Taylor & Francis
Taylor & Francis Group

Journal of Decision Systems

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/tjds20

A scalable offline AI-based solution to assist the diseases and plague detection in agriculture

Matias Urbieta, Martin Urbieta, Mauro Pereyra, Tomas Laborde, Guillermo Villarreal & Mariana Del Pino

To cite this article: Matias Urbieta, Martin Urbieta, Mauro Pereyra, Tomas Laborde, Guillermo Villarreal & Mariana Del Pino (2023): A scalable offline AI-based solution to assist the diseases and plague detection in agriculture, Journal of Decision Systems, DOI: 10.1080/12460125.2023.2226381

To link to this article: https://doi.org/10.1080/12460125.2023.2226381

Published online: 22 Jun 2023.

JOURNAL OF DECISION SYSTEMS
https://doi.org/10.1080/12460125.2023.2226381

Taylor & Francis
Taylor & Francis Group

A scalable offline AI-based solution to assist the diseases and plague detection in agriculture

Matias Urbieta[a], Martin Urbieta[a], Mauro Pereyra[a], Tomas Laborde[a], Guillermo Villarreal[a] and Mariana Del Pino[b]

[a]Facultad de Informática, Universidad Nacional de La Plata, La Plata, Argentina; [b]Facultad de Agronomia, Universidad Nacional de La Plata, La Plata, Argentina

ABSTRACT

Early detection of diseases and pests is a key factor in eradicating or minimising the damage that these may cause. In this work, a comprehensive solution is presented that is based on the composition of existing cloud solutions and mobile tools to detect in-situ issues. The platform presented was used for the detection of powdery mildew and Cladosporium diseases in tomatoes. The results of using the approach to carry out this task were more than satisfactory since it managed to correctly detect the symptoms, having mAP of 0.41 in at least some of these symptoms. We analysed the performance of our dataset, on the one hand, and the combination of PlantDoc dataset, on the other hand. This shows that the platform can be used in the agriculture sector, as an additional tool for detecting diseases and pests in order to combat the problem and reduce its consequences.

ARTICLE HISTORY
Received 2 December 2022
Accepted 13 June 2023

KEYWORDS
Agriculture; Cloud; Machine-learning; Mobile; agriculture; tomato; powder mould; and cladosporium

1. Motivation

Food security refers to people's physical, social, and economic access to safe, nutritious, and sufficient food to meet their dietary needs and food preferences FAO-PESA Centroamerica 2011. Agricultural crop wastes and losses are so high that researchers and companies spend resources on studying solutions. The main goal is to avoid throwing away aliments that could be used to meet the unsatisfied food demand of a large part of the world's population. Diseases and pests in horticultural crops are part of the production losses since they cause damage to crops that reduce the yield, the quality of harvested products, or the total loss of them International Plant Protection Convention FAO 2017. It is conservatively estimated that diseases, insects, and weeds cause annual losses of between 31% and 42% of all crops produced worldwide. Losses tend to be lower in more developed countries and higher in developing countries, i.e. countries that need more food. It has been estimated that of the average 36.5% of total losses, 14.1% are caused by diseases, representing approximately $220 billion FAO 2019. In the case of small producers, who generate 80% of the world's food production FAO 2015, Instituto

CONTACT Matias Urbieta matias.urbieta@lifia.info.unlp.edu.ar Facultad de Informática, Universidad Nacional de La Plata, La Plata, Argentina

Nacional de Tecnología Agropecuaria 2017, the economic consequences can be devastating if they do not have sufficient means to counteract the situation.

Early detection of diseases and pests is a key factor in eradicating or minimising losses. However, it is not an easy task as a myriad of diseases affects crops and wild plants. On average, each crop can be affected by 100 or more diseases. In this type of infection, at first, the disease is localised in one or a few cells and is invisible, but quickly the reaction becomes generalised and the affected plant parts develop changes visible to the naked eye, which constitute the symptoms of the disease G.N. Agrios 2005.

On the other hand, the crops are located scattered on a large land extension making it hard and expensive to transport professionals who can diagnose crop issues in an efficient way.

Taking advantage of the fact that this problem generally presents visible symptoms in crops, Convolutional Neural Networks can be used to train models that enable its detection.

During some interviews with local farmers, we identified some challenges to monitoring vegetable production. Firstly, the employees' training is complicated as they are hired temporarily based on the season and sometimes they have basic or no training. Smartphone device adoption grows year after year and businesses are profiting from them to support operations. In this case, cultural work is not an exception and mobile devices can help with the in-situ task.

On the other hand, professional assistance is also difficult because of the long distances they need to travel to reach a given location. Travel hours can be from a few hours to a few dozen and the corresponding travel expenses must be paid, which increases the cost of professional services provided, reducing the frequency of attendance at the production site.

Finally, internet access is limited in some places in Argentina when trying to get online cloud solutions. It is the eighth-largest country in the world so mobile antennas do not have full land coverage. However, taking into account the technological advances related to mobile hardware, it is possible to run high-performance computing applications on mobile devices.

Convolutional Neural Networks are one of the techniques used nowadays for object detection tasks in images, largely due to their high accuracy in performing this task Krizhevsky et al., 2012. There is a large literature on them and different free tools are available, including TensorFlow and TensorFlow Lite, which allow the training and use of Machine Learning models on devices with limited resources, such as smartphones affordable by any producer.

Although diseases and pest detection have already been studied for decades, the emergence of object recognition using neural networks has become the leading approach in many fields, with a vital role in the early detection and classification of plant pests and diseases Hasan et al., 2020. We propose a scalable approach that enables the real-time detection of diseases and pests in crops through the camera of a smartphone using Convolutional Neural Networks to accomplish the classification offline.

Many researchers have reported different diseases and plague datasets Amara et al. 2017, Barbedo et al., 2018, Hughes and Salathe 2015, LeCun et al., 1989, Prajapati et al., 2017, Singh et al., 2019, Thapa et al., 2020, Wiesner-Hanks et al., 2018, Wu et al., 2019, Huang and Chuang 2020. Unfortunately, none is able to document most of the diseases worldwide

available to train a model. This makes sense because they vary depending on the location. So, the dataset generation involves farmers allowing them to enrich datasets with a new vegetable, their diseases, and plagues. The most critical problem facing the detection of plant pests and diseases is the lack of meaningful datasets Liu and Wang 2021. On the other hand, the results vary if the dataset contains the crops whose leaves have been separated from the plants for presentation presented with a background or were taken in the field Hasan et al., 2020.

It is not to our knowledge the existence of datasets that include tomato powder mould and Cladosporium. These pests and diseases had been selected as recognition targets for helping local tomato growers to detect and treat them during the APP trial phase.

