

A Scoring App for the Efficient Recording, Analysis and Documentation of Pest Infestation on Plants

Matthias Becker¹[0000-0003-4358-1258] and Alexandre Bertin¹

Leibniz University Hannover, Welfengarten 1, 30167 Hannover, Germany
`xmb@hci.uni-hannover.de`

Abstract. Assessing pest infestation in agriculture is essential but labor-intensive and time-consuming, often involving manual counting of insects on plant leaves. This process is prone to errors and can become physically demanding. To address this, a mobile application was developed to automate pest detection and streamline data documentation using computer vision. The app allows users to take pictures of plants, enabling an object detection algorithm to identify and count pests automatically.

The application includes several key features: QR-code scanning and generation for automatic metadata input (e.g., field, plant ID), pest identification and counting, results aggregation and email sharing, editable recipient lists, and a user-friendly, weather-readable interface. Developed in React Native with the Expo framework, the app supports both Android and iOS platforms and enables real-time testing.

At the core of the app is a YOLOv5 object detection model trained on annotated datasets of whiteflies, using Roboflow and Docker for back-end deployment. Initial training showed potential, prompting further improvement using public datasets. However, larger and more diverse datasets did not consistently yield better results, leading to the decision to retain weights from the earlier, more effective training. The application ultimately aims to make pest monitoring more efficient, accurate, and accessible for farmers.

Keywords: whitefly · scoring · pest detection and classification.

1 Introduction

1.1 Motivation

The assessment of pest infestation levels in agricultural fields is a necessary, yet tedious process. To ensure the right measures are taken to prevent parasites from destroying crops and spreading diseases, farmers must identify and count the pests present in their fields; manually counting hundreds of tiny insects on dozens of leaves can take up to several hours, does not guarantee foolproof results and can turn into exhausting physical labor.

Multiple steps of the pest assessment process seem to have a noticeable automation and streamlining margin: the species identification and the counting

of individuals itself are tasks that could be handled by modern computer vision algorithms, and the process of documenting the results, a crucial step in tracking the evolution of infestation levels, could easily be automated.

Hence the idea of developing an application for mobile phones, which would allow its users to simply capture pictures of the plants and let computer vision algorithms detect and count the pests for them.

1.2 Objectives and specifications

Though it remains the main motivation for this application, and its core purpose, the application should not only count parasites: it should be designed to tackle the diverse issues regarding manual pest assessment, in order to streamline this process as much as possible by making it more efficient and less challenging. Given these objectives, the application should aim to allow for:

- **QR-Code scanning, generation and export for automatic plant data entry** To classify the parasite counting results, the pictures have to be labeled using various parameters (location, field, block number, treatment, plant type, plant ID). In addition to allowing the user to manually input those parameters, the application should allow them to scan a QR-Code to automatically retrieve the data. It should also offer the user the possibility of generating such QR-Codes given input parameters, and exporting them for later use.
- **Automatic pest detection, identification and counting** The application should be able to identify and count the exact population of each parasite on plant leaves pictures taken by the user.
- **Sharing of classified results by e-mail** The application should compute total infestation levels across the various fields/blocks by classifying and aggregating the results of the object detection algorithm, and export the results in the form of a readable document shared via an e-mail sent to selected addresses.
- **Editing e-mail addresses** The application should let the user add/remove e-mail addresses from the recipient list for QR-Codes and object detection results exports.
- **Intuitive, user-friendly and contrasted interface** The application should provide a simple and readable interface which highlights its functionalities, and has a clear color contrast to make it usable in all weather conditions.

2 Related Approaches and Previous Work

Several recent studies have explored the use of computer vision and deep learning for automated detection of plant diseases and pests. Each approach offers distinct contributions in terms of dataset type, model architecture, platform integration, or targeted crops.

Liu et al. [2] applied deep convolutional neural networks to identify apple leaf diseases, demonstrating the effectiveness of CNNs in distinguishing between

subtle visual differences on plant surfaces. Unlike most others, this study focused specifically on apples and did not integrate mobile deployment. In another approach, [7] applied CNNs to rice diseases and emphasized classification accuracy on a single crop. Their model was not designed for real-time pest counting or broader application scenarios.

