Noname manuscript No.
(will be inserted by the editor)

PLANT DISEASES DETECTION AND CLASSIFICATION BASED ON DEEP LEARNING

immediate

the date of receipt and acceptance should be inserted later

Abstract In the CEMAC zone, the main economic activity carried out by the population is agriculture. Majority of the farms are owned and operated by families. The low productivity of these farms is due to several problems. These include plant diseases caused by bacteria, viruses, or fungi that result in considerable economic losses. Family farmers face difficulties knowing about these diseases. This problem can be solved by using deep learning. State-of-the-art approaches have built classifiers using images of plants. But some rely on handcrafted methods such as ANNs, and KNNs and others rely on fully automated methods such as convolutional neural networks. Nowadays, deep learning is increasingly being used for image classification thanks to convolutional neural networks. We are going to use them to solve the problem of the classification of plant diseases. We will apply GAP and LRN in neural networks to fight against overlearning and make the training smooth and stable. The main objective of our work is to build a classifier that can distinguish plant diseases. Then we will deploy it on different platforms. For this we will use as datasets the google plantVillage consisting of 54305 images separated in three groups (training images, validation images and test images) due to the sparsity of the data. We utilized the GoogleNet and AlexNet architectures, both of which are widely recognized as some of the most commonly used neural network in plant disease detection and classification, and applied them in our detection and classification system. We used GAP in the GoogleNet architecture and LRN in AlexNet, achieving a precision of **95.77%** with 16 GoogleNet models and a precision of **95.77%** with AlexNet for the classification of 38 classes.

Keywords
Plant diseases, machine learning, deep learning, convolutional neural networks, LRN, GAP

1 INTRODUCTION

1.1 Context

Agriculture is the main economic sector in CEMAC zone, after the hydrocarbons sector, whose production is currently developing in 5 out of 6 of the member countries of the CEMAC zone. Overall, more than 50% of the CEMAC population depends on agriculture [1]. The low productivity of family farms is due to many problems. These include the poor organization by actors and the biotic constraints (diseases and pests) [12]. Plant diseases are plant alterations caused by viruses, bacteria or fungi. They cause considerable economic losses in agriculture. Farmers face difficulties knowing about plant diseases due to lack of experience, hence the need to consult a field expert for analysis. The experts lack an automated system for the detection of plant diseases, since they have been using visual inspection so far.

1.2 Scientific problem

At the state of the art, [2] [7] [4] [6] have proposed approaches to detect and classify plant diseases using image processing and machine learning. These approaches build diseases classifiers using images taken on cultures. Nowadays, deep learning is increasingly used by the computer vision community and in many fields. The main advantage of deep learning in computer vision is the direct exploitation of the image without any artisanal features. [4] and [6] used deep learning to build classifiers while [2] and [7] used machine learning to build classifiers. foot of the classifiers. [2] and [7] are based on handcrafted features designed by experts to manually

Address(es) of author(s) should be given

extract relevant information for the classification image.
How to build a powerful classifier using convolutional neural networks? To answer this question we will use an image database by training a convolutional neural network where we will use the training, validation and test data. Afterwards, we will appreciate the results we obtained.

1.3 Objective

Knowing about plant diseases allows you to: facilitate the control of diseases and epidemics, reduce economic losses caused by the spread of pests, help family farmers in decision-making concerning the application of pesticides and fungicides, improve the quality of the family harvest, reduce the costs of production by applying chemicals only when necessary. The problem we are going to solve is therefore an image classification problem. Our main objective is to set up a classifier that makes it possible to distinguish plant diseases, and deploy it on different platforms.

1.4 Plan

Our work is as follows: in part 1 we will review the literature, in part 2 we propose an approach, in part 3 we implement our model, in part 4 we analyze our results, and finally we conclude our work.

2 Review of the literature for the classification of plant diseases

The state of the art is based on various existing works in the detection and classification of plant diseases [2] [7] [4] [6] that we have studied.

Al-Hiary et al [2] worked on the detection and classification of plant diseases, they used 500 images of 54 healthy and un-healthy plant leaves separated into 5 classes (diseases are: early blight, cottony mold, ash 55 mold, late blight, and tiny whiteness) in the AlGhor region in Jordan based on machine learning. The k-means 56 (near neighbors) method for image segmentation and ANN (artificial neural networks) for classification, and 57 feature extraction using the GLCM (Gray Level Co-Occurrence Matrix) method to calculate the features of 58 pixels located only within the boundaries of the infected areas of the leaf, they had an accuracy between 83% and 94%.

