# Smart Farming with Computer Vision: Detecting Diseases in Strawberry Crops via Mobile Applications

Mateus R. Cruz, Adriana M. F. Grael, Guilherme P. Piedade, and Samuel B. Mafra

***Abstract*— Strawberries are red fruits and an excellent choice for a healthy and delicious meal. They are produced from a sweet flower and are considered an excellent source of vitamin C, vitamin A, and potassium. Strawberry production in Brazil is concentrated in the states of São Paulo, Minas Gerais, and Rio Grande do Sul. According to Embrapa, in 2016, these states produced about 2.5 million tons, which corresponds to almost 60% of national production. Similar to other fruits, strawberries can be affected by various problems, such as diseases, pests, and adverse weather conditions. The use of pesticides in strawberry production is quite common, as strawberries are very susceptible to pests and diseases. However, pesticides can be harmful to human health, so it is important that they are used safely. Currently, diseases are detected by growers by eye and identified through the experience of the grower. However, less experienced farmers may encounter problems in identifying and even detecting diseases present in the crop. Computer Vision is a field of computing that studies how to represent and manipulate information about objects and environments in a way that allows computers to "see" the world similar to humans. The application of vision techniques for disease detection allows any farmer to obtain knowledge about the disease present in the plantation. This study proposes the use of Convolutional Neural Networks in mobile applications for the detection of diseases present in strawberry crops.**

***Keywords*— Computer Vision, Mobile, Edge Computing, Smart Farming**

## I. Introduction

Strawberries are a popular fruit consumed in almost every part of the globe, and this expansion in production is a result of the plant's ability to adapt, enabling it to be grown in a variety of climates. Not only this, but the efforts made by growers, students and scientists have been essential for the generation of several adaptive production systems. Hence, it is now possible to grow the fruit even in region-specific local conditions [1]. Thus, the production and marketing of strawberries is present in 76 countries, reaching a global production of over 7.7 million tons in 2013 [1]. Moreover, this significant increase in the consumption of fruit during the 21st century, added to the improvements achieved through agricultural innovation, makes it possible to make it available throughout the year. Given the current possibilities, strawberries tend to have a continuous demand from consumers over the years. Currently, Brazil is the largest producer of strawberries among the countries of South America [2].

However, one of the challenges to be faced will be in producing the fruit in an economical and sustainable way [1]. Not only that, problems related to diseases and pests are common and can be devastating to the plantation. As an example, Anthracnose (a disease caused by Colletotrichum spp) is the most destructive disease present on strawberry plants in Brazil [2]. Because of this, two approaches are commonly used by growers: prevention and combat. Prevention is associated with good practices in the cultivation of the plant, such as (i) Acquisition of seedlings from reliable suppliers, besides control and monitoring of water quality [3]. Combat, on the other hand, is necessary during more advanced stages of diseases and pests, where control may not have been sufficient or correctly applied. Moreover, most of the combat processes use chemical products for treatment [3] However, the use of these inputs brings several harms to the health of the farmer and consumer, and may develop problems in farmers such as hearing loss and congenital malformations through the alteration in the mother-fetus binomial [4]. Not only that, the correct identification of the disease for the appropriate choice of input usually requires specialized professionals or prior knowledge on the part of the producer. Moreover, certain processes may require laboratory analysis, adding time to the detection of the disease. Thus, disease detection processes in the traditional way can be time-consuming, leaving the crop unprotected during this interval. Knowing this, several researchers in the field of Artificial Intelligence (AI) have proposed the use of algorithms to automate the recognition of pests and diseases in crops.

Computer vision (CV) is a sub-field of AI that aims to provide visual capabilities to machines. Not only that, a large portion of diseases cause visual changes in plants, making it possible to develop solutions that use CV in agriculture [5]. Currently, systems based on neural networks are able to detect and classify in real-time diseases found in the crop [6]. Through these algorithms, several diseases can be quickly rec-

