERIMENTAL RESULTS

We measure the models with precision, recall and mean average precision on IoU 0.5. Precision is the fraction of true positive detections out of all positive detections made and recall is the fraction of relevant items that are detected. The F1 score considers precision and recall and the mAP is a summary of the precision-recall curve over the IoU threshold selected and over the detection confidence. The metrics are stated as:

$$\text{Precision} = TP / (TP + FP)$$

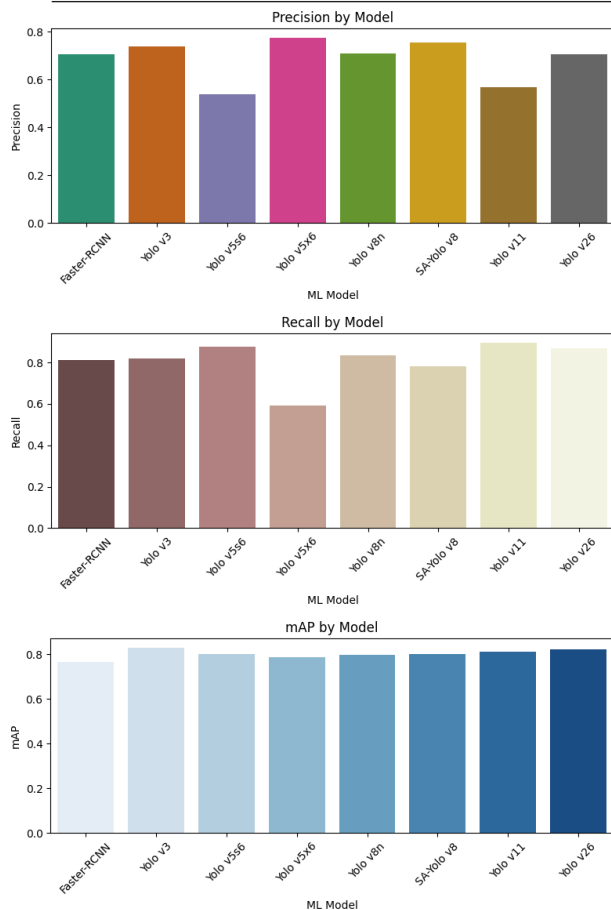
$$\text{Recall} = TP / (TP + FN)$$

$$F1 = 2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$$

“Table 2. Performance evaluation of detection models.”

Model	Precision	Recall	mAP
Faster R-CNN	0.707	0.813	0.765
YOLOv3	0.737	0.819	0.829
YOLOv5s6	0.540	0.875	0.801
YOLOv5x6	0.774	0.593	0.788
YOLOv8n	0.709	0.835	0.797
SA-YOLOv8	0.756	0.782	0.800
YOLOv11	0.567	0.895	0.811
YOLOv26	0.704	0.869	0.822

It can be seen from Table 2 that there is no detector which is best in all the criteria. The precision of YOLOv5x6 is the highest (0.774), which means that the positive detections are more conservative; however, the recall is low (0.593). The recall of YOLOv11 is the highest (0.895) which indicates the high sensitivity of detecting disease cases, and the mAP of YOLOv3 is the highest (0.829). YOLOv26 is very close with 0.822 mAP and 0.869 recall. However, the performance of the bespoke SA-YOLOv8 is overall balanced with a precision 0.756, recall 0.782, and mAP 0.800, but it is not the best. This discovery suggests that other methods can be used for choosing a detector to be deployed.



“Fig. 3. Precision, recall, and mAP comparison across evaluated detectors.”

The diagnostic curves provide an alternative way of interpreting the suggested model. The F1-confidence curve identifies a trade-off in terms of missed detections and false alarms. Precision-Recall curve represents the tradeoff between the sensitivity and the decision thresholds, and the confusion matrix indicates which classes are the most confused. This distribution of the validation is strongly skewed towards canker, with only one event of Melanose. The overall scores should therefore be interpreted in the context of the class level evidences. Evaluation should be done on a larger and more balanced geographically distributed test set in the future.

