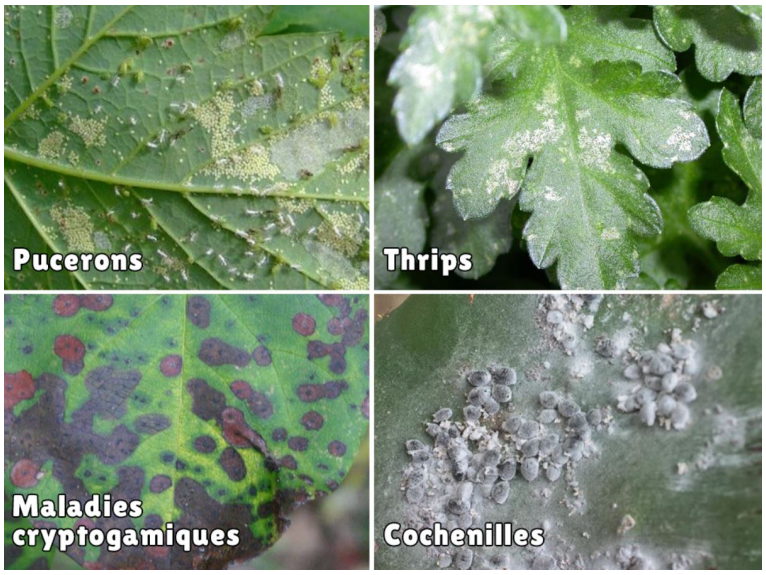
UCLouvain

AGRO LOUVAIN
BIOINGÉNIEURS

LBIRTI2101B - Data science
in bioscience engineering (Partim B)

Image-based diagnostic of plant diseases



Authors:

Claessens Oliver
de Marnix Madeleine
Gandjeto Martine
Heroufosse Gauthier
Van Bambeke Quentin

Professors:

P. Bogaert
E. Hanert

Year 2021 - 2022

Contents

List of Figures

List of Tables

1 Introduction

In the context of our bioengineering studies at UCLouvain, we are following the course "LBRTI2101B - Data science in bioscience engineering (partim B)" for which we implemented a project that aims to identify plant diseases based on images. The objective is to identify a practical problem, in our case what can be found on these images, to manage the information and to make conclusions.

Since the choice of the topic was very broad, we decided to go for a subject that is related to our masters in environment and agronomy: plants, their diseases and the climate that can influence the conditions in which these organisms are found.

To achieve this goal, we worked with artificial intelligence, which has become more and more widespread in the last few years and which allows strong performances for image recognition. This project has allowed us to learn many things in a domain that was previously quite unknown to us.

This report is structured as follows: we will start with a general presentation of the topic in which we will discuss the context, the objectives of the project and the potential applications. Then, we will discuss about the tools we used, such as machine learning, deep learning, and convolutional neural networks.

Next, we will go into practical details on our implementation. We will present the dataset used, the choice of packages and softwares, methods, and finally expected results. The last section will present the performance parameters chosen for our data analysis and the results obtained with the models for all implementations. We will conclude this report with some possible ways of improvement.

2 Presentation of the project

2.1 Context

Plants are the main source of all human food production. The current world population is growing and will continue to do so in the years to come. In the meantime, challenges related to food are already highlighted by the one billion human beings still suffering from hunger. In addition, climate change is disrupting an already vulnerable system by making farming conditions more difficult. The increasing temperatures and rainfall in our regions are bringing new pests that are already having dramatic effects on certain productions (notably the **olive plague** (Milan, 2020) and the increase of **mildew** and **powdery mildew** on the vine).

Figure 1: Mildew attack on a squash leaf

Most studies predict the extension of similar phenomena to other crops. These pests reach ecosystems with native species that are not prepared for these new attacks, causing serious production losses. To make matters worse the increasing globalization, and the development of new commercial roads are increasing the transport of insects and microorganisms across the earth. Unfortunately, our best strategy at the moment is prevention, since treatments are often bypassed by the pests, which become more and more resistant to them.

As future bioengineers, our challenge is to think about ways to reduce losses in the food production system as much as possible. We have decided to contribute to this task by focusing on the losses linked to the production in fields, which is currently estimated to be between 10 and 60% of the total losses depending on the product (SOLAAL, 2018). These losses are generally due to several factors, such as nutrient or water deficiencies, but also due to pest attacks as previously mentioned.

These diseases are numerous and can affect plants in many different ways. They can

manifest themselves with marks on the leaves, on the stems or on the roots. Some symptoms are inoffensive, but others are symptomatic of serious diseases and it is necessary to be able to recognize and identify them in order to treat them correctly. Their early detection could allow to limit the damages caused to the plants, and thus the yield losses.

2.2 Specific goals

The goal of this project is to implement an automatic detection of diseases based on leaf images. We will focus on this part of the plant since it seems difficult to combine in the same algorithm a detection on roots, stems or flowers, for example because these parts are hidden under the soil or are present only during a short period of time. Also, as it is our first time dealing with Deep Learning, we will simplify the problem to match our skills. This method still saves a significant amount of work for the farmer, while giving better results than detection by human observation.