In this work, we present a solution that combines cloud solutions for training neural networks and mobile devices to run Convolutional Neural Networks in situ. It combines the existing free cloud solutions for training and a mobile device app. It is intended to be easy to configure, extend and modify, allowing anyone to make use of it for any disease or pest detection. The mobile app is generic so it can be used to assess the quality of crops, for the recognition of species and weeds, and for substrate problems such as saltpetre, among others. The mobile app does not pretend to replace professional services but it aims at providing a preliminary diagnosis that could be used to warn about a possible issue. In the context of under-development countries, many of the crops are driven by unqualified farmer operators with low economic resources and do not count economic resources to contract specialists.

This approach can extrapolate to other crops, identify those most common diseases and propose their prospective treatment with specific pesticides.

To be able to identify more than one disease in an image, with the highest possible speed and the least consumption of computational resources, it has been selected to use SSD Mobilenet v2 Chiu et al., 2020. Therefore, other techniques that allow image segmentation, such as Mask R-CNN He et al., 2017, were not selected for our solution, because identification by bounding boxes was considered sufficient. We picked SSD Mobilenet v2 network because its well-known model among other models like EfficientNet-Lite 2011 or newest Mobilenet versions. The reader should note we are not pursuing to compare and report best performing engines and models. Instead, we aim at introducing an extensible approach to support the plague and diseases

identification that can be extensible to support alternative engines (i.e. MNN Jiang et al., 2020 and models.) Our contributions include:

(1) An approach that enables new strategies for decision-making in agriculture using Machine Learning.
(2) A platform that trains a neural network through the use of different Cloud services and runs the trained model on a mobile device to allow the detection of objects in real time and without the need to be connected to the internet (except for downloading the model and its successive updates).
(3) A dataset for tomato diseases and pests that includes powdery mildew and Cladosporium.

In section 2, we will discuss related work. In section 3, we will present the rationale for this work. The solution will be presented in section 5 and finally, in section 6, we will present conclusions and future work.

2. Related work

Machine Learning, and in particular Convolutional Neural Networks, has been used for several years to perform different tasks in the field of Agriculture, among which is the detection of crop diseases through the analysis of their symptoms. There is a large amount and variety of research on this, where different techniques and strategies are used to generate a model with high accuracy.

In general, research related to plants uses the Plant Village dataset as a source of images, which contains more than 50,000 images of healthy leaves and ill leaves with disease symptoms divided into 38 categories by species and disease. This dataset was used, for example, to identify two diseases in bananas with an accuracy of 98.6% using a LeNet LeCun et al., 1989, Amara et al., 2017; to identify 26 diseases in 14 crops with an accuracy of 99.35% Mohanty et al., 2016, using AlexNet Krizhevsky et al., 2012 and GoogleNet Szegedy et al., 2015 architectures; to detect black rot in apple by using Transfer Learning and VGG−16 Simonyan and Zisserman 2014 architecture with an accuracy of 90.4% accuracy, and

https://github.com/tensorflow/tpu/tree/master/models/official/efficientnet/lite

to detect diseases in the potato crop with an accuracy of 95%. In the same way, Singh et al., 2019 presents a dataset that does not include the diseases introduced in this work. This shows that there is no universal dataset documenting every disease on Earth, so a flexible approach for enriching the diseases database to train the neural network.

Fuentes et al., 2017 made use of the Faster R-CNN architecture Ren et al., 2017 with VGG−16 for training a tomato disease detector, which obtained an accuracy of 83% using the image magnification technique.

Ghoury et al., 2019 detect grape and grape leaves using Faster R-CNN Inception v2, with a classification accuracy of 78% to 99% with a processing time of 25–30 seconds per image. The performance was compared with a Single Shot Detector (SSD) Liu et al., 2016 MobileNet v1 Howard et al., 2017 which was unsuitable for real-time classifications due to a high percentage of misclassifications. Despite the poor performance shown in mobilenet v1, the Mobilenet-SSDv2 Chiu et al., 2020 retains the advantage of fast processing of the first version, but also improves the detection accuracy. These works focus on analysing two different models but our work focuses on implementing a solution for real-time image classification that runs offline on mobile devices.

Tahir et al., 2018 trained a model capable of detecting fungi with 94.8% accuracy using a proprietary dataset with 40,800 labelled images.

Ahmed and Reddy 2021 present a mobile system to detect plant leaf diseases using deep learning. The development uses CNN to classify 38 disease categories in 14 crop species, using 96,206 collected images to train, validate and test the model. The dataset source has been Kaggle, PlantVillage, and Google Web Scraper. This approach allows capturing or uploading an image to the phone, which uploads it

to the cloud server to detect the disease class by applying the CNN model. The accuracy achieved reaches 94% operation. This operation of prediction and displaying results took around 0.88 s, including the communication overheads. Our approach differs mainly in being an offline solution that allows operating in places without internet coverage and the dataset includes images of powdery mildew and Cladosporium obtained in the field.

Sarangdhar and Pawar 2017 presented a system for the detection and control of five cotton leaf diseases, which also performs soil quality monitoring. Once the disease detection is done, the name and its treatment are provided to farmers through a mobile application, which also provides different soil parameters, such as moisture and temperature, among others.

Although the aforementioned research has achieved good results in disease identification, it has certain limitations. The first limitation is the use of the Plant Village dataset as a source of images. As described in some papers, network training using this dataset generates models with poor accuracy when inferring real-world images. The reason for this is that Plant Village relies on images taken in controlled or laboratory environments, where real situations are not considered, such as the non-uniformity in the background of the images, the different resolutions of the cameras, the differences in illumination, the variety in the size and shapes of the symptoms and the possibility of the appearance of several leaves in the same image. Another limitation is that some of the investigations do not allow the detection of multiple symptoms and diseases on the same leaf, since they perform classification instead of object detection, and therefore take the leaf as a whole, instead of taking the different symptoms as distinct entities. Moreover, in the cited works, image magnification is not a strong point. In some of them, the images are neither prepared nor enhanced; in others, only conversion to black and white is performed, but in none of them a robust enhancement is implemented, which contemplates a wide variety of transformations that allow generating different conditions for the images. Finally, the analysed works were developed specifically to work on certain diseases and neural network architectures.

Unlike them, this work proposes an end-to-end platform that, among other features, allows training a model capable of detecting objects in general and is fully configurable, where image enhancement is part of the workflow and the choice of the architecture to be used will depend on the type of problem and user preferences. Finally, among the specific objectives of this work is the availability of a mobile tool that allows running the neural network from the device without requiring an internet connection.