A) Metric-Wise Interpretation

The conservative behaviour of YOLOv5x6 is highlighted by the precision comparison as it has 0.774 precision that decreases sensitivity. The unique SA-YOLOv8 is close to the precision of 0.756 and significantly bigger in the recall rate. The accuracy of faster R-CNN and YOLOv8n is similar, while YOLOv5s6 and YOLOv11 are on the lower side due to compromising on recall. It is operationally crucial that the distinction is made for both orchard screening and a high precision detector will minimize false positive alerts and a high recall detector is desired if the cost of missing an infected sample is high.

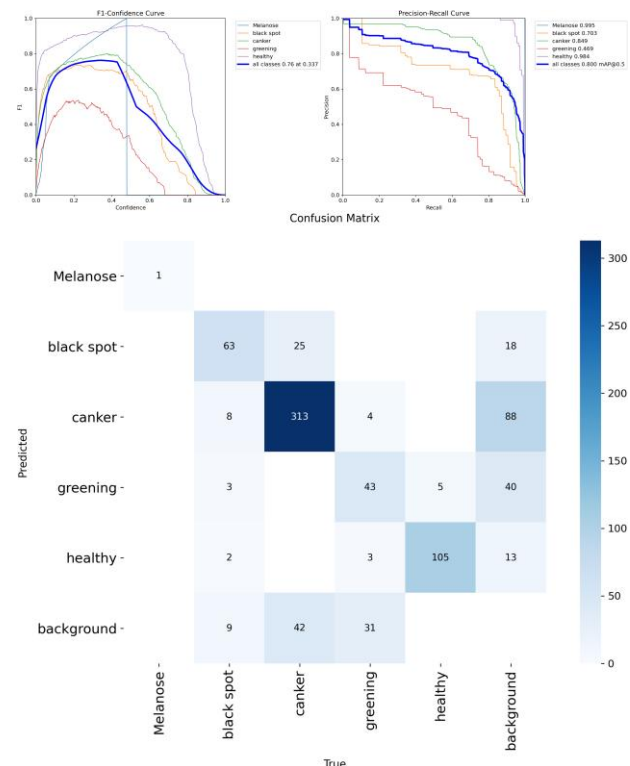
Based on the recall results, the YOLOv11 model achieves a score of 0.895, which is higher than YOLOv5s6 (0.875) and YOLOv26 (0.869). These models detect more positive examples, but have varying false positive tendencies with the related precision values. The balanced precision-recall pair of

SA-YOLOv8 is 0.756 and 0.782. This balance is important when the program must deliver stable forecasts, but not be too conservative, nor too sensitive.

For the mAP comparison, the overall detection quality is demonstrated at the given IoU threshold. The highest accuracy of YOLOv3 is 0.829 mAP, followed by YOLOv26 with 0.822 and YOLOv11 with 0.811. The performance of SA-YOLOv8 is the same as YOLOv5s6 and YOLOv8n, which is 0.800. The results indicate that the assessment of architectural innovation should not be based purely on the best overall score. Therefore, the custom model should be considered as a lightweight hybrid attention model, and future studies may be confirmed by control ablation, latency, memory profiling and testing with a wider independent test set.

B) Diagnostic Behaviour of SA-YOLOv8

The F1-confidence, precision-recall and confusion-matrix diagnostic plots are shown in Figure 4 for the examination of SA-YOLOv8. The F1-confidence curve is important to make sure that the deployment threshold is not too high or too low. The precision-recall curve is a threshold-independent representation of precision-recall trade off. The confusion matrix completes the overall scores and it also highlights the errors for each class and the effect of the imbalanced distribution of data across the validation set.



“Fig. 4. SA-YOLOv8 diagnostic curves: F1-confidence, precision-recall, and confusion matrix.”

These plots should be considered in conjunction with the class distribution. There are 657 examples in the validation set. Canker has 380 occurrences, melanose has just one. So, a high aggregate score doesn't necessarily reflect a high reliability of all the categories of illness. So the present results are presented as comparative project data, and should not be considered as universal field level diagnostic accuracy. A

stronger validation approach would include stratification of data by orchard, season, device, symptom stage, and acquisition settings as well as by a “held out” test set never used during model selection.

