# Automatic detection of diseases in horticultural crops

José Antonio Brenes Carranza, Ricardo Zamora Mennigke
July 1, 2020

**Abstract**

Crop diseases are a major treat to productive sector and food security. In Costa Rica, but also around the world, farmers suffer the presence of diseases resulting in severe crops yield loss. Diseases detection is a hard task, usually farmers do not have all the knowledge required to deal with diseases. In recent years, there has been increased efforts to implement computer vision approaches in this field. The increased global use of smartphones together with deep learning and machine learning methods have intended to provide a supplementary mobile assisted plant disease diagnostic. In this research, we compare convolution neural networks (CNN) by analyzing the public dataset of PlantVillage project. That data set consist in healthy and diseased plant leaf images. By implementing CNN, we achieved high accuracy levels of 97.83% for a pre-trained densenet-161 model. We also created a custom model, with which we achieved an accuracy of 92.06%. We also register the execution time of the learning process and the size of resulting models. As a result, we note that custom model was significantly faster and smaller than pre-trained model. We concluded that for the creation of a mobile-based disease detection solution, it could be better to use a custom CNN instead of use the pre-trained option.

**Keywords:** convolutional neural network, image classification, plant disease detection, deep learning

## 1 Introduction

Currently, agriculture is one of the most important economic activities in Costa Rica. According with the 30th Agricultural Statistical Bulletin [1], during 2019, the agricultural sector contributed to the country's economy in a 4.2% of the GDP (Gross Domestic Product). Due to the current economic dis-acceleration, it is very important to help farmers to optimize their production by improving the productive processes. On a global perspective it has been a major concern to produce enough food for more then 7 billion people. According to the Food and Agriculture Organization (FAO) of the United Nations (UN) [2], due to damage caused by plant pests, insects, and diseases, every year global crop yields are reduced in about 20% and 40%.

Tai, Martin, and Heald [3] pointed out that today, agricultural sector is facing big challenges due to climate change such as pollution, reduced water availability, and presence of pests and diseases. The introduction of modern technologies has helped to increase food production and reduce those threats. According to the Management report of the Agricultural and Rural Sector in Costa Rica [4], during the quadrennium 2014-2018, there were many efforts focused on helping farmers to deal with problems that affect directly the production, through mitigation and adaptation strategies.

Ramírez and Nienhuis [5] indicate that horticultural crops are very affected by plagues in Costa Rica. The timely disease detection keeps being a major issue for farmers. That situation forces them to use agrochemicals causing, at the same time, an impact to the environment. In order to avoid the increasing environmental damage, Abarca [6] recommends to join efforts between different areas of knowledge to help farmers to improve early detection and treatment of plagues in crops.

Singh, Ganapathysubramanian, and Sarkar [7] mention that recently, deep learning (a sub-field of machine learning) has been having an increased use. Its objective is to incorporate models that analyze large sets of heterogeneous data and estimate reliable predictions, even on complex and uncertain situations. Plant science, specially with applications in farm and sustainability, are having increasing studies applying those methods. Mohanty, Hughes, and Salathé [8] argue that the increased use of smartphones, along with recent advances in computer vision, has made it possible to create mobile technology assisted disease diagnosis through deep learning.

In this research we compare the use of pre-trained CNN models and custom CNN models to create disease detection solutions. Regarding the future implementation of mobile-based solutions, where the computational resources are very limited, we consider helpful to know which model is better to use in terms of accuracy, learning execution time and size of solution.

## 2 Materials and Methods

Computer vision, and in particular machine learning approaches, have made special advances in the last few years. There has been a widespread use of numerous visualization-related problems specially for object classification. Rawat and Wang [9], discuss that Convolutional Neural Network (CNN) are gaining renown in the field. By implementing that technique, it is possible to train models and achieve a very precise image classification.

### 2.1 Data sources

To create a highly precise and accurately solution, it is needed to use a very well defined and verified data set. In the case of plant disease detection, it is required to use a data set that have images from affected and healthy plants. In out case, for the construction of the classification tool, we propose to use the Plant Village project data set [10]. The Plant Village is an effort made by Penn State University [11] and it consists of 54303 images of healthy and unhealthy leaves of horticultural crops. The images are divided in 38 categories according to species and diseases.