The process of building the detection model, and classification of plant diseases starts from image acquisition,and segmentation to the classification described as follows:
Given acquisition: digital images of healthy and unhealthy plant leaves of six classes (early blight, cottony mold, ashen mold, late blight, tiny and normal whiteness); are gotten from the environment using a digital camera;

Preprocessing: creating a color transformation structure for the RGB sheet image and independent color space transformation for color transformation;
Segmentation: grouping of images using the K-means grouping technique, identification of pixels of mainly green color and specified thresholding and variable calculated for these pixels according to Otsu's method which is applied in order to specify the value of variable threshold which chooses the threshold to minimize the intra-class variance of the threshold black and white pixels. The K-means clustering algorithm attempts to cluster objects (pixels in our case) based on a set of cluster K-number features. The classification is done by minimizing the sum of the squares of the distances between the objects and the centroid of the corresponding group or class;
Feature extraction: Texture features of segmented infected leaves of this phase are extracted using the GLCM (Gray Level Co-Occurrence Matrix) method. The color gray to extract features with the co-occurrence matrix:

The co-occurrence matrices in [13] consist in identifying patterns of pairs of pixels separated by a distance d in a direction . Generally, we consider $d = 1$ and θ= (0°, 45°, 90°, 135° . . .) multiples of 45°. If N is the maximum value of the gray levels, the matrix has a size N x N. For each pair (d, θ), we construct a matrix $\varphi(d, \theta)$. An average matrix is then calculated making it rotation-invariant. The characteristics are calculated from the following attributes: Angular momentum,a moment of production, sum, and difference of entropies, entropy, information measures of correlation, contrast, correlation.
Analysis and statistics: conversion of the infected cluster from RGB format to HS format, characteristics calculation is only for the pixels located in the boundaries of the infected areas of the sheet (deletion of the

healthy areas);

Training: the set of training features that is used to train the NN (Neural Networks) model. In the training phase, the connection weights were always updated until they reached the defined number of iterations or an acceptable error. Thus, the ability of the ANN model to respond accurately was ensured by using the Mean Squared Error (MSE) criterion to emphasize the validity of the model between the inputs and the output of the network;

Classification: a feature set of the test data is used to verify the correctness of the trained NN model;

Hossain et al [7] worked on the detection and classification of plant diseases. They used data (277 images of diseased and healthy plant leaves) collected from two large plant database websites of Arkansas and Reddit which are used for the acquisition of images separated into 5 classes. They used machine learning and the technique of image processing to overcome human visualization methods which are difficult for experts and required the adoption of scientific methods, which are very expensive according to experts. The process took place from l image acquisition, segmentation to classification where they used the k-means clustering method for segmentation, the GLCM method for extracting the features of pixels located only inside the boundaries of the infected areas of the leaf and the KNN classifier for the model training. They obtained a classification rate of 96.76% described as follows:

Data Acquisition: Digital images of infected and non-infected plant leaves of five classes are gotten from the environment using a digital camera. These are the images of Alternaria alternata, anthracnose, fire blight, leaf spot and citrus leaf canker.

Pre-processing: the RGB images of the leaves are transformed into an L*a*b* color space comprising an "L*" luminosity layer, an "a*" and "b*" chromaticity layer. The layers of chromaticity "a*" and "b*" provide the color information along the red-green axis and the blue-yellow axis respectively;

Segmentation: The small sample regions of a leaf image determine the average color of each sample region in the "a*b*" space. The k-nearest neighbor classifier with three neighbors is used for segmentation. The segmented regions are represented by the colors green, red and blue respectively for the leaf, the disease and the bottom part;

Dilation and erosion: the most interesting part of the leaf image is the disease infected part, and in the color segmented image it is represented by the green color. Thus, for thresholding with no fixed level to generate the binary image, a morphological opening is made which is followed by dilation and erosion is applied to the binary image to separate the infected region from the original leaf image. Dilation increases the width of the highest region, so it can sort out noise from images. Erosion is used to minimize objects in the image, and it reduces the width of the smallest region;

Feature extraction: Six features, including GLCM and color features, are extracted from the segmented part of the disease. These characteristics are: mean, standard deviation, energy, contrast, homogeneity and correlation;

Training: for K-NN, there is no actual learning phase. We load in memory the training data, where the leaf images are labeled with their classes;

Classification: Classification in KNN is the process of associating a class with a new sample based on the learning achieved by the classifier model during training. The classification of plant diseases into five classes is carried out using the KNN classifier. The leaf images (test images) are unlabeled and the algorithm gives the list of k nearest data points (training data point) to label the unlabeled point and classify their classes.