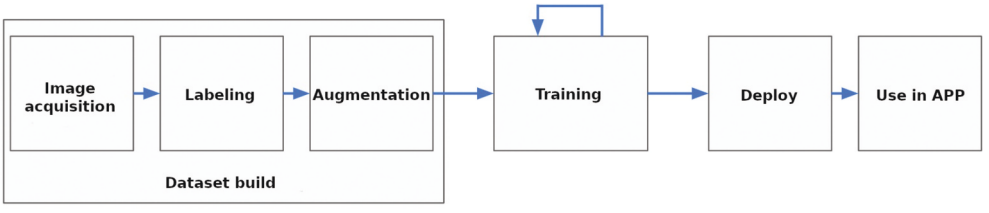

Figure 1. Activities contemplated in the platform.

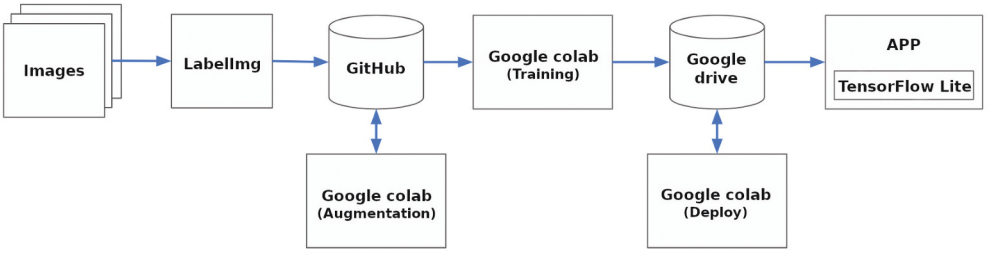

Figure 2. Platform components.

3. Design

With the aim of guaranteeing that the proposed platform covers the complete process of training and delivering the Machine Learning model using a mobile app, the following activities have been contemplated in the platform:

Figure 1 shows the four main activities: the image compilation for producing the training dataset, the training of the neural network, the deployment of the generated model and its delivery, and the use of the model by the mobile application. The dataset compilation includes the acquisition, labelling, and augmentation of the images to be used for training. Training is an iterative process during which the algorithm provides information and different metrics to the user regarding the performance of the model being trained. Based on this information, the user can choose to make adjustments to the network,

parameters, and input data, or even choose to use a different model, in case the results returned are not satisfactory. The model deployment consists of two activities. The first is the generation of the model from the files generated during training. The second activity is the storage of the model in a cloud storage service.

Finally, the use of the model by the application contemplates the synchronisation (downloading and updating) of the model on the smartphone and its use.

Storing a machine learning model in the cloud has some advantages: the size of the mobile application will be smaller during installation time since it does not include any file (model) in its source code and, in addition, in the case of wanting to update the model it is not necessary to generate a new version of the application with the updated model, since it can be downloaded from the Internet.

In order to fulfil the above-mentioned activities, the platform has the following components and their respective connections:

The workflow and data flow in the architecture showed in Figure 2 are as follows:

(1) First, the images to be used for training must be collected.
(2) Then, the images must be labelled using the *LabelImg* software, which will generate an .xml mapping file for each image. Once the images are labelled, they must be divided into two sets (training and testing). Both sets must be uploaded to a Github repository.
(3) A Colab document is going to be in charge of the augmentation of the images uploaded to Github. After that, it will generate the necessary files to perform the network training, which it will upload to Github. A second Colab document, using

the files generated in the previous point, will perform the network training. During the training, checkpoints will be generated, which will be uploaded to Google Drive. A third Colab document will obtain the most recent *checkpoint* from Google Drive and will generate the TFLite model, which will be available for download in the same storage service. The mobile application will download the model from Google Drive. Once downloaded, the application will be ready to start detecting objects in real-time through the phone's camera.

Next, we will describe step by step the different activities of the platform, from the assembly of the training dataset to the use of the model in the application. At each stage, we will explain the components involved and the exchange of information within the platform.

3.1. Building the training dataset

The first activity contemplates capturing the images, their labelling, and their augmentation.

3.1.1. Getting images

The first thing we must do is obtain the images that are going to be used for network training. As far as possible, they should be images with good definitions and where the objects of interest are clearly displayed. It is desirable not to include images that have zoom, noise, low light, excessive brightness, and so on, since the augmentation step will generate all these variations.

As far as possible, images that do not have the objects to be detected should also be included. The goal of adding this type of image is to reduce the number of false positives, a situation in which the model erroneously detects the objects. The variety of images to include will depend on several factors, such as the type of object to be detected, and the ideal is to include both images related to the context of the objects and images not related to it at all.

Before proceeding to tag the images, it is convenient to resize them to a single size, generally to the size of the smallest image, so that they take up less storage space. The resizing will also allow reducing the number of parameters and computations used during the training of the neural network. There are many free tools to resize images. There are available free tools for image resizing like the *Image Resizer for Windows* 2017 software (that allows us to select all the images and with a few clicks resize them to the chosen size), or scikit-image and Pillow which are Python libraries that run cross-platform.

https://www.bricelam.net/ImageResizer

3.1.2. Image tagging

The type of learning required by the convolutional network is supervised and therefore the learning uses labelled data.

When working with images and object detection, one of the ways we provide labelled data is through .xml mapping files. So, the image will serve as input and the.xml file as output.

For the generation of these files, we have used the software *LabelImg* 2019, a free, light, and easy-to-use image annotator that is available for Windows, Mac, and Linux.

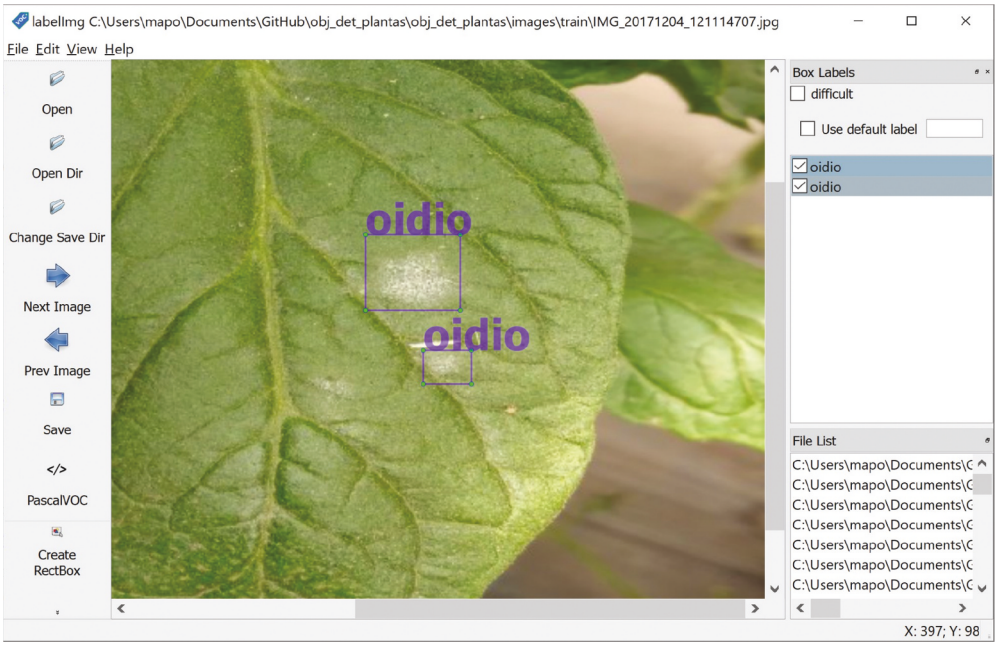

Figure 3. Labellmg main screen.