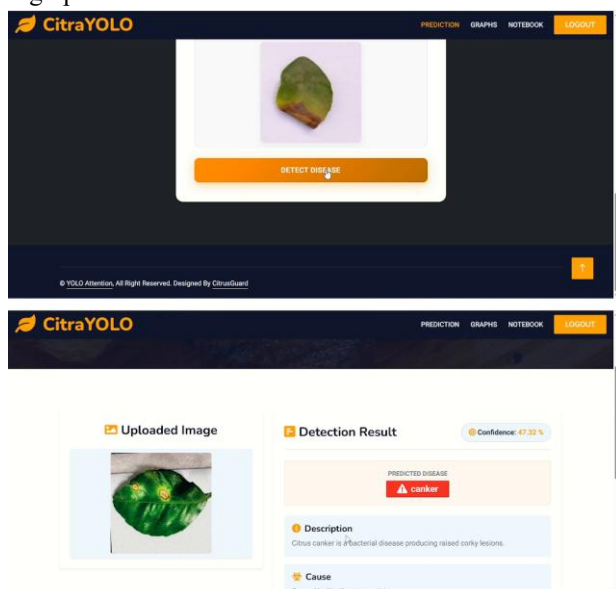
C) Comparative Discussion

The full detector comparison indicates that the type of detector to choose will depend on the application in which they are deployed. If an alert is needed to recall, YOLOv11 or YOLOv26 may be a better choice, but in case of conservative alert, YOLOv5x6 may be better. YOLOv3 is found to have the maximum mAP in the above mentioned comparison. This shows that it is not always possible to say a newer model version is superior in performance on one particular set of data. The customized SA-YOLOv8 has a mini architecture for more attention-focused research and is placed in the balance zone.

The hybrid design also naturally allows ablation research. Fast_C2f can be evaluated on its own to evaluate efficient feature extraction, while BRA can be evaluated on its own and ASF can be evaluated on its own and in combination to measure the value of routing contextual attention and adaptive feature fusion, respectively. The same splits of the data set, image size, optimisations parameters, random seeds and evaluation procedure are used for all configurations. These controlled studies would clarify the contribution of each module without allowing the effects of training conditions to mask the contribution of each module.

D) Application Results

The interface deployed displays the whole operational sequence. Once the authenticated user registers and signs in they are taken to the prediction page, where they choose a citrus image, preview it, and request it to be analyzed. The outcome screen displays the image that was submitted, the predicted category, confidence, disease description, cause, cure/prevention details. The system has been trained for healthy class as well as disease such as black spot and canker. As experiments run, the confusion matrix, precision-recall curve, F1-confidence curve and comparison charts appear on the graphics dashboard.



“Fig. 5. Web deployment workflow: image upload interface and canker detection result.”

The utility of the deployment is to convert the raw signal from the detector into a meaningful answer for a decision support. The model inference, web routing, SQLite authentication and illness knowledge retrieval are separated so that further weights (classes), guidance entries or visualisation pages can be added on without rebuilding the entire program. In the meantime, the present application is not a laboratory confirmation or therapy prescription service, but an image-based instructional and decision support service.

The need for class stability is also indicated in the result workflow. The local knowledge structure is directly accessed to get disease information using detector class name. If the detection is correct, but there is a difference in spelling, capitalisation or class ordering between the training model and the guidance dictionary, then the missing information will be produced. This is why model settings, dataset labels and application knowledge keys are kept the same for the training and deployment phases.

The valid and duplicate registration, valid and invalid sign-in, missing-file upload, empty file names, disease detection, handling of healthy-class and graph-page navigation are included in functional tests. The model is loaded only once at the start of the Flask application, and is not being loaded repeatedly per request. In reality, it would be better to add support for password hashing, secure session, CSRF protection, MIME/extension validation, file-size limits, HTTPS, logging and a real WSGI server for local demo.