Three thermal cards (requiring 800Mo of Memory). DeChant and al[4] worked on the automatic identification of maize plants infected with fire blight from field images, using deep learning. Images of NLB infected and uninfected leaves were taken by hand with a Canon EOS Rebel or Sony a6000 camera at dates ranging from 28 to 78 days post inoculation (DPI). A total of 1,834 images were taken over eight dates. A total of 38 images were excluded due to poor quality. The images were first classified according to the presence or absence of any visible lesions on the image. Then, all visible lesions were marked with a line along the main axis of the lesion using the annotation functions of the Bisque image processing platform hosted on CyVerse (formerly iPlant). Infected and uninfected leaves were photographed in the inoculated test whenever possible. The 1028 images of infected leaves and the 768 images of uninfected leaves were randomly divided so that 70% of the images were used for training, 15% for validation and 15% for testing.

All network architecture choices and training were done without considering the test set, which was only used at the end to evaluate the performance of the final complete system. They developed a three-step process to analyze the images to determine if they contained infected leaves. During the first stage, they trained several CNNs to detect the presence of lesions in small patches of leaves. These CNNs were used in the second step to produce heat maps indicating the probability of infection of each region of the images. The third step used these heat maps to classify the images.

They used Adam's optimization algorithm in networks A and C of stage 1 and in the network of stage 3; RMSprop was used in Network B to add diversity to the training methods used in Stage 1 classifiers.

Training was done with NVIDIA Titan X GPU, take about 3 days per first stage network and 30 minutes for third stage network. At runtime, it takes about 2 minutes to generate a heatmap for one image (requiring 1581.6 GB of memory) and less than a second to classify a set of three heat maps (requiring 800 MB of memory).

They used an automatic system that can automatically identify diseases. This approach uses a convolutional neural network (CNN) computing pipeline, and the system achieved a classification rate of 96.7%, an accuracy of 96.8% (number of true positives [i.e. really sick] divided by the number of total images), and an error rate of 10%, really sick divided by the number of true positives plus false positives), 97.4% recall (number of true positives divided by number of true positives plus number of false negatives), and an F1 score (2 × precision × recall, all divided by precision plus recall) of 0.971 . A confusion matrix, which shows the distribution of errors across the set of tests that were not used for training.

This is why the step 3 classifier was needed; he learned to combine all local segment scores, including inaccurate ones, into an overall classification.

Francis and al[6] worked on the Detection and classification of plant leaf diseases using a convolutional neural network on the PlantVillage dataset. The dataset consisted of 44016 images. It has been divided into 38 different categories of 14 species. For the experimental purpose, they used 200 images for each class, which means a total of 7121 images were loaded in which 6408 sheets were used for training, and the remaining 713 images for testing. They used a CNN as a classifier, during pre-processing of the images from their original size, they were resized to the correct dimensions. The changes applied are rotation, zoom, height adjustment and width shift. The following changes: rotation, zoom, height adjustment and width displacement have been applied to the images to double or triple the number of samples.

For the conv2D layers, 32, 64 and 128 filters were selected and 1024 in the last fully connected layer. Part of the image was transformed (defined by kernel size) using the conv2D_filter for the kernel. The pooling layer is the second most important on CNN. In this case, MaxPool2D was used. This layer functions as a subsample filter.

Pooling has been used to minimize device costs and also, to some extent, to eliminate overfitting. The pooling and conv2D layers are used to extract features from images. Finally, they used 1024 fully 'Dense' two-layered connected filters. The probability distribution of the neural network outputs for each class (Dense(38, activation="softmax")) in this last layer.

They used Dropout which is a regularization that arbitrarily removes nodes from the layer (sets the weights to zero) on each sample. The Relu activation function was used in order to not to linearize the network. Using the "Flatten" layer in the model, the feature maps was transformed into a single one-dimensional vector.