The Plant Village data set structure and number of samples for each crop type is shown in table (1):

Table 1: Plant Village data set structure and number of samples [12]

| **Crop** | **# of classes** | **# of samples** |
|---|---|---|
| Apple | 4 | 3171 |
| Blueberry | 1 | 1502 |
| Cherry | 1 | 1906 |
| Corn | 2 | 3852 |
| Grape | 4 | 4062 |
| Orange | 4 | 5507 |
| Peach | 1 | 2657 |
| Pepper | 2 | 2475 |
| Potato | 3 | 2152 |
| Raspberry | 1 | 371 |
| Soybean | 1 | 5090 |
| Squash | 1 | 1835 |
| Strawberry | 2 | 1565 |
| Tomato | 10 | 18160 |

Several studies using deep learning methods for disease detection have implemented the Plant Village data set [13]. In those studies, the models performance have been measured through the ability to discern between diseases and healthy samples. It is important to point out that leaf affectation vary from one disease to another, and it can be very similar for diseases in the same crop. For that reason it is important to train models enough in order to obtain good results.

For this research, we decided to work with tomato samples only. We think that it is useful to train several models, one for each crop, in order to create solution for disease detection. Solution like the one presented in [14] use the whole data set. We consider that this option can slow down the learning process because the model have to process a bigger data set (54303 samples) and deal with 38 classes. In our case, by using only tomato samples, we reduce the data set size to 18160 samples and the number of classes to only ten. It is crucial to keep in mind that farmers always know which crop they are producing. For that reason it is not necessary to create a model that deal with all crops and diseases at a time. Instead, we can create several models, one for each crop, and offer the user the possibility to select the crop (the model).

In table (2) we show the detail about the classes for tomato crop (healthy and diseased) taking into account in this research.

Table 2: Analyzed tomato diseases and number of samples for each class

| **Classes for tomato crop** | **# of samples** |
|---|---|
| Bacterial spot | 2127 |
| Early blight | 1000 |
| Healthy | 1591 |
| Late blight | 1909 |
| Leaf mold | 952 |
| Septoria leaf spot | 1771 |
| Spider mites two spotted spider mite | 1676 |
| Target spot | 1404 |
| Mosaic virus | 373 |
| Yellow leaf curl virus | 5357 |

## 2.2 Image classification

The main goal in this research is to detect a plant leaf in an image, and then to classify it according to the disease present in the leaf. In some cases, the leaf was not affected by a disease, so, in those cases the solution have to classify the leaves as healthy.

We proposed to carry out a comparison analysis between the use of a pre-trained model and a custom created model. To do that we explored the distinct pre-trained models available to use with *pytorch* package and python programming language. As it was pointed out in [15], pre-trained models enable the researcher to get good results when classify images by using a small to medium size data set.

We selected the densenet-161 due to, between distinct pre-trained models, it has one of the lowest scores for errors top-1 (22.35) and top-5 (6.20) [1].

To compare the results obtained by using densenet-161, we create a custom CNN model based on the recipes stated in [16]. To summarize, we run the following process. We started by extracting from Plant Village data set only the samples corresponding to tomato crop. Next, we randomly divided the data set in three parts: 70% of samples for training, 15% of samples for test and 15% of samples for validation.

Then, we fitted the selected pre-trained model by running the following configuration:

- Input image size: 224x224
- Batch size: 48
- # of epochs: 100
- Optimizer: Adam optimizer
- Initial learning rate: 0.001
- Loss function: NLLLoss
- Activation function: LogSoftmax

After that, we used the test set to predict responses for generating observations based on the fitted model and adjust the weights of the model. The validation set helped us to finally evaluate the model. The models performance was measured based on the accuracy and loss values shown in the confusion matrix. After that, we execute the same learning process for the custom model, but by following the next configuration:

- Input image size: 64x64
- Batch size: 48
- # of epochs: 100 and 1000

[1] https://pytorch.org/docs/stable/torchvision/models.html

- # of convolution layers: 2
- # of Fully- connected layers: 3
- Kernel size for convolution layers: 5
- Optimizer: Adam optimizer
- Initial learning rate: 0.001
- Loss function: Cross Entropy Loss
- Activation function: Softmax

In both cases, just after load the images and prior to process them, we normalized the input data by using a mean [0.485, 0.456, 0.406] and a standard deviation of [0.229, 0.224, 0.225] for the three color channels retrieved from images.