E) Functional Testing and Reliability Analysis

The application was seen as a series of interrelated functional parts, not just a model demonstration. Registration testing checks for a valid form, duplicate username, regular expression for user name, email, cellphone number, and password entries. Signin testing guarantees the success of a navigation with saved credentials and gives appropriate error message for bad credentials. The checks are required as a correct detector is not practically deployable if basic access control and interaction with the users are not in order.

Request validation is the first step in prediction testing. The route determines if an image “field” exists in the request, and if a file name was even selected. Upon successful upload, it is saved in the specified static folder and then passed to the uploaded YOLO model. Take special care of the no-detection path by returning a zero-confidence result and do not perform a disease knowledge lookup. This prevents issues if the program were to attempt to get class information where there is no bounding box.

The positive-detection path uses the predicted class identifier and confidence in the returned model result, resolves the readable class name from the model class map and gets the knowledge entry for the class. It is beneficial to distinguish between the detection and knowledge retrieval processes, allowing for changing the agronomic descriptions without retraining. It also allows for easier maintenance of the

system. Each of these – model weights, Flask routes, HTML templates, static assets, SQLite operations and illness guidance – are distinct pieces of the system, tightly coupled by clear inputs and outputs.

“Table 3. Representative functional test scenarios.”

Test ID	Scenario	Expected Outcome
TC-01	Valid registration	Account stored
TC-02	Duplicate username	Registration rejected
TC-03	Valid signin	Dashboard displayed
TC-04	Missing image	Upload warning
TC-05	Disease image	Class, confidence, guidance
TC-06	No detection	Zero-confidence result
TC-07	Graphs page	Evaluation plots displayed

Table 3 presents some end-to-end testing results. These tests are designed to give the user a level of confidence that the user-visible workflow is sound in normal and error conditions and do not replace formal software testing. In particular, the no-detection branch and the missing-file checks are important for image apps, since an empty response from the inference API is a valid result, and a browser form could be submitted without a valid file selection. The scenarios will be explicitly addressed, making the system more reliable and avoiding confusion of server failures.

F) Practical Limitations and Threats to Validity

There are several restrictions to be borne in mind in interpreting experimental data. The most important is the lack of balance in the validating. The overall accuracy, recall and mAP do not uniformly imply the class-level dependability. There is only one instance of Melanose and 380 instances of Canker. A larger independent test set with balanced size of orchards, seasonal variation, several camera devices, various backgrounds and different stages of symptoms development should be collected in a future study. Having independent evidence, such as validation of results at multiple locations, would give further evidence that the detector has learned the characteristics of the disease, rather than the inputs that were specific to acquisition.

The second constraint is the direct comparison between different families of detector. Final metrics for different architectures (under the project training setup) are logged into the notebook. Nevertheless, some of these initialisation, optimisation schedule, epochs, image resolution, augmentation policy, early stopping, random seeds and hardware measurement protocol should be fixed for a publication-quality benchmark. Reporting repeated run results as mean and standard deviation. These accuracy measurements should be used in conjunction with latency, throughput, parameter count, GPU memory and energy consumption to make deployment decisions.

The result returned in the present Flask prediction route will be the one returned by the box returned first. This is valid for the single result method herein specified, but for multi-lesion or multi-symptom photos, the iteration over all boxes, the non-duplicate aggregation and possibly a class-wise confidence summary are required. Future versions should include the regions found in an annotated image and a well-organized list of the findings. Besides, segmentation can

further aid lesion area estimation and disease severity grading.

The disease guidance module is not intended as a substitute for expert agronomic diagnosis and is a local source of educational knowledge about disease. The suggested cures and prevention information should be checked by experts in the subject, adapted to the local legislation and crop management methods and accompanied by proper disclaimers. There also needs to be a way to build confidence in the model, as well as a technique for detecting out-of-distribution images — highly blurry, or photos of a different kind, or out-of-the-citrus category, or anything else unusual — so that they are sent off to experts rather than wildly overconfident predictions.