The convolutional neural network algorithm for plant disease detection and classification was run in python, and driving tests are performed on google colab with an integrated GPU: Tesla K80, 25.51GB of RAM, and 68.40GB of Hard disk. They used 90% of the data for training the model and 10% for the test and they got a classification rate of 99.89%. They trained the model for 200 training epochs. Comparison of methods and criticism of existing methods/approaches

We have studied several existing methods/approaches. Each method has its advantages and disadvantages. ANNs are methods that also produce good accuracy. They are able to work with noisy data or incomplete data, but the problem of overfitting is not always obvious. They face difficulties in handling large images. We also have the KNNs methods which are very simple, easy to implement but which consume huge amount of memory, they slow down considerably as the number of observations and/or depends on independent variables increases. With (ANNs and KNNs) classification methods, all the steps are not fully automated (extraction of features is done in an artisanal way), it is first necessary to extract data features from the data before passing them as input to the classifier.

CNN are classifiers which are able to extract features automatically. CNN replaces artisanal methods where you first need to extract features by hand before training the classifier to understand. One of its disadvantages

Table 1: comparison of methods [2] [7] [4] [6] for the detection and classification of plant diseases

Article	Year	No. of Images	Nbr. ima ges	Met hod	Benefits	Disadvantages	Rate of classification
[2]	2011	6	502	ANN	- It's abilities to learn how to model systems are adapted to the treatment of linear systems. - The ability to work with noisy or incomplete data.	- The problem of overlearning learning is not always obvious. - Difficulty managing large images.	94.67%
[7]	2018	5	237	KNN	- The algorithm is versatile. - The algorithm is super simple and easy to implement.	- The algorithm slows down considerably when the number of observations and/or dependent/independent variables increase.	96,76%
[4][6]	2017,2021	2, 38	1 834, 44016	CNN	- The use of a single weight associated with the incoming signals in all neurons of a single convolution nucleus. - The main advantage is their good accuracy in image recognition problems. of image recognition.	- High computing cost. - If you don't have a good GPU, they're pretty slow to train (for complex tasks); - They needed They needed a lot of training data.	96,7 %/99,89%

is its huge time consumption for the model training but gives very good performances. We note that [4] used the set of methods which combined the results of different classifiers, which often exhibit better performance than a single classifier. They got the best result with a combination of three of the first stage networks. However, even when they used the single-network heat maps in the third stage, they still observed a significant improvement over the baseline initial network, which took scaled-down versions of the full images as input. The performance of neural networks is greatly affected by the amount of data available for training. It's also possible, however that producing final classifier independent heat maps introduces some kind of regularization, as such an end-to-end system could more easily scale to the training set, which would be particularly dangerous for training sets as small as this.

In [6] the authors showed that the recognition accuracy was improved by adjusting the number of epochs, changing the training and test combinations, modifying the dropout values and the linear unit rectified functions (Relu). They used 200 images for each disease class due to the small amount of RAM supporting the machine, which is one of the drawbacks of the proposed system. Nowadays, there is CNNs architecture which has been used for plant image classification in many documents and has produced a good performance. The architecture used in [4] was not easy to set up, as choosing the best classifiers was not an easy task. The performances obtained by these classifiers can be improved by other techniques. In the architecture setup, the major drawback is that the learning and loss curves oscillate a lot. This can sometimes make interpretations difficult, and the risk of predicting data with noisy data can be high (see Figure 1).

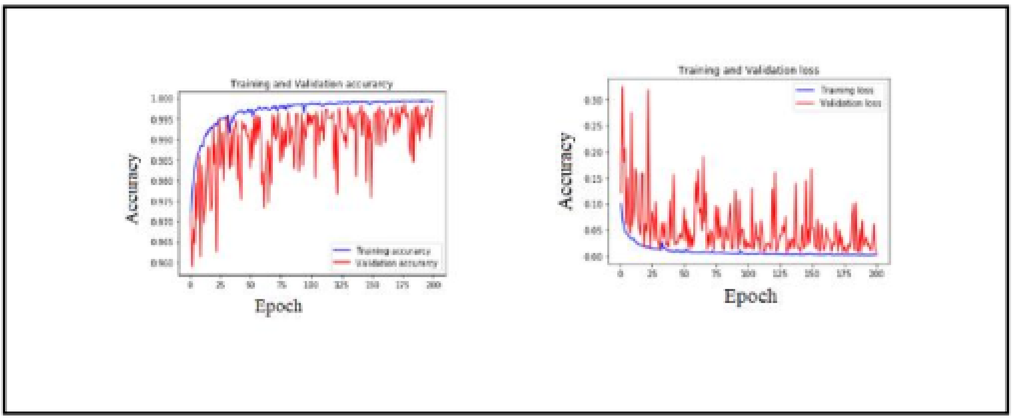


Fig. 1: training and validation curve [6]