To achieve this, the model must be able to perform different actions: first recognize the diseased plants from the healthy ones, and then classify the individuals by disease type and by species. Indeed, it is obvious that not all diseases are found in all types of plants. It seems to be a good objective for this first project.

2.3 Possible applications

To make our model useful in the world of agriculture, the use of technology seems to be the best option. Indeed, modern digital technologies such as smartphone applications are simple and fast communication methods (Ministère de l'agriculture et de l'alimentation, 2019). The first basic application is to allow farmers to use the model if they see anything strange on field. With a photo taken by their smartphone, the model could directly return the disease infecting their crops.

A possible improvement is to map all the information saved by the application. This could help follow the progress of an invasion in space in real-time, all you have to do is take a picture and it will be geolocated. Moreover, everyone could have access to this data: from the private individual who has an orchard in his garden, to farmers who take care of fields of several hectares, as well as governments or organizations that closely follow food production.

The use of this type of application could help prevent pest migration to new areas. Indeed, as explained, we live in a world where climate change and the increase in long-distance transportation amplify the migration of pests and diseases. Populations that are in contact with a new disease will not be able to identify it, since it is unknown to them, and will hence not be able to fight it. This application could therefore make it possible to identify a disease, already known in another part of the world, in a place where it had never been observed before.

This kind of application would make it possible to detect diseases at their first signs, which would increase the chances of being able to eliminate them especially before they

can do too much damage. It would also make it possible to apprehend the great variability of conditions and situations to which the detection of plant symptoms is submitted. This technology should be flexible and respond directly to farmers' needs and be able to adapt quickly to new situations and regions.

3 Mathematical fundamentals

3.1 Machine learning and deep learning

Artificial intelligence is a new concept that appeared in the 1950s. It can be defined as "*all the techniques that allow machines to achieve tasks and solve problems normally reserved for humans and certain animals*" (Chaire AgroTic, 2018). It is therefore a question of techniques that aim to reproduce, imitate, simulate intelligence, or in any case the abilities that we can associate with this term.

The field is therefore very large and is organized into multiple branches that combine cognitive sciences, mathematics, electronics, computer science, and so on. Machine learning is one of them, and gathers the means to train models and to classify data.

3.1.1 Machine learning

Machine Learning is a discipline in artificial intelligence that uses mathematical and statistical approaches to give to computers the ability to learn from existing data. This allows them to improve their performance in solving tasks without being explicitly programmed for each of them. To make it work, we need to start by manually selecting the most pertinent features of an image (such as angles, borders, the color of a leaf, and so on) in order to train the model to recognize and classify the images we give to it (Chaire AgroTic, 2018). To realize our project, we will make some Machine Learning but with the helps of neural networks. This is known as Deep Learning.

3.1.2 Deep learning

Deep Learning is a subcategory of Machine Learning. It is a group of tools and methods for automatic learning that are based on the use of neural networks. These neural networks are composed of several layers connected to each other. The more layers there are in the network, the deeper it is and therefore the more complex problems it is able to process.

This network helps to decompose a problem into very simple combinations in the first layers and then into more complicated combinations as we progress in depth in the network. We use the word "deep" to refer to the number of layers of neurons that make up these networks (Chaire AgroTic, 2018).

In order to simplify the understanding, here is a case of weather forecasting (represented in figure 2) for which the goal is to predict the weather from different variables such as

temperature, atmospheric pressure, hygrometry and wind speed. The expected result will vary between "1", which will represent "good weather" and "0" for "bad weather"(Chaire AgroTic, 2018).

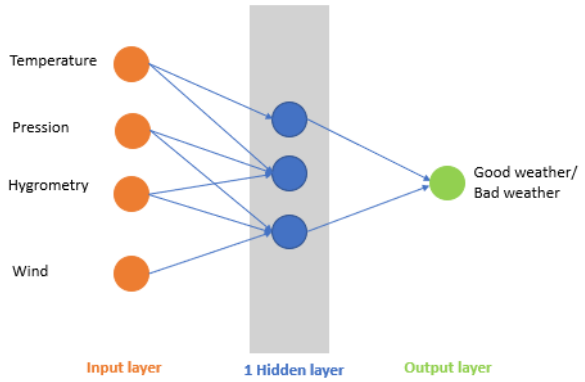


Figure 2: Figure showing the neural network (Chaire AgroTic, 2018)

However, it is important to specify that not all input parameters have the same importance on the final result. Indeed, depending on the temperature level, in relation to a certain wind speed and a certain humidity level, the same qualifier will not be given to the day's weather.

As just said above, this is an example of what a neural network can be used for. It will allow us to quantify the weights of the different variables for the expected prediction.

Similarly to biological neurons, artificial neurons can "transform" multiple input variables into a single output result. Nowadays, we can distinguish several large groups of deep networks that allow us to solve different types of problems, such as :

- Convolutional Neural Networks (ConvNets or CNN) ;
- Recurrent Neural Networks (RNR) ;
- Generative Adversarial Networks (GANs)

3.2 Convolutional Neural Networks

For the implementation of the model, we will use a Convolutional Neural Network (CNN) because it is the most efficient in terms of image analysis. CNNs reduce the size of the image in order to make them easier to process. As a result of the training this is possible without losing the main features useful for prediction.