V. CONCLUSION

We propose CitraYOLO, an end-to-end platform for citrus disease identification and guidance, comprising of comparative object-detection studies and a deployable web application. A standard dataset approach was used to assess a faster R-CNN, several iterations of the YOLO, and a proprietary Light SA-YOLOv8 architecture. The complementing advantages of the models were revealed by the comparative analysis: YOLOv5x6 has the highest accuracy, YOLOv11 has the highest recall, YOLOv3 has the highest mAP, and the customized SA-YOLOv8 has the best precision, recall, and mAP in a compact architecture. The custom model demonstrates how Fast_C2f, BRA and ASF can be seamlessly integrated into the detector pipeline with just a few lines of code in the YAML-based architecture and custom layer registration.

The Flask application provides provision for the study workflow including the signup, signin, uploading of images, YOLO inference, reporting of confidence, disease description, cause, remedy options, prevention guidance and visualisation of study performance. The study highlights the need for more than just training an agricultural AI model; it also demonstrates that clear interpretation of metrics, comprehensive class mapping, modular deployment, error management, and user-friendly result visualization are essential for successful implementation.

Future studies should include augmentation and rebalancing of the data set, segmentation based on illness severity, and conducting ablation studies on Fast_C2f, BRA, and ASF. AI visualisations like Grad-CAM or detector-specific saliency maps can make AI more transparent. Among other improvements are multi-box result aggregation, calibrated uncertainty, out-of-distribution detection, password hashing, enhanced upload validation, real-time video inference, mobile or edge deployment with the optimised model export and multilingual guidance, as well as production-grade secure hosting.

REFERENCES

- [1] Y. Lin, Z. Huang, Y. Liang, Y. Liu, and W. Jiang, “AG-YOLO: A Rapid Citrus Fruit Detection Algorithm with Global Context Fusion,” *Agriculture*, vol. 14, no. 1, Art. no. 114, 2024, doi: 10.3390/agriculture14010114.