Mateus Raimundo da Cruz, Instituto Nacional de Telecomunicações (Inatel), Santa Rita do Sapucaí (MG), e-mail: mateusrc@dtel.inatel.br; Adriana Maria Fuzer Grael, Escola Superior de Agricultura "Luiz de Queiroz" da Universidade de São Paulo (ESALQ/USP), Piracicaba (SP), e-mail: tinos.amfg@gmail.com; Guilherme Pires Piedade, Instituto Nacional de Telecomunicações (Inatel), Santa Rita do Sapucaí-MG, e-mail: guilherme.pires@mtel.inatel.br; Samuel Baraldi de Mafra, Instituto Nacional de Telecomunicações (Inatel), Santa Rita do Sapucaí (MG), e-mail: samuelbmafra@inatel.br; This work is partially supported RNP, with resources from MCTIC, Grant No. 01245.020548/2021-07, under the Brazil 6G project of the Radiocommunication Reference Center (Centro de Referência em Radiocomunicações - CRR) of the National Institute of Telecommunications (Instituto Nacional de Telecomunicações - Inatel), Brazil, Huawei, under the project Advanced Academic Education in Telecommunications Networks and Systems, contract No PPA6001BRA23032110257684, Brazil, the National Council for Scientific and Technological Development-CNPq (403827/2021-3), FAPESP (2021/06946-0),and by Minas Gerais State Agency for Research and Development (FAPEMIG) via Grant No. TEC - APQ-03283-17.

ognized and their proper corrections applied, speeding up the fight against disease in the plantation. Several researchers have already proposed solutions that make use of neural networks for disease detection for a variety of crops, such as tomatoes [7], corn and many others [6]. However, many of the proposed solutions do not offer an easy-to-implement condition. These conditions prevent less tech-savvy users from enjoying the benefits offered by the technological advances offered by the advancement of AI and CV. Therefore, the proposed solution aims to bring to the user an Android application that can be installed on the smartphone and detect diseases present in strawberries. Not only that, convolutional neural networks will be used for the classification of the seven diseases most commonly found on strawberry trees. In this way, farmers who own Android smartphones will be able to make immediate use of the application and thus take better care of their crops [10].

The application of Yolo V5 for disease detection in strawberries using the same dataset has already been proposed in the literature [11]. However, the implementation scenario considered the use of a wireless sensor network using LoRa in an IoT approach. This type of system is able to capture, store, and automate the process of pest detection in the plantation, but it requires a prior investment as well as installation and configuration of the WSN. This application makes the process simpler by bringing the VC model to the farmer's mobile device. Today a large part of the population owns a smartphone with enough technology and computing power to run AI applications in real-time.

The article is structured into four sections. Section II provides a comprehensive description of the proposed application, outlining the various steps involved in training, test, and implementation. This section offers detailed insights into the processes and methodologies utilized. Moving on to Section III, the results obtained from the application are presented and thoroughly discussed, exploring the outcomes in-depth, and highlighting their significance and implications. Finally, Section IV concludes the article by summarizing the key findings and offering suggestions for future research directions.

## II. YOLO Training, Test, and Mobile Implementation

The developed application aims to implement on mobile devices neural network models capable of detecting diseases present in strawberries. Currently one of the tools available, and TensorFlow Lite library was the one used in the proposed application to implement object detection models. The library can be used for developing Android applications that aim to classify or detect objects in real-time. Figure 1 presents an illustration of the application in camp.

Object detection was the technique chosen for the strawberry disease recognition task. This technique consists in creating bounding boxes around the classified subject to network and also presents which subject is classified in the image. In this way, the user is able to simultaneously identify different subjects present in a single image. Not only that, it is possible to visualize, in percentage form, the confidence level that the network presents when performing a certain classification.


Fig. 1. Illustration of the application.

The network chosen for detection was the You Only Look Once [Yolo] in its fifth version because it is an architecture developed to also serve mobile devices and lower computing power. However, the network is developed using the PyTorch library, an open-source library developed in Python for Machine Learning (ML), Deep Learning, and CV tasks. Therefore, it is necessary to convert the generated weights to a format compatible with the one used by the TensorFlow Lite library that will be used to run the model at the edge.

The development of the proposed application was divided into three main stages: (I) Training and test of the detection model, (II) Development of the mobile application, and (III) Implementation of the model in the application. The first stage aimed to train a performative model for the detection of diseases present in strawberries as well as validate its performance through metrics such as accuracy, and precision, among others.

The training of a detection model is a computationally demanding task and therefore has as a prerequisite the presence of dedicated graphics processors. However, there are currently free tools, such as Google Colab, that offer the user remote access to computers sufficiently capable of running the training of a neural network for image detection. However, the amount of time available to the resource is limited, and it is necessary to adjust the network parameters so that the time spent during training is compatible with the time allotted to use the resource.

The main parameters configured during the model training were the number of epochs and the batch size. The number of epochs sets the number of times the model will go through the training data set, while the batch size sets the number of training samples to be worked on before the weights are updated on the net. The parameters used during training were 100 epochs and a batch size of 16.