CNNs are made up of three types of layers. Convolution layers, Pooling layers and Fully-Connected layers, as described in the following subsections (IBM Cloud Education, 2020):

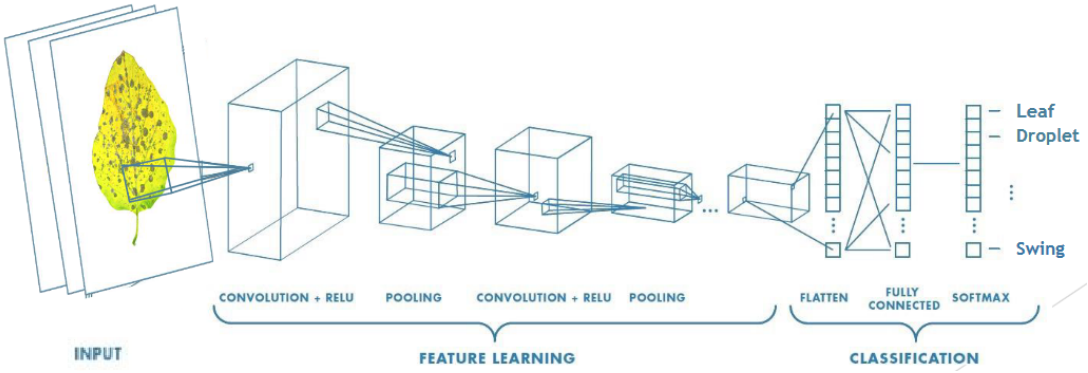


Figure 3: Generalised CNN diagram (Sumit, 2018)

3.2.1 Convolution layers

Images can be seen as a matrix of pixels, and each of these pixels contains a certain amount of red, green and blue (RGB). Convolution layers work by applying a filter (or kernel) which captures all the information of the image (Sumit, 2018).

The objective of the Convolution Operation is to extract the high-level features such as edges, from the input image. The more layers you add, the more feature you will extract. Of course, this operation needs to be done 3 times, as we have a three color-matrix to investigate. But in return, the size of the data is increasing, that's why we use a second type of layer.

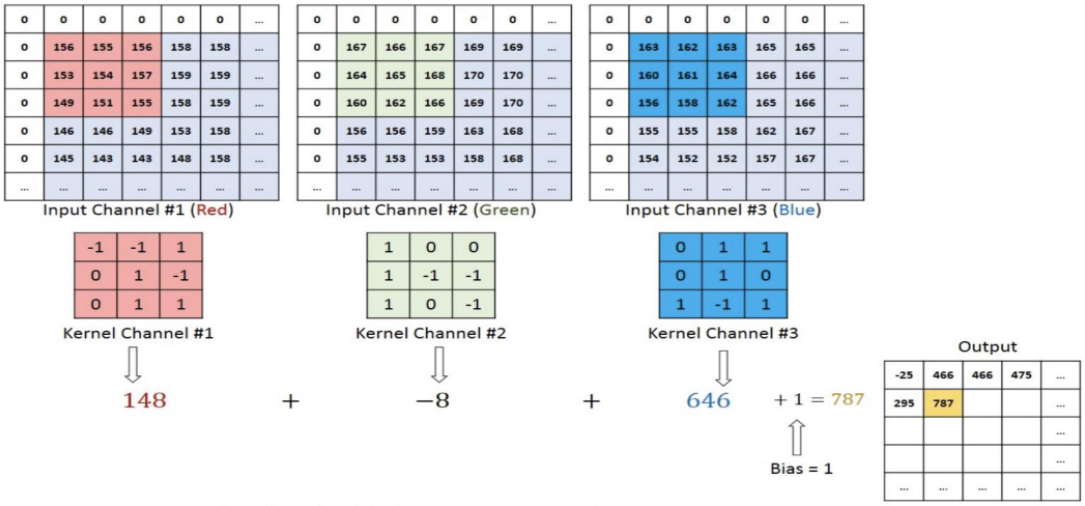


Figure 4: Example of Convolution diagram (Sumit, 2018)

3.2.2 Pooling layers

The result obtained in the previous step goes through a "Pooling" layer with the objective to reduce the size of the image and to bring out the dominant features of the convolutional layer. This allows a reducing of the computer power needed to achieve the goal. To give an example, a "Max-pooling" layer will always keep the maximum value present in the matrix (refer to figure 5).

This pooling step is very advantageous because it also eliminates the noise and errors that can be present on the image.

We will proceed in this way several times, with alternating layers of convolution and pooling. There is therefore once again a reference to "deep" learning, since there are many layers that come into consideration.