It should be clarified that this software is the only component of the platform that is not cloud, because the Web tools available at the time that could be used to carry out this work have errors when handling a large number of images or are paid, which does not meet our requirement of using only free services.

Using this tool is easy and intuitive. In Figure 3 you can see its main screen, in which two labelled objects appear (with the class 'oidio').

After labelling the images, they must be divided into two folders (training and test). The first will be used for training, and the second to periodically evaluate training performance, which will allow the algorithm to make the necessary adjustments to the network in order to improve predictions. The percentage of distribution between both folders is usually 80% for training and 20% for the test; however, the percentage is left to the programmer's choice, according to his preferences or needs.

After dividing the files, they must be stored in the cloud, for this Github 2015 will be used. This service provides free repositories (both public and private) with a limit of 100MB per file uploaded and a maximum of 1GB (recommended) per repository 2017, which is more than enough for our use.

3.1.2.1. Image augmentation. What we need to do next is expand the set of labelled images. To do this, we are going to use the augmentation technique, which allows us to generate new images from existing ones. This technique is really useful when we have few images or when they are difficult to label. Its importance lies in the fact that it allows for increasing the effectiveness of the model since a large number of extra images will be provided that contemplate different conditions or situations that were surely not

contemplated in the original images. It is also useful to avoid *overfitting*, that is, the memorisation by the model of the training images, which leads to low accuracy in detecting new images.

Augmentation generates images by performing transformations to the original image. It is possible, for example, to rotate, add zoom, change colours or lighting, add noise, etc.

Modifying an image means that we also have to update its mapping file since if, for example, we flip and rotate an image, the positions of the objects it contains will change.

We are going to carry out this augmentation process using the Google Collaboratory service Agrios, 2005 (Colab), which is a free Jupyter Notebook environment Krizhevsky et al., 2012.

Jupyter Notebook is an open-source web application that allows you to create and share documents that contain code. Each document is made up of cells, which can contain code or text, and their result (text, graphics, tables, results of operations) is displayed after each one of them. By supporting Python, it is widely used in Machine Learning tasks.

Google Colab, for its part, requires no configuration, runs completely in the cloud, and provides a K80 GPU, 12GB of RAM, and 12 hours of continuous use in its free version until the environment is restarted. In addition, it provides a

(1) https://github.com/tzutalin/labelImg
(2) https://www.github.com
(3) https://help.github.com/es/github/managing-large-files/what-is-my-disk-quota
(4) https://colab.research.google.com
(5) https://jupyter.org

filesystem, has pre-installed libraries ready to be used, and allows you to connect to services such as GitHub and Google Drive in a simple way. This approach is not restricted to Google Colab, you can use any other cloud service like AWS.

When generating each new image, its mapping file will also be created with the positions of the updated objects.

It is important to clarify that image augmentation will be applied over images having tags defined.

Once all the images have been created, the files *train.record*, *test.record* and *label map. pbtxt* will be generated, which are required by the neural network to perform the training. The first two files are in the *TFRecord* format, which is a record-oriented binary file format that allows efficient storage and processing of large data sets. The remaining file is a text file where the ID and name of the tagged objects are specified.

After generating the files, they will be uploaded to the */annotations* directory of the GitHub repository specified in the document parameters.

3.1.2.2. Training. The next activity of the platform is the training of the Machine Learning model. In order to perform the training, we provide a script. These resources are available at Ibrahim Hasan et al., 2020 that takes the images, and annotations from a GIT repository, and generates a set of checkpoints.

Certain parameters must first be configured in the document, such as the number of training stages and the address of the GitHub repository that contains the files generated by the previous activity. In addition, the pre-trained model to be used must be selected. By default, the platform selects the '*ssd mobilenet v2 quantised*' model since it is an optimised version to run on mobile phones and also works correctly with any problem involving simple objects.

To carry out the training it is necessary to make some modifications in the configuration file of the selected model to be reused. This file can be downloaded from the TensorFlow repository LeCun et al., 1989 and has all the configurations to be used during the training. At a minimum, the paths of the directories where the *train.record*, *test.record* and *label map.pbtxt* files are located must be modified, and the number of classes. The file provides several parameters (specific to TensorFlow), such as the feature extractor, the learning rate, and the *loss* function that will be used to perform the training.

Running the Colab document will link the file system to our Google Drive account Amara et al., 2017. This storage service was selected as it has a maximum capacity (shared with Gmail and Google Photos) of 15GB Singh et al., 2019 for its free version, which will allow us to work without problems.

Then, the pre-trained model to be used will be downloaded and the training will begin, which will be carried out using the free TensorFlow Hughes et al., 2015 library.

During the training, Colab will show relevant information, such as the *loss* and the metric *mAP*. The value of the *loss* will be displayed at the end of each training stage. A normal value for this indicator is between 0 and 1, and although in the early stages, the value is normally high, it will decrease as training progresses. If the value of the loss does not decrease, it may be due to a problem in the set of images. If the value increases rapidly, it may be due to the appearance of *overfitting* in the model. For a throughout report, it is available in the TensorBoard Prajapati et al., 2017 tool, which is included in the TensorFlow framework. This tool allows not only to observe of the *loss* and the *mAP*, but also a large number of different metrics and also the predictions made in the different training stages.

During the training, different checkpoints will be generated, which save the state (variables, operations, weights) that the neural network contains at a given moment.

The checkpoints will be uploaded to Google Drive automatically, which will allow the TFLite model to be generated and also to be able to resume the training in the event of an error.

3.1.2.3. Deploy from TFLite model. The mobile application uses the *TensorFlow Lite* Garcia Arnal Barbedo et al., 2018 library to perform object detection, which is designed to run models efficiently on devices with limited resources. Part of that efficiency comes from using the TFLite format.

Models generated by TensorFlow must be converted to TFLite format before *TensorFlow Lite* can make use of them. This conversion substantially reduces its size and introduces optimisations that do not affect its accuracy. It is also possible to further reduce the size of the model and increase its execution speed, at the cost of some slight penalties, such as its detection efficiency.