To train a network for object detection it is necessary to have or develop a suitable image bank that can represent the subject in various light conditions and locations, as well as at different angles. In addition to images with good quality and variety, it is necessary to have a good amount of samples (images) of the subjects (classes) that you want to detect and classify. It is also necessary to label the subjects contained in the images you wish to detect, and this step is the most time-consuming for the

developer. The image bank used in the proposed solution was developed and labeled by members of the AI lab, Computer Science and Engineering department, JBNU [5]) and published on the publicly accessible Kaggle.com website. The image bank features a total of seven diseases present in the strawberry crop. They are: I) Angular leafspot, II) Anthracnose, III) Gray Mold, IV) Mycosphaerella spotted, V) Powdery mildew leaf, VI) Powdery Mildew fruit, and VII) Blossom blight.

An important feature to be observed when using ready-made image banks is the representatives of each class. For this, it is necessary that each class has, on average, the same amount of samples as the others. If this criterion is obeyed, problems related to performance discrepancies between different classes can be avoided. Therefore, further analysis was performed in order to verify how many images per class are present. Table I displays the numbers obtained.

TABLE I
NUMBER OF SAMPLES PER CLASS PRESENT IN THE DATASET.

| Class | Quantity of Images |
|---|---|
| Angular Leafspot | 500 |
| Anthracnose Fruit Rot | 111 |
| Blossom Blight | 208 |
| Grey Mold | 591 |
| Powdery Mildew Fruit | 684 |
| Powdery Mildew Leaf | 172 |
| Leaf Spot | 534 |
| Total | 2800 |

The entire image bank was sliced, making it possible to train and validate the model without the need for new images. The sliced configuration meant that 80% of all available images were used for model training and 20% of the total images were used for model test. Metrics related to the performance of each class (disease) were captured during the training and test phase, making it possible to visualize the predictive ability of the model with respect to each disease. A graph was generated to better visualize the evolution of the model over each epoch.

With training completed, a mobile application for Android devices is required to run the model. The Google team responsible for developing TensorFlow Lite provides a sample application for download on Github [4], but trained with the Common Objects in Context (COCO) image bank for the detection and classification of 80 different subjects. The available sample application was used for the development of the proposed application, and some modifications were made to allow the application to run the previously trained model.

The application runs completely on the edge, at no time requiring an Internet connection for its operation. This is essential for plantations and remote locations where there is no cellular or Internet coverage yet. The application also allows the user to determine the number of threads (smallest processing unit) that will be occupied by the application. If the user chooses to increase the number of threads, the inference time of the model will be shorter, thus ensuring a higher number of detections per second.

The last step consisted of implementing the trained model in the Android application, and the weights had to be converted for its implementation. The trained model was generated from the Yolo v5 network that uses the PyTorch library, and the application uses the TensorFlow Lite library. Therefore, it is necessary to convert the weights of the generated model into a format that the application can use. To do this, we used a program written in Python available on the Yolo v5 home page that allows the generated model to be exported to other known formats, including the one used by the TensorFlow library. Not only that, the program was also used to transform the TensorFlow format into its Lite version, the same version used for mobile devices such as microcontrollers, smartphones, and others.

Metrics were used to measure the performance of the model over and after the training run. Each metriMeasures the harmonic mean of precision and recall, giving equal weight to both metrics. It is used in situations where achieving a balance between precision and recall is important.c has its own peculiarity, thus being able to demonstrate various characteristics of the model trained and proposed for implementation. The metrics used and their mathematical equations are described below.

- **F1-Score**: Measures the harmonic mean of precision and recall, giving equal weight to both metrics. It is used in situations where achieving a balance between precision and recall is important.

$$\text{F1-Score} = \frac{2 * TP}{2 * TP + FN + FP} \ [\%]. \tag{1}$$

- **Precision**: Precision measures the accuracy of the model in its predictions. The result is presented through the percentage of correct predictions made.

$$\text{Precision} = \frac{TP}{TP + FP} \ [\%]. \tag{2}$$

- **Recall**: Metric that points to the ability of the model to detect all relevant classes within the data set.

$$\text{Recall} = \frac{TP}{TP + FN} \ [\%]. \tag{3}$$

The metrics captured during training are often better than those captured during the test stage. This is because the model views the entire image set N times (epochs) during its training, making the detection and correct classification of each class more predictable. Whereas test presents the model with a portion of the image bank that has not yet been used, making the presented metrics more similar to that of a real scenario.