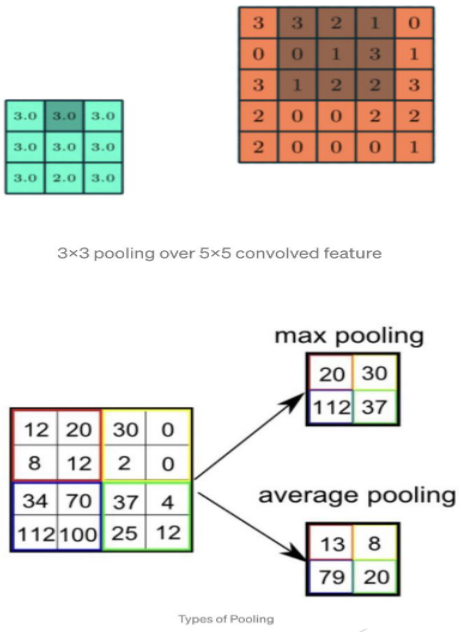


Figure 5: Figure showing the pooling diagram (Sumit, 2018)

3.2.3 Fully connected layers

The final passage through a Fully-connected layer allows to combine all the information coming from the different inputs, and to classify them in different classes. In our case, we will define these classes in advance. This is known as Supervised-Learning.

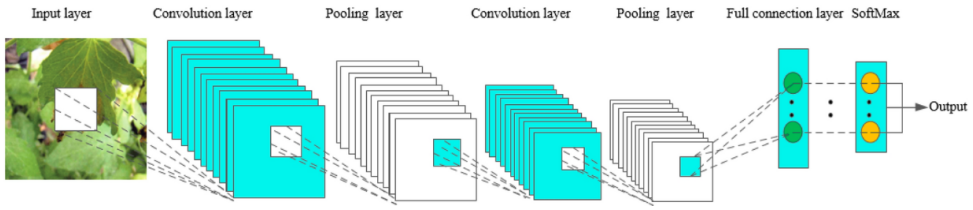


Figure 6: Figure showing the fully connected's structure (Wang & Xuewei, 2021)

3.2.4 Softmax function

The Softmax function is the last activation function of the neural networks. It is used to normalize the output of a network to a probability distribution over predicted output classes (Sako, 2018). The Softmax function is defined by :

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_i}} \quad \text{with :}$$

- $\vec{z}$: the output of the previous layer of the model ;
- z_i : the input values of the softmax function, they can be negative, zero or positive;

- K the number of classes in the multi-class layer ;
- e^{z_i} : application of the exponantial function ;
- $\sum_{j=1}^{K} e^{z_i}$: standardisation term, ensure that the sum of the components of the output vector $\sigma(\vec{z})$ is 1.

3.3 Models used

As the first implementation was done, we realized that a single training of the initially used model takes about 8 hours. Since several tests are required for the model to successfully work, we decided to do things differently : a second (smaller) model was built from scratch.

3.3.1 Simple 5-layers CNN

The small CNN built is composed of 5 layers : 2 convolution and max-pooling successions, and a fully-connected layer. Two benefits were identified with the introduction of this model. We first hoped that the training time would be drastically reduced ; and we will compare results obtained with both models.

3.3.2 GoogLeNet

The main CNN we used is called GoogLeNet (Alake, 2021), it was created in 2014. Like the other CNN'*s* developed so far, it consists of different layers, for a total number of 22, including convolution layers and pooling layers. The big improvement made with GoogLeNet is the introduction of inception modules. The purpose of such a module is to work on the same layer with several convolutions of different kernel sizes (1x1, 3x3, 5x5) and several max-poolings. The information is then joined and fed into the next Inception module.

The goal is to have multiple layers at the same level of depth: the structure becomes wider rather than deeper and therefore more efficient. However, gathering information from so many different layers drastically increases the number of operations required. This is where 1x1xN convolution layers become useful. N correspond to the depth of the input layer. More information can be found on (Sakthi, 2020).

This module is used to reduce the dimension and thus reduce the computation time. To compare, if we wanted to perform a convolution with a 5x5 kernel the number of operations is 112.9M whereas with the initial use of a 1x1 layer the number of operations becomes 5.3M.

The following diagram (figure 7) represents the final structure of an Inception module.

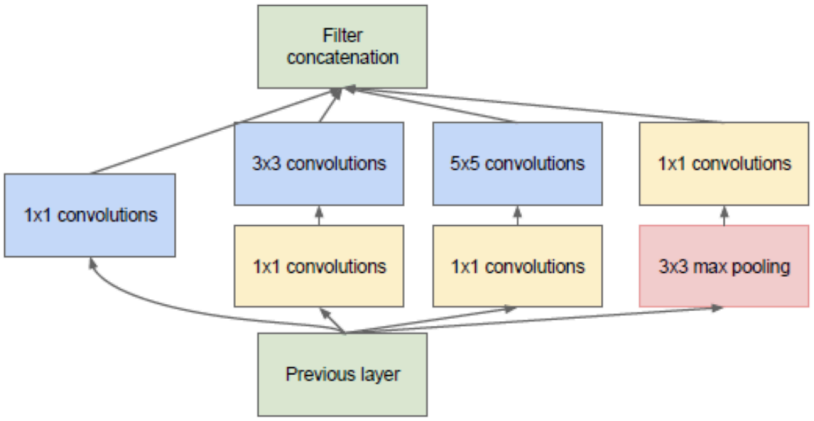


Figure 7: Figure showing the structure of an Inception model (Chatterjee, 2021)

Another big advantage of using GoogLeNet over a CNN that we would build ourselves is Transfer Learning. GoogLeNet was initially developed to classify objects into 1000 different classes. The idea of Transfer Learning is to keep the main trained structure and replace the final layers so that the CNN returns a result that matches the 39 classes in our Data Set. Thus, we do not need to train the whole model with random weights between the layers, but only the 39 classes for the final layers.