In [3], the authors evaluated multiple CNN architectures across a wide range of crops and diseases, showing high accuracy and robustness in plant disease detection. This work stands out for its comprehensive comparison of deep learning models, yet lacks a real-time or mobile implementation.

A smart mobile system combining deep learning with smartphone hardware has been introduced in [4], making disease recognition more accessible in the field. The focus was broader on diseases in general, rather than specific pests.

In [5] a robust object detection system for identifying both tomato pests and diseases has been developed, using real-time deep learning models. It is particularly relevant due to its combined pest-disease focus and object detection framework, which resembles our approach using YOLOv5.

Building trust is important for real-life applications, since the decisions based on the machine learning results can have a huge impact on economic success. Thus in [6] interpretability has been contributed through saliency map visualizations, helping users understand the decision-making process of CNNs. This approach is valuable for trust-building but is not optimized for mobile or field conditions.

Finally, in [8] usability has been emphasized by collecting real-world data using mobile phones in uncontrolled conditions. This study is particularly relevant for field applications, though it focused more on disease classification than on precise pest quantification.

In contrast to the above works, our approach combines real-time pest detection and counting with streamlined data documentation through QR-code integration and mobile app functionality—targeting both usability and automation for field technicians and researchers.

In previous works we already developed approaches for automated pest detection and classification inside greenhouses [1]. In that approach we developed an integrated hardware that automatically takes pictures, analyses the species and their numbers and reports automatically. However those approaches can not be used outside.

3 Application development

3.1 Tools and framework

The app was developed using React Native [9] in order to make it accessible on both Android and iOS platforms, in an Expo [10] framework: in addition to providing several integrated modules that fit our application’s needs, such as camera access and file system operations, Expo also streamlines the development process thanks to the Expo GO app, which enables us to live test our app on mobile phones during development with hot reload on changes.

3.2 Object detection algorithm

At the core of the application, an object detection algorithm is what enables the automatic pest count. Our choice for this task was to train a YOLOv5 [11] object detection model on an annotated whitefly dataset using Roboflow [12], and run this model on the app's backend in a Docker container [13].

3.3 Deploying the application

To deploy the application and allow users to use it anywhere at any time, a server was provided by the Gottfried-Wilhelm-Leibniz University of Hannover to permanently host the backend. The server was set to automatically restart on crashes and reboots to ensure uninterrupted uptime.

4 Performance analysis

4.1 Current features

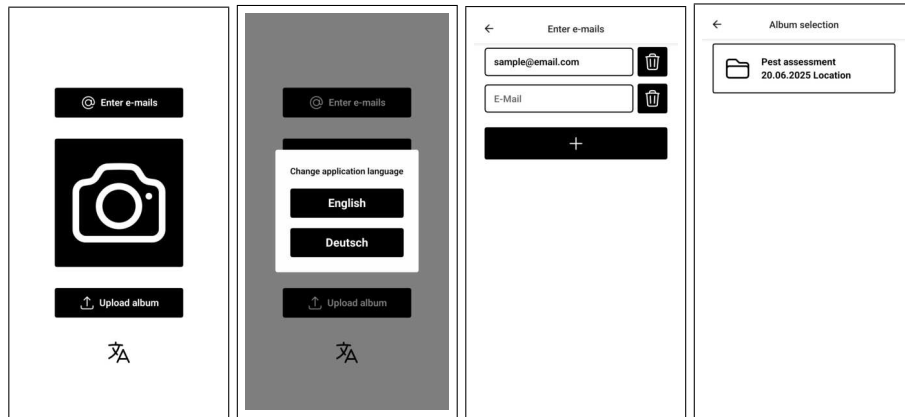
The features of the application, in its current state, are the following:

- A main menu in Fig. 1a, where the user can access the various pages and change the language of the application (Fig. 1b)
- An "E-Mail" menu (Fig. 1c), where the user can edit the recipient list for analysis results and QR-Code exports
- A "Load Album" menu (Fig. 1d), where the user can select a saved photo album and upload it to the application's backend to trigger the automatic pest detection process (the application offers to delete successfully uploaded albums), Fig. 1e
- A "New Album" menu (Fig. 1f), where the user can create a new photo album corresponding to a pest assessment measure. They will be required to enter the plant data first (by hand, or by scanning a QR-Code), and have the possibility to generate and export a QR-Code corresponding to the given data (Fig. 1g). The user can then take all the pictures relevant to this pest assessment measure to create an album which they can later upload via the "Load Album" menu