## III. RESULTS AND DISCUSSION

The model was first tested on the training dataset and then validated using the test dataset. During training, excellent results were observed regarding the models' ability to distinguish between different classes present in the database. Figure 2 demonstrates the performance of the model during training. The results presented during training show a good ability of the model to correctly detect and classify each of the classes contained within the image base. The model presented False Positive Background (FP background) which are detections and classifications of "objects" in the background that should

not have been classified, and also False Negative Background (FN background) which are classes that were not classified but should have been. It is then concluded that the reason for the declining performance of the model during training is due to the inability to detect all the diseases present in the image since those detected were correctly classified overall. The model was also validated using a test data set. This step is extremely important, as it assesses the model's ability to detect and classify samples not yet presented during training. Also, the ability to detect and classify correctly in this step demonstrates how well the model will perform in a real-world scenario. There was a decrease in the performance of the model when presented with a set of unseen images. However, in general, the model was able to maintain a good ability to distinguish between the presented classes. The deficiency, already present in the training results, of not detecting all diseases and at the same time misdetecting diseases is still present. However, the result presents a reasonable performance to be applied and tested in real situations.

The next step is to implement on a smartphone, capture, and evaluate the metrics of the application. This because, the model will make use of the computational power of the device to perform disease detection and classification. The technical data of the device used to test the application are shown in Table II.

TABLE II
MOBILE DEVICE CONFIGURATION.

| Component | Configuration |
|---|---|
| Processor | 1.8 GHz 8 Core |
| Chipset | Snapdragon 450 |
| System | 64 bits |
| Grafical Processor Unit | Adreno 506 |
| RAM Memory | 3 GB |
| Camera Resolution | 4163 x 3122 pixel |

The smartphone has basic settings and limited computing power but was able to run the application. However, it is possible to use hardware acceleration features to speed up the prediction calculations. In this case, TensorFlow Lite makes use of software modules that accelerate the execution of ML models using specialized processing hardware available in the handset, such as Graphics Processing Units (GPUs), Tensor Processing Units (TPUs), and Digital Signal Processors (DSPs). Graphics processing units (GPUs) are hardware devices that perform graphics processing operations and are usually located on computer video cards. Tensor processing units (TPU) are chips that specialize in performing tensor processing operations, such as matrix multiplication. They are designed to work in parallel and provide high performance in AI applications, such as ML. Digital Signal Processors (DSP) are integrated circuits that perform input signal manipulation and are designed to operate at high speeds. DSPs are used in a variety of applications, including telecommunications, audio and video processing, motor control, and measurement. Although it is possible to use, it is not necessary to have such units to run the application. Not only that, but older smartphones may not have such processing components, limiting the number of users that can enjoy the application. Therefore, Central Processing Units (CPU) were used to process the data in this application.

Inference time is a metric that measures the rate of frames processed by the model during disease detection and classification. This information is presented in the application's configuration menu on the main screen. Several samples of the inference time were captured in order to evaluate the performance of the application on the test handset. The number of processing Threads was changed. Table III presents the captured samples and also the average times.

TABLE III
VALUES CAPTURED DURING THE BENCHMARK, 10 SAMPLES OF EACH METRICS AND THE MEDIAN OF THE VALUES.

| Sample | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|
| 1 | 1206 | 1052 | 959 | 829 | 941 | 1134 | 1074 |
| 2 | 1242 | 1059 | 947 | 909 | 937 | 1117 | 1164 |
| 3 | 1221 | 1045 | 989 | 926 | 938 | 1149 | 1147 |
| 4 | 1216 | 1053 | 979 | 915 | 952 | 1093 | 1179 |
| 5 | 1232 | 1082 | 964 | 911 | 932 | 1088 | 1143 |
| 6 | 1217 | 1052 | 954 | 914 | 949 | 1172 | 1170 |
| 7 | 1208 | 1045 | 926 | 915 | 944 | 1131 | 1105 |
| 8 | 1207 | 1089 | 945 | 919 | 954 | 1100 | 1158 |
| 9 | 1229 | 1061 | 950 | 920 | 950 | 1035 | 1253 |
| 10 | 1216 | 1118 | 928 | 912 | 938 | 1123 | 1150 |
| Median | 1219.4 | 1065.6 | 954.1 | 907 | 943.5 | 114.2 | 1154.3 |

Threads are a form of communication between two or more processes. Threads allow different parts of a program to communicate with each other without having to share the entire memory space of the program. Processing threads are a way of dividing the execution of a program into multiple parts so that different parts of the program can be executed at the same time. This can improve overall program performance since it allows the program to take advantage of multiple CPU cores. On the specific smartphone, the application performed best with a setting of 6 threads. It is noticeable that values higher and lower than 6 presented an increase in the inference time of the model, disturbing the fluidity in the use of the application. The CPU temperature present in the smartphone, despite having presented a noticeable increase during the use of a high value of threads, did not exceed 62ºC at any time. The CPU usage during the use remained at 100%, presenting great stress for the mobile device when running the application.