To carry out this activity, a final Colab document will be used under the name generate model.ipynb Wiesner-Hanks et al., 2018.

When executing the script, the filesystem will be linked to a cloud storage (i.e. Google Drive account) where the checkpoints generated by the previous activity are located. The checkpoint to be used will be the most recent.

(1) https://cololaborde.github.io/notebooks/train model.ipynb, ready to be executed in Google Colaboratory.
(2) https://github.com/tensorflow/models/tree/master/research/object detection/samples/configs 10. https://drive.google.com
(3) https://one.google.com/faq/storage
(4) https://www.tensorflow.org
(5) https://www.tensorflow.org/tensorboard
(6) https://www.tensorflow.org/lite
(7) https://cololaborde.github.io/notebooks/generate model.ipynb

To convert the model, two TensorFlow scripts will be used (*tflite convert* and *export tflite ssd graph*), which will be automatically downloaded by the mobile app as the model will be publicly available using a URL. The script '*tflite convert*' converts the model to TFLite format. This script operates with a file extension of '.pb', so the checkpoint must first be converted to that format.

The *.pb* extension refers to '*protobuf*', a type of file used by TensorFlow that contains the neural network graph definition and model weights.

When executing the script, the file '*tflite graph.pb*' will be generated, which will be used in the execution of the command '*tflite convert*'.

To allow the application to download a model and its successive updates using the same URL, it is necessary that the model *detect.tflite* is always available at the same location.

3.1.2.4. Use of the model in the mobile application. The last activity is the use of the object detector model through the mobile application.

An Android application was developed to run the trained model and its source code is available at Thapa et al., 2020.

The first time the app is open, it downloads the latest trained model from the cloud. This only happens once and no internet connection is required to use the trained model later in-situ.

The detected objects are marked with a rectangle that indicates the position, confidence in the prediction, and optionally the name of each of them. Each object class has a randomly generated colour, and it is possible to detect multiple instances of multiple classes at once. In addition to showing the objects in the camera image, they are listed below it, where the name, colour, and number of instances detected are indicated.

In terms of customisation, the app allows some UI adjustments, such as the thickness of the rectangle used to mark the objects, the minimum percentage of confidence used to filter the results of the predictions, and whether or not to show the name of the detected objects. A minimum confidence percentage of 80% indicates that if predictions with 30% or 40% confidence are returned, they will be rejected.

4. Detecting diseases and pests in crops

In this section, we will describe how the platform was used to train a model capable of detecting symptoms of two tomato diseases, demonstrating that it is possible to use it for the detection of symptoms in any type of crop.

The image gathering was performed by professors from the Faculty of Agronomy of the National University of La Plata, whom we thank since they provided us with a large number of tomato images with symptoms of two of their diseases: Cladosporium and powdery mildew. This could be extended to more vegetables and their diseases.

Tomato leaf mould Cladosporium is a disease that develops in humidity conditions above 70% and temperatures between 5 and 25 Cº. The symptoms appear on the underside of the leaves, first as coloured spots yellowish green on older leaves. These correlate with yellow chlorotic spots on the upper surface. Later, the spots enlarge and coalesce, turn brown to black, may spread to the remaining younger leaves, and may cause defoliation G.N. Agrios 2005.

Dry environmental conditions, without rain, but with high relative humidity, favour tomato infection due to powdery mildew. They can infect leaves at extreme temperatures (10 to 32ºC), but the optimal temperature is 27ºC. Powdery mildews, although common and causing severe disease in cool or warm humid areas, are even more common and severe in warm and dry weather G.N. Agrios 2005, Rossini et al., 2010. A white powdery mould develops on the upper surface of older leaves, which then become chlorotic and finally necrotic. They can also affect stems and other organs. Powdery mildew is most common on upper leaves but also affects lower leaves, young shoots and stems, buds, flowers, and young fruit peduncles.

The number of images used was 146, 49 of which responded to Cladosporium, 43 to powdery mildew, and the remaining 54 showed no symptoms of either diseases. These latter images were added to reduce possible overfitting and false positives. These

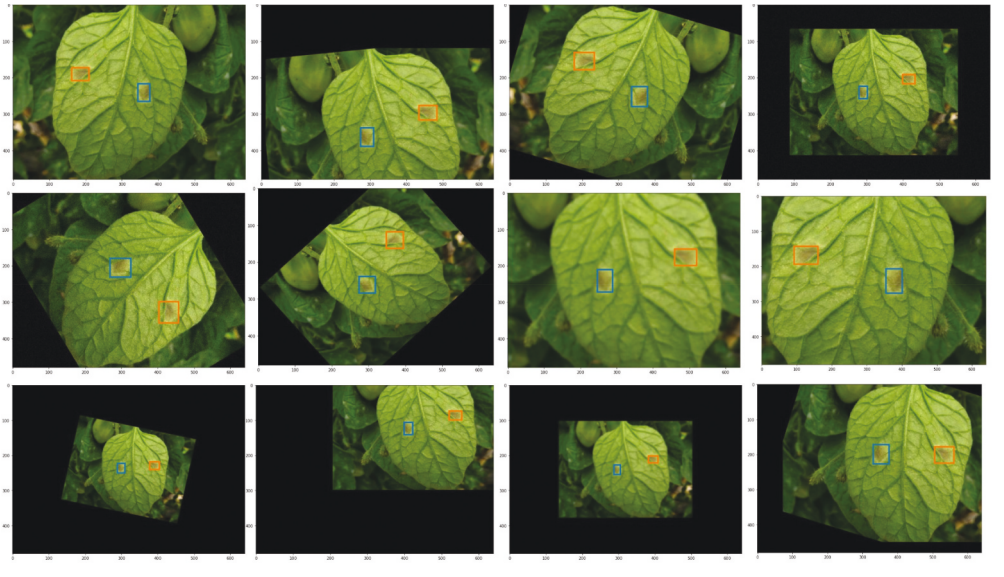

Figure 4. Transformations performed on an image.

included images of healthy plants, images that had symptoms of other diseases, and some random images downloaded from the internet.

After labelling the images with LabelImg, they were divided into the *training* and *test* sets, with a distribution of 70 and 30%, respectively, and uploaded to GitHub.

Images were then augmented using the respective Colab document. As we had only 146 images in total, it was necessary to generate a large number of extra images, to allow the neural network to learn better. We chose to generate 30 images for each image that had labelled symptoms, resulting in a total of 2,906 images for training.

In Figure 4, you can see some of the transformations applied to a particular image.

After performing the augmentation, the Colab document generated the files *train.record*, *test.record*, and *label map.pbtxt*, which it uploaded to the Github repository.

Then, we continued with the Colab document in charge of training the neural network.

After finishing the training, the last Colab document was used to generate the *TFLite* model. With the model uploaded to Google Drive, he downloaded it from the mobile app.