## 2.3 Platform

We used a scientific workstation to run the learning process, ie.e to fin the models. First we started creating all the code necessary to prepare the data set and to train the classifiers. After that, we move the script to the machine in which we executed the learning process. That workstation has the technical specs detailed as following:

- Brand: Dell Inc.
- Model: Precision 5820 Tower
- CPU: Intel Xeon W-2133 CPU @ 3.6 GHz (12CPUs)
- Memory: 32 GB RAM
- GPU: NVidia Quadro P400 (2GB dedicated memory)

After training and testing the models, we exported them for future use. So, we decide to register the time execution and model size. To get the first, we used the internal system clock and record the starting time and the time when the process have finished. To obtain the models size, we exported the both, model weight and entire model, by using the package *pytorch* and its save function to produce a .pth file.

# 3 Results

The table (3), shows the results for the pre-trained model for 100 epochs. The general accuracy of this model was 97.84%. The obtained training loss was 0.0065 and the test loss was 0.1560. The model has a good fit for the data sets, resulting in a precision of 85% in precision and a sensitivity (recall) of about 0.90.

Table 3: 100 epochs results - pre-trained model - densenet-161

| | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| 0 | 0.99 | 0.99 | 0.99 | 320 |
| 1 | 0.87 | 0.97 | 0.92 | 150 |
| 2 | 0.98 | 0.97 | 0.97 | 287 |
| 3 | 0.99 | 0.97 | 0.98 | 144 |
| 4 | 0.99 | 0.93 | 0.96 | 267 |
| 5 | 0.95 | 0.99 | 0.97 | 252 |
| 6 | 0.95 | 0.97 | 0.96 | 212 |
| 7 | 1.00 | 0.99 | 1.00 | 805 |
| 8 | 1.00 | 0.96 | 0.98 | 57 |
| 9 | 0.99 | 0.99 | 0.99 | 240 |
| | | | | |
| accuracy | | | 0.98 | 2734 |
| macro avg | 0.97 | 0.97 | 0.97 | 2734 |
| weighted avg | 0.98 | 0.98 | 0.98 | 2734 |

The table (4), shows the results for the created custom CNN model after 100 epochs. The general accuracy of this model was 92.73%, the general training loss was 0.0001 and the test loss was 0.5083. The model has a considerable fit, the accuracy on all categories is very precise, resultant precision was about 75% and sensitivity (recall) was 70%. Those values suggest that the model does not perform as well as the pre-trained model.

Table 4: 100 epochs results - custom CNN model

| | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| 0 | 0.93 | 0.94 | 0.93 | 320 |
| 1 | 0.77 | 0.72 | 0.74 | 150 |
| 2 | 0.87 | 0.88 | 0.87 | 287 |
| 3 | 0.93 | 0.92 | 0.93 | 144 |
| 4 | 0.93 | 0.93 | 0.93 | 267 |
| 5 | 0.93 | 0.92 | 0.92 | 252 |
| 6 | 0.88 | 0.90 | 0.89 | 212 |
| 7 | 0.98 | 0.98 | 0.98 | 805 |
| 8 | 0.93 | 0.95 | 0.94 | 57 |
| 9 | 0.95 | 0.95 | 0.95 | 240 |
| | | | | |
| accuracy | | | 0.93 | 2734 |
| macro avg | 0.91 | 0.91 | 0.91 | 2734 |
| weighted avg | 0.93 | 0.93 | 0.93 | 2734 |

In order to achieve better results with the custom model, we decided to run that model again, but in this case for 1000 epochs. The table (5), shows the results in which it can be seen that general accuracy increased to 92.06%, and training loss was reduced to 0.000007 while test loss was 1.099852. We achieved a considerable accuracy on all categories, getting a whole precision of 90.44% and a sensitivity of 89.51% in recall.