## IV. CONCLUSIONS

The proposed application makes it possible to run models that make use of the Yolo V5 architecture on devices with limited computing power in a satisfactory manner. The need to run models at the edge is due to the lack of coverage of mobile networks, not guaranteeing access to all locations. This limitation, in turn, makes it impossible to deploy and use CV models in the cloud on remote farms and plantations. However, a large portion of people nowadays have access to a mobile device, making it possible for these technologies to be implemented and leveraged on these devices. The proposed application was able to perform disease detection

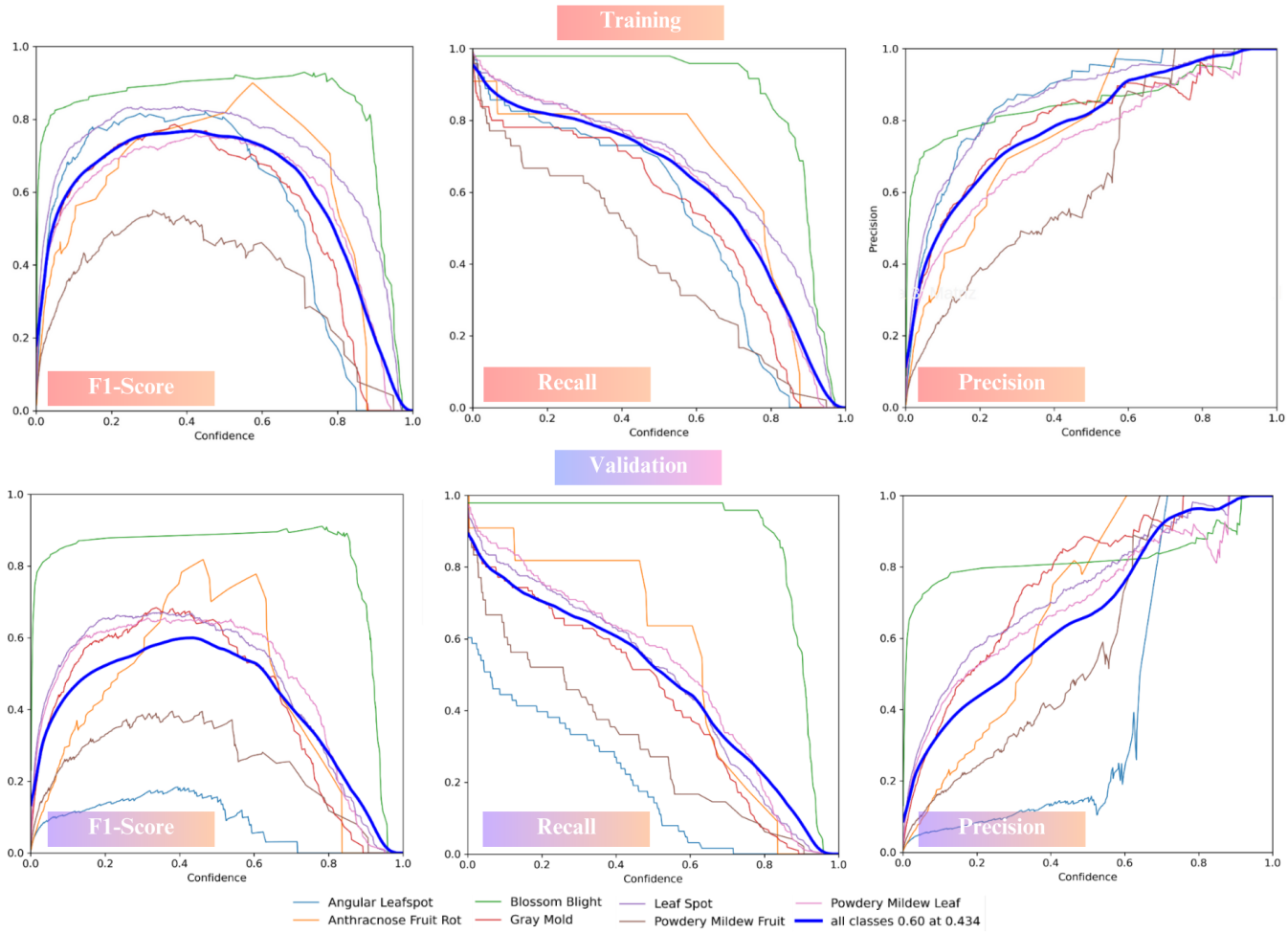


Fig. 2. Yolo v5 metrics in the training and test dataset.