4 Implementation proposition

4.1 Data set

The database we used for this project comes from an online data directory called Plant Village and contains a total of 61484 images. The Plant Village dataset is an open access repository of images developed to help the development of disease diagnostic through machine learning and crowd sourcing. It is composed of healthy and unhealthy plant leaves, like shown below (Arun Pandian and Geetharamani, 2019).

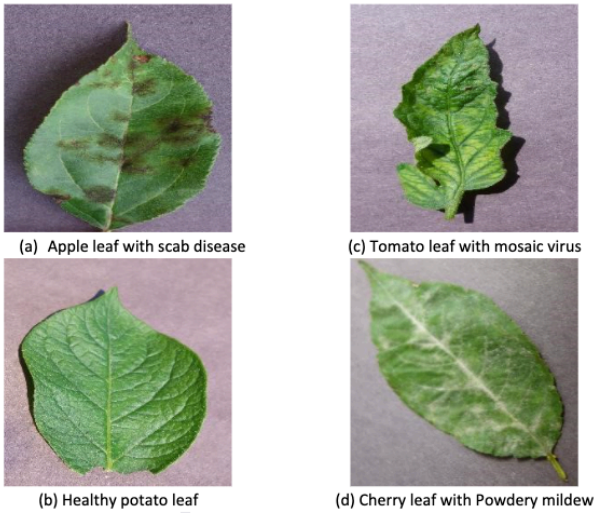


Figure 8: Image examples from the database

Although most of these images contain plant leaves (98.1%), some of them do not (1.86%). They are instead images of cars, houses, motorbikes, etc. To be more efficient for the remainder of the database presentation we have decided to focus only on images that contain plant leaves.

The dataset includes some augmented images. Image augmentation is achieved by image flipping, Gamma correction, noise injection, PCA color augmentation, rotation, and scaling. It is very useful to easily increase the size of a data set if it is too small.

The images are organised by plant category and type of disease. The following bar plot shows the number of images for each type of plant and the amount of which feature a disease. A disparity in these numbers can easily be observed, indeed for some plant categories (such as Blueberry or Soybean) there are no images of diseased plants, whereas others (such as Orange or Squash) contain only images of diseased plants. There is also a preponderance of Tomato images in our database (about 30.6% of the images) and they are mainly composed of diseased plants.

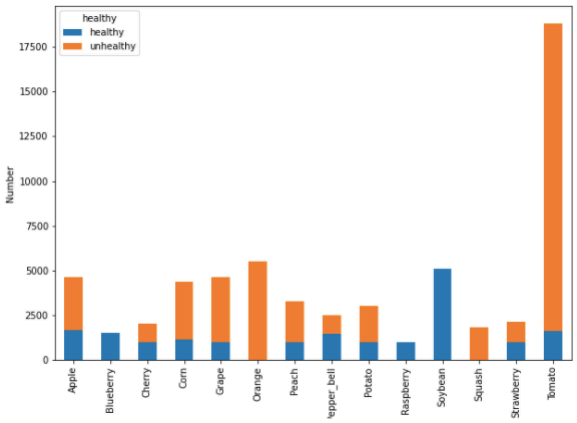


Figure 9: Figure showing the distribution of images by plant health status

All the original images in the database are in JPEG format (except for one who is in PNG format) and have an RGB mode (i.e. they are in colour). Although most of the images are the same size (256 * 256), not all of them are and therefore will required a size harmonization.

Our data set has been divided into three distinct parts for the training, validation and testing of our model. Initially, 60% of the images were for the training and the validation, then once the model has been optimised the 40% left is for the testing of our model. We will make this vary in order to maximize the accuracy of our model.

4.2 Packages and software details

For our project we decided to work with PyTorch. This is an open-sourced machine learning library based on the Torch library and released in 2016 by Facebook. There are multiple other Deep Learning libraries (such as Tensorflow) but we chose PyTorch as it provides a very strong performance in terms of image classification.

Torchvision is a library built to facilitate research and experimentation in the field of computer vision with PyTorch. Computer vision can be defined as a field of artificial intelligence that allows computers to derive meaningful information from images or videos. From this library we imported three packages including one called "transforms" that is used to perform the various different transformations needed for the image augmentation such as enlarging, rotating the images or changing their brightness.

We also used SubsetRandomSampler, this is a module which randomly selects samples of images from our data set in order to separate them into the three classes "train", "test" and "validation".

Finally we also used basic libraries such as Numpy, Pandas and Matplotlib. These are essential for scientific calculation, data structure and the creation of graphs.