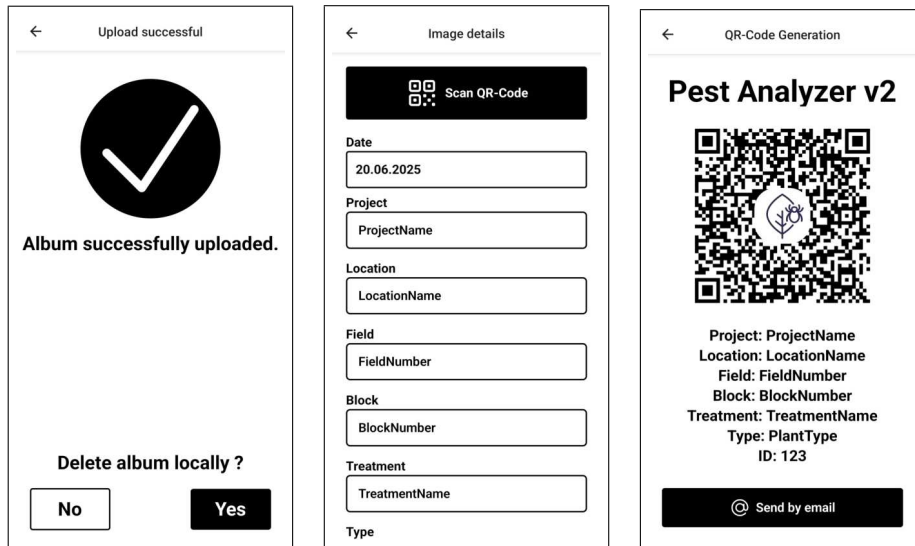
4.2 Whitefly detection

The first object detection model we used was trained on a personal dataset containing 211 original photos taken in local fields and greenhouses, and augmented by 90° rotations and mirroring. These pictures were hand-annotated, meaning the Roboflow annotation interface was used to mark all whiteflies on every picture. This first training resulted in satisfying performances, with a recall score of 65% and a precision score of 76%.

However, this first model saw its performance drop when confronted with unusual data, meaning different plants : while it was trained on cultures found



(a) Main menu (b) Language (c) E-Mail menu (d) Album menu



(e) Successful upload (f) Plant details (g) QR-Code generation

Fig. 1: Screenshots of the application

in northern Germany, it lead to much poorer performances when tested on images of cassava leaves. To decipher whether this was due to a specialization of the model on its training data, or to a possible difficulty inherent to the processing of cassava leaves, we decided to train another model with the same tools, but this time on images of cassava leaves. We decided to leverage a portion of a public dataset of annotated images of whiteflies on cassava leaves [14] and trained the model on 990 images with a 70/20/10 train/val/test split, for 100 epochs. To test the performances of this newly trained model, and compare them to that of our first model, we benchmarked their performances on a set of 10 images of the cassava dataset (unseen during training). The results were the following:

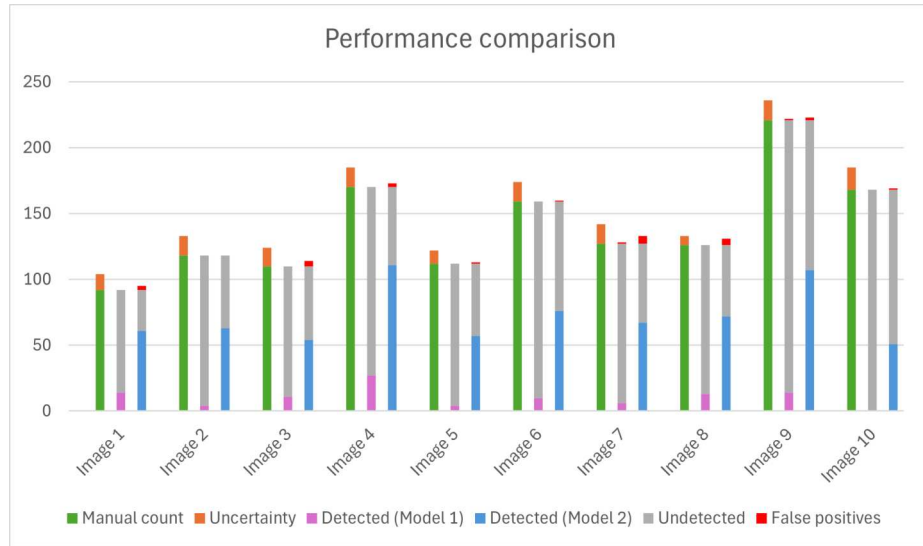


Fig. 2: Whitefly detection results on the 10 test images

The first model showed a recall rate of 07.57% with a precision of 97.67%, while the second model increased the performance to a recall of 52.14% with a precision of 96.43%. Though the recall rate is still far below the requirements for the application’s use cases, this comparison shows the impact of variety in the training dataset : a model that performs well on a specific type of leaves that resemble those which it was trained on can perform poorly on different leaves, while another model that was trained using those different leaves can achieve good scores.

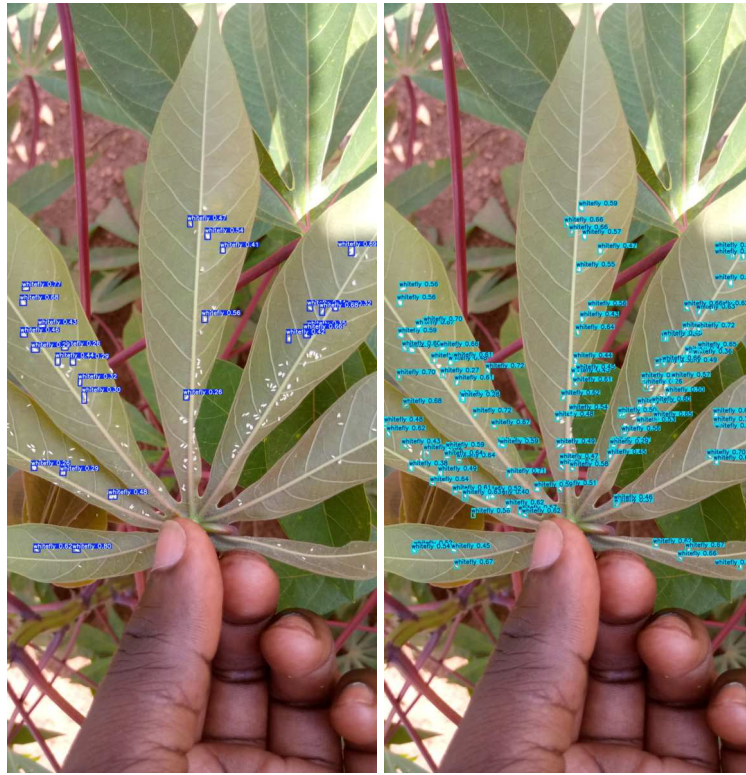


Fig. 3: Image 4 of the test sample, analyzed by model 1 (left) and model 2 (right)

5 Possible improvements

5.1 Object detection performance

The current performances of the application's object detection model being currently insufficient, new trainings must be envisioned. Our previous results highlight the limitations of training models on specific types of leaves, thus inviting us to come up with different approaches to better tackle the issue of plant variety.

A few possible solutions could be the following :

- **Image tiling** A solution to ensure consistency of the results across various plant types would be to let the object detection run on a more local scale : instead of processing the pictures of the entire leaves, they would be split into smaller tiles, each presenting a few whiteflies as the main point of interest while the leaf acts as a background rather than an object itself. This would help reduce the level of details on the picture, making it easier for the model to focus on the whiteflies, and would allow to reduce any type of plant into familiar elements for the model.

- **Classification layer & specialized models** Another solution would be to train multiple specialized models, each trained to detect whiteflies on a specific type of plant. The application could then call the relevant model for each image to process, be it via user input (specifying which plant is being studied), or by submitting the image to another model trained to identify plant leaves to automate the process of calling the specialized model.

Although the second option could eventually offer the best results, it also seems more rigid and more demanding in terms of data : each type of plant would require its own large training dataset, and results might drop in case of unknown plants. The first one, however, could be rather promising, and has the advantage of making any whitefly dataset relevant : training images should be split into smaller tiles, thus reducing species-related specifics and, as such, allowing us to merge multiple datasets into a substantially large one.