Table 5: 1000 epochs results - custom CNN model

| | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| 0 | 0.94 | 0.95 | 0.94 | 320 |
| 1 | 0.71 | 0.66 | 0.68 | 150 |
| 2 | 0.87 | 0.85 | 0.86 | 287 |
| 3 | 0.92 | 0.92 | 0.92 | 144 |
| 4 | 0.89 | 0.91 | 0.90 | 267 |
| 5 | 0.91 | 0.92 | 0.91 | 252 |
| 6 | 0.89 | 0.88 | 0.89 | 212 |
| 7 | 0.98 | 0.99 | 0.98 | 805 |
| 8 | 1.00 | 0.91 | 0.95 | 57 |
| 9 | 0.93 | 0.96 | 0.95 | 240 |
| | | | | |
| accuracy | | | 0.92 | 2734 |
| macro avg | 0.90 | 0.90 | 0.90 | 2734 |
| weighted avg | 0.92 | 0.92 | 0.92 | 2734 |

Surrounding the models execution time, although the pre-trained model present a higher accuracy, recall, and precision, it takes a lot of time to learn (28 hours, 31 minutes, and 10 seconds). By the other hand, the custom model running 100 epochs took only 24 minutes and 12 seconds, while the same model running 1000 epochs took only 04 hours, 47 minutes, and 59 seconds. It is important to notice that the custom model needed significantly less time to run 100 epochs. Under certain situations, specially considering time and resources limitations, it could be more advantageous to select it.

Regarding the model size (exported the whole net and just weights), in the case of just the weights, the pre-trained model resultant file size was 115384 KB, while the custom model resultant file size was 244 KB. In the case of the whole model, the pre-trained model resultant file size was 115430 KB and the custom model file size was 259 KB.

# 4 Discussion

The both tested machine learning models achieve a high performance classifying the PlantVillage dataset, for tomato samples. To accomplish the proposed objective, the both models result in pretty good performance, enabling the creation of new solutions in fields where it is needed. We agree other researchers [17], in the way there are needed more research in certain circumstances. However, as mentioned in [14], it is required to work in feature engineering, a major sub-field in computer vision, in this case by analyzing the data set changes. As indicated in [18], current solutions trained with PlantVillage data set fail in real case scenarios, due to lightning conditions and similarities among diseases.

The future perspective is oriented to solve the issue of uncertainty where identification of crop disease will be in more uncertain situations, and the use not only limited to smartphones, but also to other devices like drones. Recent studies have shown training models using convolution neural networks are a very good way to classify images from different classes. According to Krizhevsky *et al* [19], in more recent years those approaches are starting to grow in success. In this case using a customized CNN the estimation showed that although the accuracy of the general model is lower than normal approach, but the estimation is significantly faster. That means custom solutions can help on disease detection, but should not be already a replacement for traditional approaches, like laboratory revision.

There are two main limitations on the models tested, the models fit by using the images from PlantVillage dataset. Also, they classify single leaves that come from very homogeneous backgrounds. It is also important to note that models are based on diseases that are presented on plant leaves, not other parts of the plants. According to Sharma *et al* [20], to be able to use deep learning approaches, two factors need to be taken into account for different datasets. First if the new dataset is smaller or bigger, and secondly the

similarity of the used data to fit the new model.

Besides, we consider that it is possible to create individually models for crops. It is not required to create an unique model for all the clases, due to diseases for other crops can be very similar, and that can carry out in detection errors. Usually, the end-user (farmer) well-know the crop in which is making the analysis. For that reason, s/he can select the crop at the beginning, and next use the corespondent model to execute the detection.