4.3 Expected results

The output is a vector with a length of 39 representing the different probabilities of the image given to the model to belong to each class. We want our model to return a list of the 5 most likely classes to match the image. Logically, the more trained and efficient a model is, the greater the difference should be between the number of predictions for the most represented class and for the following four classes.

Then, the index of the class must be converted to the corresponding class name so as to understand the received output. Indeed, each disease associated to a plant has its own index number. In summary we would give an image of our test set to the model and it would be able to give us in return the corresponding class (species + disease).

Furthermore, we also expected the algorithm to provide the details of the different stages (training, validation and test precision) through which it passed as this would give an idea of the precision of the results obtained.

The prediction accuracy of the model on the testing set will then be measured. This is the percentage of correct predictions on the testing set. The prediction for an image is the class with the highest membership score.

Finally, the losses for each iteration will be examined. The loss gives an indication of the quality of the prediction. The higher the score is, the smaller the loss will be. The loss-function used is a cross-entropy loss measured like this:

$$\text{losses} = -\ln P_i$$

5 Presentation of the results of the achieved analysis

5.1 Performance parameters

During the analysis of the results, three main parameters were taken into consideration to measure the quality of the performance.

The first one is the time it takes to complete a cycle during the training of the model.

The second performance parameter is the losses that occur during the training and validation phases of the model. These losses represent the quality of the predictions that the model provides and, to express them, the model returns a probability of belonging to each class of the dataset. Then, only the highest probability is retained, which is noted P_i in the folowing formula below and the opposite of the neperian logarithm is taken : "$Losses = -\ln(P_i)$". Then, to obtain the global losses on the set, the average of all these "Losses" values is calculated for the whole set (training or validation). In summary, the closer the probability P_i is to 1, the lower the loss.

A third performance parameter is the calculation of the accuracy obtained on the testing sample. This step is done after the training phase of the model. For this, a function is used called " Accuracy ". This function calculates the accuracy of the model on the testing set. It simply divides the number of predictions that are confirmed to be correct by the total number of samples that are run through the model.

5.2 Transfer learning vs training from scratch

The pre-existing model used, GoogLeNet, whose final layers have been replaced to respect the number of classes present in the Dataset, has a large number of layers. This is why the training time is over 7 hours on the most powerful of our laptops.

The 5-layers CNN built from scratch is much smaller and runs much faster than GoogleNet. This model allowed us to perform a lot of tests for our implementations and to avoid having to troubleshoot in the script to facilitate the use of the large GoogleNet model.

We made several implementations and for each of them, we decided to compare the performances of these two models.

Initially, the entire Plant Village dataset was used, with 60% of the dataset for training and 40% for testing. The number of epochs was set to 4, this is the number of times that the training and validation set passes in its entirety through the model. Finally a batch-size of 8 was set, this is the number of images evaluated by the model before recalculating the weights related to the passing from layers to layers.

This epoch value is defined to limit the phenomenon of overfitting. This happens when the model spends too much time learning the dataset and starts to integrate noise or other

patterns specific to the dataset (training set) into its learning. When it is confronted with images coming from another dataset (validation set), the results are less good than if the learning had stopped earlier. The number of epochs was initially 5, but this increased the validation losses, resulting in a lower quality prediction. To reduce this phenomenon, the number of epochs was reduced to 4, which allows the losses on the validation set to no longer increase during the last iteration. The figure 10 shows the losses for the 5-layers CNN depending on the number of epochs and the figure 11 shows the losses for the GoogLeNet.

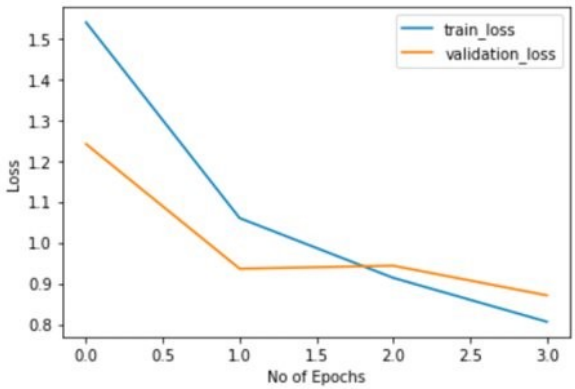


Figure 10: Losses by epoch for the 5-layers CNN

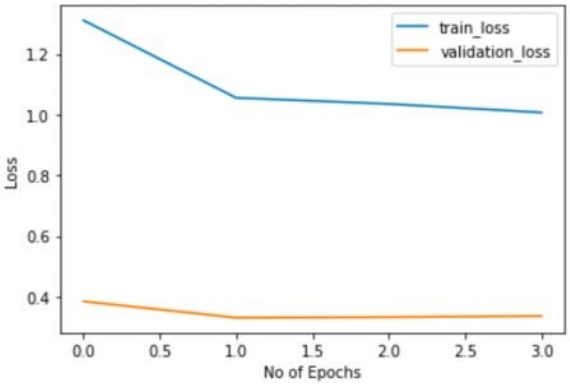


Figure 11: Losses by epoch for the GoogLeNet

Despite this decrease in losses, there is a large divergence between two successive predictions of the same leaf, which is not a good sign. This is probably due to the low number of layers. However, the only objective of this model was to verify the correct working by avoiding bugs, which is the case here. The goal is therefore accomplished. The figure 13, is the proof that the prediction varies considerably from one time to another. It shows the probability of membership to the first 5 classes for the figure 12, an Apple Black Rot.
The figure 14 shows the results of the probability that the GoogLeNet model provides.