How to build a smooth and stable trainable classifier using a CNN which does not require an assembly of several CNNs? In the next part, we will use existing architectures for image classification and we will add GAP and LRN to improve learning. The GAP and the LRN will allow us to avoid experiments on the operation dropout as assets [6] in general, the operation dropout of the probability value is chosen to be at 0.5. Several experiments were carried out from 0.5 to 0.7, with a difference of 0.05%.

3 Methodology

In this part, we explain the relevance of the approach used to solve the problem, describe the model we have chosen, explain the operations and functions of the model, compare the different architectures, explain the choice of architectures derived from our model, describe the color models that allow us to visualize images.

Deep Learning extends classic Machine Learning by adding more depth (complexity) to the model, as well as transforming the data using various functions that allow the representation of data in a hierarchical manner across multiple levels of abstraction. An important benefit of Deep Learning is feature learning, i.e. the function automates the extraction of features from the raw data. The characteristics of the higher levels of the hierarchy are gotten from the composition of the lower-level characteristics. It can solve particularly well and quickly more complex problems due to the more complex models used, which allow massive parallelization. These complex models employed in Deep Learning can increase classification accuracy or reduce errors in [8] regression problems. All the steps from pre-processing to feature extraction are fully automated and the ability of networks to extract features themselves.

– Classification: we can pass one or more of plants images that are different from the training images and the model will tell us if they are diseased or not and to which class of disease they belong; For each step of the model, we present input data and output information. We use a convolutional neural network architecture (the GoogleNet or AlexNet architecture) and a preprocessing algorithm to implement the model. We used the [5] and [3] documents to build the model as follows (see Figure 2):

Model Description The model described in Figure 2 allows us to detect, classify and visualize different plant diseases, it is built by following several steps such as:

- Data collection: the collection of images of healthy and unhealthy plants leaves by experts in the field;
- Labeling: the labeling of healthy and unhealthy plants leaves by separating them into several classes according to the disease;
- Pre-processing: resizing of images of diseased and non-diseased plants, rotation over several angles (45 degrees, 120 degrees, . . .) while keeping the original images;
- Segmentation: the model performs thresholding to select pixels according to a certain intensity;
- Feature extraction: the model extracts the pixels that characterize the important points of each image according to the classes using convolution and pooling;
- Training: the model learns through extracted features and this can be done over several epochs (hence the use of gradient descent);
- Classification: we can pass one or more of plants images that are different from the training images and the model will tell us if they are diseased or not and to which class of disease they belong;

For each step of the model, we present input data and output information. We use a convolutional neural network architecture (the GoogleNet or AlexNet architecture) and a preprocessing algorithm to implement the model. We used the [5] and [3] documents to build the model as follows (see Figure 2):

3.1 GAP (Global Average Pooling)

Classic convolutional neural networks perform convolution in the lower layers of the network. For classification, the feature maps of the last convolutional layer are vectorized and fed into fully connected layers, followed by a softmax logistic regression layer. This structure bridges the gap between the convolutional structure and traditional neural network classifiers. It treats convolutional layers as feature extractors, and the resulting feature is classified in the traditional way. However, fully connected layers tend to over-learn each other, which is detrimental to the generalization of the entire network. Dropout is used as a regularizer that randomly sets a certain percentage of activations to the lowest value. It has improved generalizability and largely prevents over-learning [10].
Fully connected layers in CNNs. The idea is to generate a feature map for each corresponding category of the classification task in the last step. Instead of adding fully connected layers on top of the feature maps, we take the average of each feature map, and the resulting vector is fed directly into the softmax layer. One of the advantages of global mean pooling over fully connected layers is that it is more native to feature maps [10].