In the images of Figure 5 you can see the different detections made by the application.

Tensorflow offers detection models pre-trained on the COCO 2017 dataset Xiaoping et al., 2019. Additional model's weights are available at TensorFlow 2 Detection Model

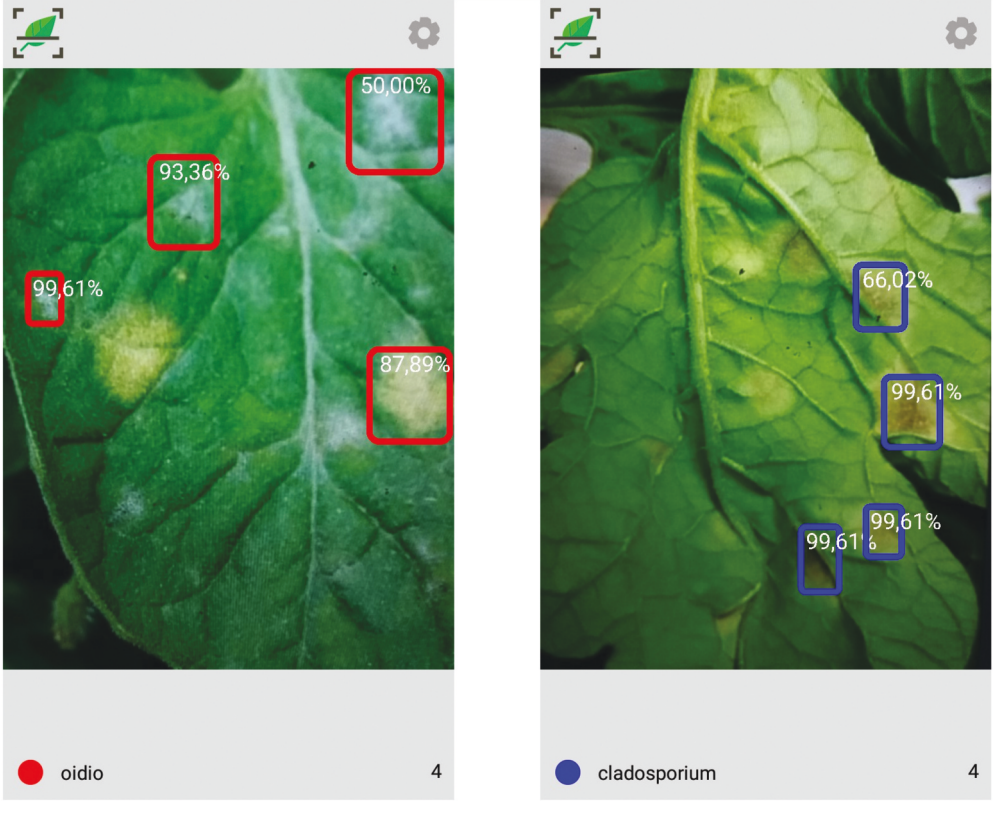

Figure 5. Predictions made by the app.

Table 1. Comparison of EfficientDet, Faster RCNN resnet v1, and SSD with mobilenet v2 based on TomAR dataset and applying 80,000 steps.

TensorFlow 2 Detection Model	EfficientNet B0 coco17 512z512	Faster RCNN Resnet50 v1 640×640	SSD with Mobilenet v2 300 × 300
Dataset		TomAR	
Steps	82300	80000	80000
Total Loss	0.477	0.363	1.125
PerformanceByCategory/mAP/cladosporium	0.578	0.561	0.416
PerformanceByCategory/mAP/oidio:	0.659	0.687	0.486
Average Precision (AP) @[IoU = 0.50:0.95 — area= all – maxDets = 100]	0.618	0.624	0.451
Average Precision (AP) @[IoU = 0.50 — area= all – maxDets = 100]	0.901	0.902	0.828
Average Precision (AP) @[IoU = 0.75 — area= all – maxDets = 100]	0.726	0.721	0.436
Average Precision (AP) @[IoU = 0.50:0.95 — area= small – maxDets = 100]	0.496	0.591	0.346
Average Precision (AP) @[IoU = 0.50:0.95 — area=medium – maxDets = 100]	0.653	0.64	0.477
Average Precision (AP) @[IoU = 0.50:0.95 — area= large – maxDets = 100]	0.589	0.597	0.49
Average Recall (AR) @[IoU = 0.50:0.95 — area= all – maxDets = 1]	0.110	0.11	0.089
Average Recall (AR) @[IoU = 0.50:0.95 — area= all – maxDets = 10]	0.554	0.568	0.447
Average Recall (AR) @[IoU = 0.50:0.95 — area= all – maxDets = 100]	0.684	0.699	0.55
Average Recall (AR) @[IoU = 0.50:0.95 — area= small – maxDets = 100]	0.573	0.655	0.405
Average Recall (AR) @[IoU = 0.50:0.95 — area=medium – maxDets = 100]	0.710	0.711	0.574
Average Recall (AR) @[IoU = 0.50:0.95 — area= large – maxDets = 100]	0.687	0.707	0.61

Zoo repository. In order to illustrate our approach, we selected EfficientDet D0 512 × 512, Faster RCNN resnet50 V1 640 × 640, and SDD with Mobilenet v2 pre-trained models for training and benchmark diseases and plague detection. We evaluated both models with 80,000 steps and measured mAP and lost metrics using coco detection metrics. When using PASCAL VOC, we got poor mAP but visual inspection of prediction showed good results. Because of this inconsistency, we decided to use COCO metrics which reported a better fit for visual inspection of the detection.

(1) https://github.com/cololaborde/app obj detection
(2) https://github.com/tensorflow/models/blob/master/research/object detection/g3doc/tf2 detection zoo.md

In practice, it is possible to combine different datasets to support a wider set of diseases and plagues. Therefore, we aggregate two different datasets in order to show how to profit from existing datasets. In this case, we combined the TomAR and The PlantDoc dataset. The latter provides healthy tomato images and eight new tomato diseases. In PlantDoc, the leaves are tagged and not pested, unlike TomAR. PlantDoc presents images within a size range from 6000 × 4000to 130 × 69. TomAR dataset has all images sized 640 × 480.

Table 3 shows the frequency of the PlantDoc dataset classes and our classes. In the first place, it is observed that we present two pests not previously reported by PlantDoc. In the second place, the class powdery mildew (oidio) and Cladosporium are positioned 20 and 28, respectively, out of a total of 31 positions.

Table 2. EfficientNet and mix dataset TomAr and PlantDoc (tomato).