5.2 Testing

The next step in this app’s development would be to deploy it for a few clients and to monitor their use and appreciation of the application, by gathering their feedback, and comparing the efficiency and accuracy of their pest assessments with and without the app.

The application was originally imagined in the context of a partnership with the Institute of Horticultural Production Systems at the University Hannover, where we also did the requirements engineering process and data collection, and such a test deployment is currently being discussed.

6 Conclusion

Manual pest assessment in agriculture is a labor-intensive, time-consuming, and error-prone process that poses challenges for both precision and scalability. This project introduces a mobile application that leverages modern computer vision techniques to automate pest detection and quantification, aiming to simplify the process and improve data quality. By incorporating functionalities such as QR-code-based plant data entry, automatic pest counting using YOLOv5, and streamlined result sharing via email, the application provides a user-friendly and field-ready solution.

Compared to existing approaches in the literature, which often focus either on disease classification or lack mobile integration, this system uniquely combines real-time pest detection with practical features designed specifically for field use. While the object detection model showed high precision and moderate recall after multiple training phases, further improvements are needed to increase reliability across diverse environmental conditions and species types.

The app has been successfully deployed with backend support provided by Leibniz University Hannover, ensuring accessibility and continuous availability.

Despite current limitations in detection accuracy, the system represents a significant step toward efficient, scalable, and accessible pest monitoring for agricultural professionals. Future work will aim to extend pest species coverage, enhance detection performance, and integrate active feedback mechanisms from users in the field.

References

1. M. Becker and K. Yeow, "An Automated Monitoring System for Controlled Greenhouse Horticulture," in *Proc. Int. KES Conf. on Intelligent Decision Technologies*, Springer, 2024, pp. 75–85.
2. B. Liu, Y. Zhang, D. He, and Y. Li, "Identification of apple leaf diseases based on deep convolutional neural networks," *Symmetry*, vol. 10, no. 1, p. 11, 2016.
3. K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," *Computers and Electronics in Agriculture*, vol. 145, pp. 311–318, 2018.
4. M. A. Mohammed, M. K. Abd Ghani, H. Hameed, S. Mostafa, B. Garcia-Zapirain, E. Abdulhay, and A. K. Sangaiah, "Smart mobile system for plant disease diagnosis using deep learning," *Computers and Electronics in Agriculture*, vol. 170, p. 105254, 2020.
5. A. Fuentes, S. Yoon, S. C. Kim, and D. S. Park, "A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition," *Sensors*, vol. 17, no. 9, p. 2022, 2017.
6. M. Brahim, M. Arsenovic, S. Laraba, S. Sladojevic, K. Boukhalifa, and A. Mousaoui, "Deep learning for plant diseases: detection and saliency map visualisation," in *Human and Machine Learning*, Springer, pp. 93–117, 2019.
7. Y. Lu, S. Yi, N. Zeng, Y. Liu, and Y. Zhang, "Identification of rice diseases using deep convolutional neural networks," *Neurocomputing*, vol. 267, pp. 378–384, 2017.
8. A. Picon, A. Alvarez-Gila, M. Seitz, A. Ortiz-Barredo, J. Echazarra, and A. Johannes, "Deep convolutional neural networks for mobile capture device-based crop disease classification in the wild," *Computers and Electronics in Agriculture*, vol. 161, pp. 280–290, 2019.
9. React Native Homepage, <https://reactnative.dev>, last accessed 2025/06/18
10. Expo Documentation, <https://docs.expo.dev>, last accessed 2025/06/18
11. YOLOv5 Documentation, <https://docs.ultralytics.com/yolov5>, last accessed 2025/06/18
12. Roboflow Documentation, <https://docs.roboflow.com>, last accessed 2025/06/18
13. Docker Documentation - "What is a container?", <https://www.docker.com/resources/what-container>, last accessed 2025/06/18
14. Cassava Whitefly Dataset, <https://data.mendeley.com/datasets/5g38399z9p/2>, last accessed 2025/06/18
15. Datasets of adult whiteflies on tomato leaflets, https://figshare.com/articles/dataset/Datasets_of_adult_whiteflies_on_tomato_leaflets/22109012?file=39300056, last accessed 2025/06/18