and classification on strawberries, even on a mobile device limiting technical aspects. Not only that, even without the use of GPU it was able to keep the inference time of the multiple disease model simultaneously less than 1.3 seconds. Although there was an increase in CPU temperature, at no time did this temperature make it impossible to use the application, causing crashes or shutdowns. The model presented satisfactory results, even using limited weight sizes. The model is usable even though the model presented some problems for the detection and correct classification of some diseases, mainly Angular Leafspot and Powdery Mildew Fruit. For future work, it is necessary to use a larger image base than the one used and fine-tune the network parameters.

## REFERENCES

[1] H. Timo, J. Graham, and R. Harrison, "The genomes of rosaceous berries and their wild relatives. Cham", Switzerland: Springer, 2018.

[2] V. F. Soares, A. C. Velho, A. Carachenski, P. Astolfi, and M. J. Stadnik, "First Report of Colletotrichum karstii Causing Anthracnose on Strawberry in Brazil," Plant Disease, vol. 105, no. 10. Scientific Societies, p. 3295, Oct. 01, 2021. doi: 10.1094/pdis-03-21-0518-pdn.

[3] N. E. Kafkas, Ed., "Recent Studies on Strawberries." IntechOpen, Jan. 04, 2023. doi: 10.5772/intechopen.98136.

[4] C. V. A. Lopes and G. S. C. de Albuquerque, "Agrotóxicos e seus impactos na saúde humana e ambiental: uma revisão sistemática," Saúde em Debate, vol. 42, no. 117. FapUNIFESP (SciELO), pp. 518–534, Jun. 2018. doi: 10.1590/0103-1104201811714.

[5] A. Patel, W. S. Lee, N. A. Peres, and C. W. Fraisse, "Strawberry plant wetness detection using computer vision and deep learning," Smart Agricultural Technology, vol. 1. Elsevier BV, p. 100013, Dec. 2021. doi: 10.1016/j.atech.2021.100013.

[6] A. Picon, M. Seitz, A. Alvarez-Gila, P. Mohnke, A. Ortiz-Barredo, and J. Echazarra, "Crop conditional Convolutional Neural Networks for massive multi-crop plant disease classification over cell phone acquired images taken on real field conditions," Computers and Electronics in Agriculture, vol. 167. Elsevier BV, p. 105093, Dec. 2019. doi: 10.1016/j.compag.2019.105093.

[7] A. Elhassouny and F. Smarandache, "Smart mobile application to recognize tomato leaf diseases using Convolutional Neural Networks," 2019 International Conference of Computer Science and Renewable Energies (ICCSRE). IEEE, Jul. 2019. doi: 10.1109/iccsre.2019.8807737.

[8] G. Jocher et al., ultralytics/yolov5: v6.2 - YOLOv5 Classification Models, Apple M1, Reproducibility, ClearML and Deci.ai integrations. Zenodo, 2022. doi: 10.5281/ZENODO.7002879.

[9] U. Afzaal, B. Bhattarai, Y. R. Pandeya, and J. Lee, "An Instance Segmentation Model for Strawberry Diseases Based on Mask R-CNN," Sensors, vol. 21, no. 19. MDPI AG, p. 6565, Sep. 30, 2021. doi: 10.3390/s21196565.

[10] B. Richey and M. V. Shirvaikar, "Deep learning based real-time detection of northern corn leaf blight crop disease using YoloV4," Real-Time Image Processing and Deep Learning 2021. SPIE, Apr. 12, 2021. doi: 10.1117/12.2587892.

[11] M. Cruz, S. Mafra, E. Teixeira, and F. Figueiredo, "Smart Strawberry Farming Using Edge Computing and IoT," Sensors, vol. 22, no. 15. MDPI AG, p. 5866, Aug. 05, 2022. doi: 10.3390/s22155866.