TensorFlow 2 Detection Model	EfficientNetB0 coco17
Dataset	TomAR+PlantDoc (tomato)
Steps	81000
Total Loss	0.5699
Average Precision (AP) @[IoU=0.50:0.95 — area= all — maxDets=100]	0.219
Average Precision (AP) @[IoU=0.50 — area= all — maxDets=100]	0.417
Average Precision (AP) @[IoU=0.75 — area= all — maxDets=100]	0.197
Average Precision (AP) @[IoU=0.50:0.95 — area= small — maxDets=100]	0.16
Average Precision (AP) @[IoU=0.50:0.95 — area=medium — maxDets=100]	0.23
Average Precision (AP) @[IoU=0.50:0.95 — area= large — maxDets=100]	0.279
Average Recall (AR) @[IoU=0.50:0.95 — area= all — maxDets= 1]	0.189
Average Recall (AR) @[IoU=0.50:0.95 — area= all — maxDets= 10]	0.399
Average Recall (AR) @[IoU=0.50:0.95 — area= all — maxDets=100]	0.456
Average Recall (AR) @[IoU=0.50:0.95 — area= small — maxDets=100]	0.379
Average Recall (AR) @[IoU=0.50:0.95 — area=medium — maxDets=100]	0.461
Average Recall (AR) @[IoU=0.50:0.95 — area= large — maxDets=100]	0.583
PerformanceByCategory/mAP/Tomato Early blight leaf	0.145
PerformanceByCategory/mAP/Tomato Septoria leaf spot	0.351
PerformanceByCategory/mAP/Tomato leaf	0.239
PerformanceByCategory/mAP/Tomato leaf bacterial spot	0.151
PerformanceByCategory/mAP/Tomato leaf late blight	0.400
PerformanceByCategory/mAP/Tomato leaf mosaic virus	0.231
PerformanceByCategory/mAP/Tomato leaf yellow virus	0.157
PerformanceByCategory/mAP/Tomato mold leaf	0.236
PerformanceByCategory/mAP/cladosporium	0.286
PerformanceByCategory/mAP/oidio	0.206

Table 1 shows the training with EfficientDet, Faster RCNN, and SDD with MovileNet v2 on the TomAR dataset, with 80,000 steps. For the EfficientDet model, the mAP performance per category is mAP = 0.659 for powdery mildew, mAP = 0.578 for Cladosporium and the total loss value is 0.477. For Faster RCNN, the mAP performance by category was mAP = 0.687 for powdery mildew, mAP = 0.561 for Cladosporium and the total loss value is 0.363. Finally for the SDD with Mobilenet v2 model, the mAP performance per category is mAP = 0.486 for powdery mildew, mAP = 0.416 for Cladosporium and the total loss value is 1.125.

Considering that the available datasets do not include the classes introduced in this work, it was not possible to perform a performance comparison for the same pest using two datasets. On the other hand, the mobile models require less computing capacity, and therefore, the performance is lower than the models without this hardware restriction.

Our approach can profit from existing datasets to enrich the diseases and pest detection. So, the Table 2 presents the results of training the EfficientDet model with the TomAR dataset together with the tomato images subset from the PlantDoc dataset. The mAP performance per category is mAP = 0.206 for powdery mildew, mAP = 0.286 for Cladosporium and the total loss value is 0.569.

The performance values per category mAP obtained from the EfficientDet model trained with the mixed TomAR and PlantDoc dataset (only tomatoes), were lower than the SDD with Movilenet v2 model trained only with TomAR, both with 300×300 images.

Table 3. PlantDoc and TomAR datasets classes and frequency.

Nr	Class	Frequency	Dataset
1	Blueberry leaf	849	PlantDoc
2	Tomato leaf yellow virus	829	PlantDoc
3	Peach leaf	620	PlantDoc
4	Raspberry leaf	556	PlantDoc
5	Strawberry leaf	492	PlantDoc
6	Tomato Septoria leaf spot	436	PlantDoc
7	Tomato leaf	396	PlantDoc
8	Corn leaf blight	372	PlantDoc
9	Potato leaf early blight	333	PlantDoc
10	Bell pepper leaf	323	PlantDoc
11	Tomato mold leaf	293	PlantDoc
12	Tomato leaf bacterial spot	280	PlantDoc
13	Soyabean leaf	266	PlantDoc
14	Bell pepper leaf spot	264	PlantDoc
15	Tomato leaf mosaic virus	261	PlantDoc
16	Squash Powdery mildew leaf	257	PlantDoc
17	Potato leaf late blight	250	PlantDoc
18	Apple leaf	247	PlantDoc
19	Cherry leaf	240	PlantDoc
20	oidio	226	TomAR
21	Tomato leaf late blight	221	PlantDoc
22	grape leaf	220	PlantDoc
23	Tomato Early blight leaf	214	PlantDoc
24	Apple rust leaf	179	PlantDoc
25	Apple Scab Leaf	171	PlantDoc
26	grape leaf black rot	133	PlantDoc
27	Corn rust leaf	127	PlantDoc
28	cladosporium	90	TomAR
29	Corn Gray leaf spot	79	PlantDoc
30	Potato leaf	11	PlantDoc
31	Tomato two spotted spider mites leaf	2	PlantDoc

5. Conclusions

This research work does not intend to delve into the field of Machine Learning since there is extensive literature on it; on the contrary, it aims to leverage a comprehensive solution by composing existing solutions available in the cloud.

The presented approach was used for the detection of Powdery Mildew and Cladosporium diseases in tomatoes. The results of using the approach to carry out this task were more than satisfactory since it managed to correctly detect the symptoms, having mAP of 0.41 in at least some of these symptoms. This shows that the platform can be used in the agricultural sector, as an additional tool for the detection of diseases and pests to combat the problem and reduce its consequences.

Although the platform was conceived for the aforementioned goals, from the first moment it was kept in mind to generate a comprehensive, generic, configurable, free, and easy-to-use solution, so that it can be applied to any domain of interest and can boost any other similar goal.

This work differs from similar investigations of symptom detection using image augmentation. In general, research related to plants uses the Plant Village dataset Hughes and Salathe 2015 which is a large dataset that has more than 50,000 images. The main challenge with training models using this dataset is that the images it contains are images obtained under controlled conditions, which means that the neural network is trained with 'perfect' images, instead of with 'real-world' images. This generates that when the model must infer real images, its precision can be greatly impaired.

To our knowledge, there is no other similar solution to the one proposed in this work and for this reason, we want to dedicate a few lines to the importance of making technology accessible so that everyone can learn about it and use it.

The approach and its assets are intended to be used as a "step by step" and for this reason, it was "separated" into simple activities connected, to guide through the whole process that it contemplates to those who want to learn about this subject.

Disclosure statement

No potential conflict of interest was reported by the authors.