Figure 12: Apple Black Rot

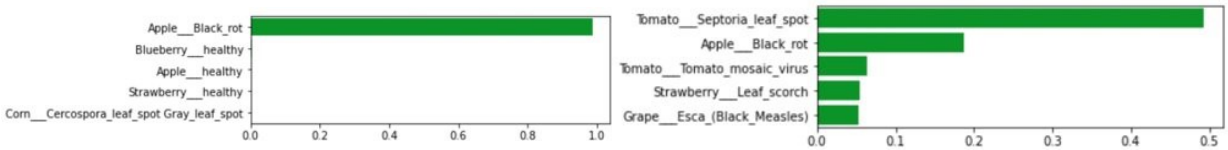


Figure 13: Prediction for an Apple leaf with black rot with the 5-layers CNN model

The table 1 shows the first results obtained. The small model takes 12 minutes per iteration and has a precision for the testing set of 75% while the GoogLeNet model takes 1h45 per iteration with a precision of 90%.

Figure 14: Prediction for an Apple leaf with black rot with the GoogLeNet model

Table 1: Summary of the first results

Variables	5-layers CNN	GoogLeNet (22 layers)
Mean time	12x4 = 48 minutes	around 7-8 hours
Precision of the testing set	75%	90%

5.3 Implementations to improve the analysis

Then, modifications were made to the dataset to try to improve the prediction performance of the model. The first change was to set up a distribution of 75% for training and 25% instead of 60% - 40% for testing to improve the training. The second change was to remove superfluous classes, such as those with only a few images or those with only sick or healthy plants. Therefore plant species containing superfluous classes are removed. These are framed in red on the figure 15. This operation reduced the number of classes from 39 to 30.

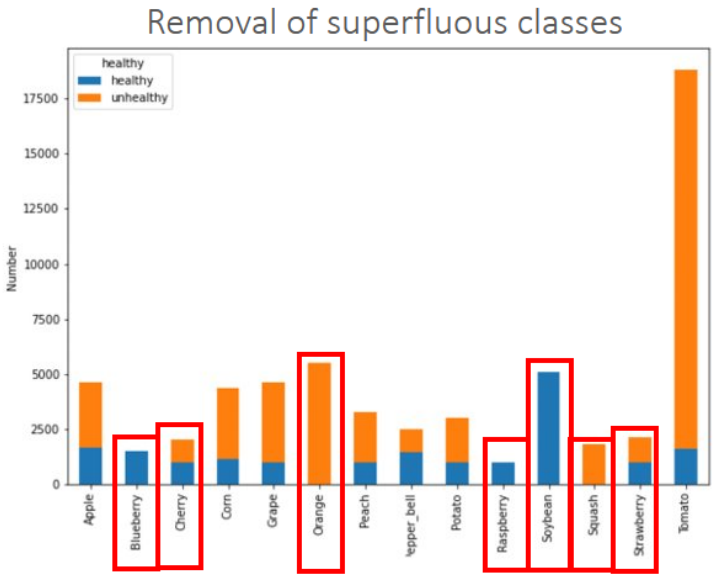


Figure 15: Figure showing the species that are keeped and removed

5.4 Final results

The figure 16 represents a table that summarizes the different results obtained, more specifically the duration of a complete cycle and the accuracy of disease identification, for

each of the three implementations for the small model of two layers and the large model (GoogLeNet). The three implementations are the original (60% - 40%), the implementation where the amount of images were increased in the training and the implementation where the images that were deemed less useful were removed.

	Original		Training optimisation		Classes removal	
CNN type	GoogLeNet	5-layers	GoogLeNet	5-layers	GoogLeNet	5-layers
Time	7 - 8h	48 min	9h	56 min	4h40	40 min
Precision	90%	75%	91 - 92%	76-77%	88%	76% - 77%

Figure 16: Summary of the results

This figure shows that the second implementation has generated an increase of about 20% on the training duration of a complete cycle while the accuracy only increased by 1 or 2%. On the other hand, the reduction of the number of classes, which is the third implementation, significantly decreases the training time for both models but does not have the same effect on the accuracy. The accuracy slightly decreases for the GoogleNet while it remains relatively stable for the five-layer CNN.

Thus to conclude on these results the best choice between the two CNN'*s* seems to be the Googlenet. Indeed, this one has a precision which is systematically higher than the two-layer CNN. On the other hand, between the three implementations, the one where the classes were removed seems to be potentially the best. Indeed, the reduction in time is very significant, especially for the GoogLeNet model which, in addition, does have a reduction in precision but only a minor one.

However this choice between the two implementations (columns two and three on the table) remains nevertheless to be discussed since a longer training time does not pose significant problems. Indeed, this step has to be done only once, so it does not cause much trouble to the users.