# 5 Conclusion

This research focused on horticultural crops disease detection by using machine learning techniques by analyzing plant leaf images. The main contribution of this study is the comparison made in which we analyze a custom CNN model against a pre-trained CNN model for plant disease detection. The comparison shows that the normal pre-trained model has an overall superior accuracy, but takes significantly more time to run than the custom model. It is important to notice that there are a lot of opportunities to create mobile-based solutions for plant disease detection. In that way, we proposed to use a custom model instead of pre-trained due to devices resource limitations. Several studies creates solutions that process all the images from the PlantVillage data set. We consider that it is possible to create several models, one for each crop, obtaining better execution times, accuracy and precision for the classifiers. We agree other authors in relation to there is needed to collects more images, to increase the performance of the detection.

As a future work, we propose to improve the custom model, changing the hyper-parameters. By improving the custom model it can be possible to increase the performance of the solution. Also we propose to work with other data sets.

# References


[1] S. E. de Planificación Sectorial Agropecuaria (SEPSA), "Boletín estadístico agropecuario. serie cronológica 2016-2019. edición 30." http://www.mag.go.cr/bibliotecavirtual/BEA-0030.PDF.

[2] I. Secretariat, "Ippc annual report 2019 – protecting the world's plant resources from pests," 2020. Rome.

[3] M. M. A, Tai and H. C, "Threat to future global food security from climate change and ozone air pollution.," Nature Climate Change, vol. 4, pp. 817–821, 2014.

[4] S. E. de Planificación Sectorial Agropecuaria (SEPSA), "Informe de gestión del sector agropecuario y rural." http://www.sepsa.go.cr/docs/2018-005-Informe_Gestion_Sector_Agropecuario_2014-2018.pdf.

[5] C. R. Vargas and J. Nienhuis, "Cultivo protegido de hortalizas en costa rica," Revista Tecnología en Marcha, vol. 25, p. 10, Aug. 2012.

[6] S. A. Monge, "Cambio climático y plagas en el trópico," Alcances Tecnológicos, vol. 12, pp. 59–65, Oct. 2018.

[7] G. B. Singh, A. and S. Sarkar, "Deep learning for plant stress phenotyping: Trends and future perspectives," Trends in Plant Science, vol. 23, pp. 883–898, Oct. 2018.

[8] W. Z. Rawat, W., "Deep convolutional neural networks for image classification: A comprehensive review," Neural Computation, vol. 7, no. 10, pp. 14–19, 2016.

[9] "plant_village dataset." https://www.tensorflow.org/datasets/catalog/plant_village. Accessed: 2020-04-19.

[10] D. P. Hughes and M. Salathe, "An open access repository of images on plant health to enable the development of mobile disease diagnostics," 2015.

[11] "Plant village." https://plantvillage.psu.edu/. Accessed: 2020-04-19.

[12] A. P. J, "Data for: Identification of plant leaf diseases using a 9-layer deep convolutional neural network," 2019.

[13] H. D. Mohanty, S. and M. Salathé, "Using deep learning for image-based plant disease detection," Frontiers in Plant Science, vol. 29, no. 9, pp. 2352–2449, 2017.

[14] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," Frontiers in Plant Science, vol. 7, Sept. 2016.

[15] M. Zabir, N. Fazira, Z. Ibrahim, and N. Sabri, "Evaluation of pre-trained convolutional neural network models for object recognition," International Journal of Engineering & Technology, vol. 7, p. 95, Aug. 2018.

[16] S. Dey, PYTHON IMAGE PROCESSING COOKBOOK: over 60 recipes to help you perform complex image processing ... and computer vision tasks with ease. United Kingdom: PACKT Publishing Limited, 2020. OCLC: 1141521266.

[17] Y. Toda and F. Okura, "How convolutional neural networks diagnose plant disease," Plant Phenomics, vol. 2019, no. 9237136, pp. 1–14, 2019.

[18] D. Singh, N. Jain, P. Jain, P. Kayal, S. Kumawat, and N. Batra, "Plantdoc," Proceedings of the 7th ACM IKDD CoDS and 25th COMAD, Jan 2020.

[19] S. I. Krizhevsky, A. and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Advances in Neural Information Processing Systems, p. 1097–1105, 2012.

[20] S. W. G. P, Sharmaa; Y, "Performance analysis of deep learning cnn models for disease detection in plants using image segmentation," Information Processing in Agriculture, p. 1097–1105, 2019.