- [2] X. Wu, J. Liang, Y. Yang, Z. Li, X. Jia, H. Pu, and P. Zhu, "SAW-YOLO: A Multi-Scale YOLO for Small Target Citrus Pests Detection," *Agronomy*, vol. 14, no. 7, Art. no. 1571, 2024, doi: 10.3390/agronomy14071571.
- [3] W. Xie et al., "A Detection Algorithm for Citrus Huanglongbing Disease Based on an Improved YOLO Model," *Sensors*, vol. 24, no. 14, Art. no. 4448, 2024.
- [4] Z. Zheng et al., "Improved YOLOv8-Based Algorithm for Citrus Leaf Disease Detection," *IEEE Access*, 2025.
- [5] W. Feng, J. Liu, Z. Li, and S. Liu, "YOLO-Citrus: A Lightweight and Efficient Model for Citrus Leaf Disease Detection in Complex Agricultural Environments," *Frontiers in Plant Science*, vol. 16, 2025, doi: 10.3389/fpls.2025.1668036.
- [6] Q. Dai et al., "YOLOv11-RDTNet: A Lightweight Model for Citrus Pest and Disease Identification Based on an Improved YOLOv11n," *Agronomy*, vol. 15, no. 5, Art. no. 1252, 2025, doi: 10.3390/agronomy15051252.
- [7] X. Guo et al., "Citrus Disease Detection Based on Dilated Reparam Feature Enhancement and Shared Parameter Head," *Sensors*, vol. 25, no. 7, Art. no. 1971, 2025, doi: 10.3390/s25071971.
- [8] R. Xue and L. Wang, "Research on Lightweight Citrus Leaf Pest and Disease Detection Based on PEW-YOLO," *Processes*, vol. 13, no. 5, Art. no. 1365, 2025, doi: 10.3390/pr13051365.
- [9] A. Goyal and K. Lakhwani, "Integrating Advanced Deep Learning Techniques for Enhanced Detection and Classification of Citrus Leaf and Fruit Diseases," *Scientific Reports*, vol. 15, Art. no. 12659, 2025, doi: 10.1038/s41598-025-97159-0.
- [10] P. Dinesh and R. Lakshmanan, "Multiclass Semantic Segmentation for Prime Disease Detection with Severity Level Identification in Citrus Plant Leaves," *Scientific Reports*, vol. 15, Art. no. 21208, 2025, doi: 10.1038/s41598-025-04758-y.
- [11] S. R. M., A. Gladston, and K. N. H., "A Multi-Kernel CNN Model with Attention Mechanism for Classification of Citrus Plants Diseases," *Scientific Reports*, vol. 15, Art. no. 24047, 2025, doi: 10.1038/s41598-025-08557-3.
- [12] H. Zhu et al., "YOLOv8-CMS: A High-Accuracy Deep Learning Model for Automated Citrus Leaf Disease Diagnosis," *Plant Methods*, 2025.
- [13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *Advances in Neural Information Processing Systems*, vol. 28, 2015.
- [14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2016, pp. 779–788.
- [15] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2017, pp. 7263–7271.
- [16] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv:1804.02767*, 2018.
- [17] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv:2004.10934*, 2020.
- [18] G. Jocher et al., "Ultralytics YOLOv5," open-source repository, 2020–2025.
- [19] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information," in *Computer Vision – ECCV*, 2024.
- [20] Ultralytics, "Ultralytics YOLO Documentation and Open-Source Computer Vision Framework," software documentation and source repository, accessed for project implementation.
- [21] C. Qiu et al., "An Automatic Identification System for Citrus Greening Disease (Huanglongbing) Using a YOLO Convolutional Neural Network," *Frontiers in Plant Science*, vol. 13, Art. no. 1002606, 2022, doi: 10.3389/fpls.2022.1002606.
- [22] X. Lyu et al., "Accurate Detection Algorithm of Citrus Psyllid Using the YOLOv5s-BC Model," *Agronomy*, vol. 13, no. 3, Art. no. 896, 2023, doi: 10.3390/agronomy13030896.
- [23] P. Dhiman, A. Kaur, Y. Hamid, E. Alabdulkreem, H. Elmannai, and N. Ababneh, "Smart Disease Detection System for Citrus Fruits Using Deep Learning with Edge Computing," *Sustainability*, vol. 15, no. 5, Art. no. 4576, 2023, doi: 10.3390/su15054576.
- [24] J. C. F. da Silva, M. C. Silva, E. J. S. Luz, S. Delabrida, and R. A. R. Oliveira, "Using Mobile Edge AI to Detect and Map Diseases in Citrus Orchards," *Sensors*, vol. 23, no. 4, Art. no. 2165, 2023, doi: 10.3390/s23042165.
- [25] I. Pacal et al., "A Systematic Review of Deep Learning Techniques for Plant Diseases," *Artificial Intelligence Review*, vol. 57, Art. no. 304, 2024, doi: 10.1007/s10462-024-10944-7.
- [26] A. Upadhyay, N. S. Chandel, K. P. Singh, et al., "Deep Learning and Computer Vision in Plant Disease Detection: A Comprehensive Review of Techniques, Models, and Trends in Precision Agriculture," *Artificial Intelligence Review*, vol. 58, Art. no. 92, 2025, doi: 10.1007/s10462-024-11100-x.
- [27] M. Shoaib, A. Sadeghi-Niaraki, F. Ali, I. Hussain, and S. Khalid, "Leveraging Deep Learning for Plant Disease and Pest Detection: A Comprehensive Review and Future Directions," *Frontiers in Plant Science*, vol. 16, Art. no. 1538163, 2025, doi: 10.3389/fpls.2025.1538163.
- [28] G. Cong et al., "YOLOv8-Scm: An Improved Model for Citrus Fruit Sunburn Identification and Classification in Complex Natural Scenes," *Frontiers in Plant Science*, vol. 16, Art. no. 1591989, 2025, doi: 10.3389/fpls.2025.1591989.
- [29] Y. Liao, L. Li, H. Xiao, F. Xu, B. Shan, and H. Yin, "YOLO-MECD: Citrus Detection Algorithm Based on YOLOv11," *Agronomy*, vol. 15, no. 3, Art. no. 687, 2025, doi: 10.3390/agronomy15030687.