6 Ways of improvement

To go further, it is useful to realise that the results obtained with our model are quite satisfying but of course, it is far to be perfect. Indeed there are several elements that can be modified and that would generate better results. Those elements are presented below.

The first element is to increase the size of the dataset. As said we only had 61.486 images which is not much for this kind of model and it becomes even less when some classes are removed. Additionally having both healthy and diseased leaves for each class would also be relevant, as there would be no need to remove images and it would result in a more rich and balanced training.

Another very important detail is the quality of the image. Indeed, some of the images in this dataset are of very poor quality, which has strong negative effects on the accuracy of the results.

Furthermore, to reduce the problem of overfitting mentioned previously it would be pertinent to perform transformations on the images to increase the data set available for training.

Next, all this will allow the model to be able to detect diseases at certain stages of development in order to act at the right time in the treatment. Depending on the stage reached by the disease, we can adapt the solution or the quantity given to the plant in order to minimize the costs related to the medical monitoring of the plants (time, money, use of phytosanitary products, and so on).

The ultimate goal would be to make the model accessible to as many people as possible, the creation of an application could make this possible and could help identify contagion cluster grouping individuals contaminated by the same pathogen. A zoning strategy can thus be developed to stop the disease before it spreads to healthy individuals nearby (Ministère de l'agriculture et de l'alimentation, 2019).

Finally, a last parameter that could always help with the use of this model is more powerful computers. Indeed, this would allow to decrease as much as possible the time of training of the model in order to not lose too much time during the update of the database.

7 Conclusion

In conclusion, it seems pertinent to revisit the evolution of this project and its strengths and weaknesses.

First for the training portion of our model, we were able to compare transfer learning and learning from scratch and observe what each method implicated. This allowed us to better understand how complicated training from scratch can be and therefore how useful pre-existing models such as GoogLeNet can be for the elaboration of such a project.

The model was successful in achieving the initial objective of this project, indeed with the help of deep learning it was capable of identifying the species of the plant in the image and the disease that it most probably is suffering from.

However in an effort to improve the efficiency of our model, two different implementations were tested out; an increasing in the size of the training set and the removal of superfluous images from the dataset. These implementations had significant effects on the duration of the cycle and the precision of the identification. Furthermore, this analysis allowed us to arrive at the conclusion that the CNN model is the better of the two options due to its higher precision and that both implementations are advantageous, the one with the removed images seemingly being the better choice.

Nonetheless, some additional ways of improvement have been identified to further increase its accuracy notably a larger and more balanced dataset.

Furthermore, the realisation of this model also opened our eyes to the world of artificial intelligence, which allowed us to learn so much about PyTorch, Convolutional Neural Networks, GoogLeNet and their increasing usefulness in the world of agriculture.

References

Alake, R. (2021). Deep learning: Googlenet explained.
https://towardsdatascience.com/deep-learning-googlenet-explained-de8861c82765.

Arun Pandian and Geetharamani (2019). Data for: Identification of plant leaf diseases using a 9-layer deep convolutional neural network.
https://data.mendeley.com/datasets/tywbtsjrjv/1.

Chaire AgroTic (2018). Deep learning et agriculture: comprendre le potentiel et les défis à relever.
https://www.agrotic.org/wp-content/uploads/2018/12/2018/ChaireAgroTIC/DeepLearning/VD2.pd

Chatterjee, S. H. (2021). Various types of convolutional neural network.
https://towardsdatascience.com/various-types-of-convolutional-neural-network-8b00c9a08a1b.

IBM Cloud Education (2020). What are convolutional neural networks?
https://www.ibm.com/cloud/learn/convolutional-neural-networkstoc-types-of-c-yL2bT7qZ.

Milan, L. (2020). La peste des oliviers des pouilles menace le bassin méditerranéen.
https://lepetitjournal.com/milan/actualites/la-peste-des-oliviers-des-pouilles-menace-le-bassin-mediterraneen-282912.

Ministère de l'agriculture et de l'alimentation (2019). E-phytia:un portail d'applis qui révolutionne la santé des plantes.
https://agriculture.gouv.fr/e-phytia-un-portail-dapplis-qui-revolutionne-la-sante-des-plantes.

Sako, Y. (2018). is the term softmax driving you nuts?.
https://medium.com/@u39kun/is-the-term-softmax-driving-you-nuts-ee232ab4f6bd.

Sakthi, R. (2020). Comprehensive look at 1x1 convolution in deep learning.
https://medium.com/analytics-vidhya/talented-mr-1x1-comprehensive-look-at-1x1-convolution-in-deep-learning-f6b355825578.

SOLAAL (2018). Pertes agricoles : des leviers de réduction selon lademe.
https://www.solaal.org/pertes-agricoles-des-leviers-de-reduction-selon-lademe/.

Sumit, S. (2018). A comprehensive guide to convolutional neural networks the eli5 way.
https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

Wang, L., & Xuewei, J. (2021). Plant diseases and pests detection based on deep learning: a review. Plant Methods Vol 17, ISSN 1746-4811, BioMed Central.