ORCID

Matias Urbieta http://orcid.org/0000-0002-4508-1209

References

Abdelmoamen Ahmed, A., & Harshavardhan Reddy, G. (2021). A mobile-based system for detecting plant leaf diseases using deep learning. *AgriEngineering*, *3*(3), 478–493. https://doi.org/10.3390/agriengineering3030032

Adhao, A.S. and Pawar, V.R. Machine learning regression technique for cotton leaf disease detection and controlling using iot. *2017 International conference of Electronics, Communication and Aerospace Technology (ICECA)*, 2:449–454, 2017.

Agrios, G.N. (2005). *Plant Pathology*. Elsevier Science.

Amara, J., Bouaziz, B., & Algergawy, A. (2017). A deep learning-based approach for banana leaf diseases classification. *BTW*.

Chiu, Y.-C., Tsai, C.-Y., Ruan, M.-D., Shen, G.-Y., and Lee, T.-T. Mobilenet-ssdv2: An improved object detection model for embedded systems. In *2020 International conference on system science and engineering (ICSSE)*, pages 1–5. IEEE, 2020.

Cladosporium. https://www.syngenta.es/cultivos/tomate/enfermedades/cladosporiosis. [Online; accessed 20-July-2020].

FAO. *El estado mundial de la agricultura y la alimentacio´n*. 2015.

FAO. Ano internacional de la sanidad vegetal. Roma, Italia, 2019. (CA6992ES/1/11.19).˜

FAO-PESA Centroamerica.Seguridad alimentaria nutricional, conceptos basicos.http://www.fao.org/3/a-at772s.pdf, 2011. [Online; accessed 20-July-2020].

Fuentes, A., Yoon, S., Kim, S.C., & Park, D.S. A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition. (2017). *Sensors*, *17*(9), 2022. *Basel, Switzerland*. https://doi.org/10.3390/s17092022

Garcia Arnal Barbedo, J., Vieira Koenigkan, L., Almeida Halfeld-Vieira, B., Veras Costa, R., Lima Nechet, K., Vieira Godoy, C., Lobo Junior, M., Rodrigues Alves Patricio, F., Talamini, V., Gonzaga Chitarra, L., Alves Santos Oliveira, S., Nakasone Ishida, A.K., Cunha Fernandes, J.M., Teixeira Santos, T., Rossi Cavalcanti, F., Terao, D., & Angelotti, F. (2018). Annotated plant pathology databases for image-based detection and recognition of diseases. *IEEE Latin America Transactions*, *16*(6), 1749–1757. https://doi.org/10.1109/TLA.2018.8444395

Ghoury, S., Sungur, C., and Durdu, A. Real-time diseases detection of grape and grape leaves using faster r-cnn and ssd mobilenet architectures. In *International conference on advanced technologies, computer engineering and science (ICATCES 2019)*, pages 39–44, 2019.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Hughes, D., & Salathe, M., et al. (2015). An open access repository of images on plant health to enable the development of mobile disease´ diagnostics. *arXiv Preprint arXiv: 151108060*.

Ibrahim Hasan, R., Mohd Yusuf, S., & Alzubaidi, L. (2020). Review of the state of the art of deep learning for plant diseases: A broad analysis and discussion. *Plants*, *9*(10), 1302. https://doi.org/10.3390/plants9101302

Instituto Nacional de Tecnolog´ıa Agropecuaria. La agricultura familiar produce casi el 80 por ciento de los alimentos. https://inta.gob.ar/noticias/la-agricultura-familiar-produce-casi-el-80-por-ciento-de-los-alimentos, 2017. [Online; accessed 20-July-2020].

International Plant Protection Convention FAO. Plant health and food security. http://www.fao.org/3/a-i7829e.pdf, 2017. [Online; accessed 20-July-2020].

Jiang, X., Wang, H., Chen, Y., Ziqi, W., Wang, L., Zou, B., Yang, Y., Cui, Z., Cai, Y., Tianhang, Y., Chengfei, L., & Zhihua, W. (2020). Mnn: A universal and efficient inference engine. , .

Kaiming, H., Gkioxari, G., Dollar, P., and Girshick, R. Mask r-cnn. In´ *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

Krizhevsky, A., Sutskever, I., & Hinton, G.E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 1097–1105.

LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., & Jackel, L.D. (1989, December). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, *1*(4), 541–551. https://doi.org/10.1162/neco.1989.1.4.541

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Cheng-Yang, F., and Alexander, C.B. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer.

Liu, J., & Wang, X. (2021). Plant diseases and pests detection based on deep learning: A review. *Plant Methods*, *17*(1), 1–18. https://doi.org/10.1186/s13007-021-00722-9

ML Huang and TC Chuang. (2020). A database of eight common tomato pest images. *Mendeley Data*.

Mohanty, S., Hughes, D., & Salathe, M. (2016). Using deep learning for image-based plant disease detection. *Frontiers in Plant Science*, *7*, 04. https://doi.org/10.3389/fpls.2016.01419

Prajapati, H., Shah, J., & Dabhi, V. (2017 07). Detection and classification of rice plant diseases. *Intelligent Decision Technologies*, *11*(3), 357–373. https://doi.org/10.3233/IDT-170301

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031

Rossini, M., Azar, G., Iglesias, N., Giayetto, A., Azpilicueta, C., Gonzalez, M., Ohaco, P., & Ruiz, C. (2010). *Enfermedades de mayor importancia de los principales cultivos hort´ıcolas*. Ediciones INTA.

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. https://doi.org/10.48550/arXiv.1409.1556

Singh, D., Jain, N., Jain, P., Kayal, P., Kumawat, S., & Batra, N. (abs/1911.10317, 2019). Plantdoc: A dataset for visual plant disease detection. *CoRr*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

Thapa, R., Snavely, N., Belongie, S., & Khan, A. (2020). The plant pathology challenge 2020 data set to classify foliar disease of apples. *Applications in Plant Sciences*, *8*(9). https://doi.org/10.1002/aps3.11390

Waseem Tahir, M., Abbas Zaidi, N., Akhtar Rao, A., Blank, R., Vellekoop, M.J., & Lang, W. (2018). A fungus spores dataset and a convolutional neural network based approach for fungus detection. *IEEE Transactions on NanoBioscience*, *17*(3), 281–290. https://doi.org/10.1109/TNB.2018.2839585

Wiesner-Hanks, T., Stewart, E.L., Kaczmar, N., DeChant, C., Wu, H., Nelson, R.J., Lipson, H., & Gore, M.A. (2018). Image set for deep learning: Field images of maize annotated with disease symptoms. *BMC Research Notes*, *11*(1), 1–3. https://doi.org/10.1186/s13104-018-3548-6

Xiaoping, W., Zhan, C., Lai, Y.-K., Cheng, M.-M., and Yang, J. Ip102: A large-scale benchmark dataset for insect pest recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8787–8796, 2019.