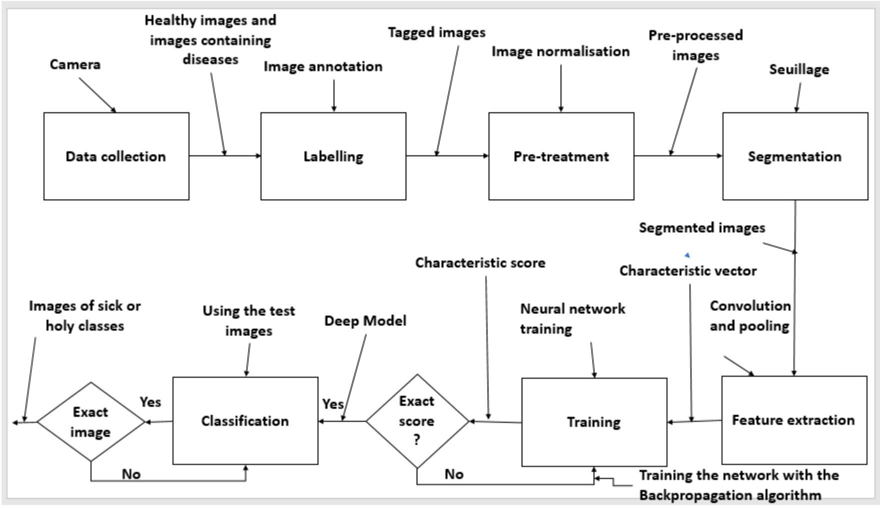


Fig. 2: Plant disease detection and classification model

One of the advantages of the overall average over fully connected layers is that it is more native to the convolution framework by strengthening the correspondences between feature maps and categories. Thus, feature maps can be easily interpreted as category confidence maps. Another advantage is that there are no parameters to optimize in the clustering of the global mean. Furthermore, the clustering of the global mean summarizes spatial information, which makes it more robust to spatial translations of the [10] input.

3.2 The LRN (Local Response Normalization)

ReLUs have the desirable property of not requiring input normalization to prevent them from saturating. If at least a few training examples produce a positive input for a ReLU, learning will occur in that neuron. However, we still find that the following local normalization scheme facilitates generalization. We denote by $a^i_{x,y}$ the activity of a neuron calculated by applying the kernel i to the position (x, y) then by applying the non-linearity ReLU, the normalized activity of the response $b^i_{x,y}$ is given by the expression [9]

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=max(0,i-n/2)}^{min(N-1,i-n/2)} (a^2_{x,y}) \right)^{\beta}$$

Where the sum is over n "adjacent" kernel maps at the same spatial position, and N is the total number of kernels in the layer. The order of the core cards is of course arbitrary and determined before training begins. This type of response normalization implements a form of lateral inhibition modeled after that found in real neurons, creating competition for important activities between neuron outputs computed using different nuclei. The constants k, n, , and are hyperparameters whose values are determined using a validation set [9]. We will apply this normalization after applying the ReLU non-linearity in some layers.

3.2.1 Choosing the architecture

There are several architectures for the classification of plant diseases, according to [11] we will make our choice in order to have a better classification performance. We will choose the AlexNet and GoogleNet architectures for the implementation of the model because they are the most used at present according to [11].

In the architectures we have chosen, we will use GAP and LRN to make the training smooth and stable and to fight against overlearning. We have thus shown how relevant the Deep Learning approach is, and in this approach, we have described the different mathematical theories based on the architecture of the convolutional neural network which allows us to detail the different operations used in the model, then we have chosen the two most used architectures. We will use the GAP and the LRN in the chosen architectures to implement the model in the following part.

4 List of tools and materials

There are several tools (hardware and software) for implementing the plant disease detection and classification model. We will implement with the most used architectures. These tools and materials will allow us to train a model.

4.1 List of materials

The characteristics of the materials used for the experiments are:

- Dual core computer (Genuine Intel(R) CPU, T2250 @ 1.73GHz 1.73 GHz, RAM(2.00GB) which serves as an interface to access Google colab,
- 4G Mobile Phone,
- 4G modem + internet connection,
- GPU (Google compute Engine backend, RAM (12, 72G), Disk (358, 27)).

4.2 Tools

The major digital players offer all their open-source frameworks and tools for creating neural networks: TensorFlow at Google, Torch at Facebook, Cortana NTK at Microsoft, the Watson platform at IBM, or even DSSTNE at Amazon. One of the frameworks that stands out, at least in terms of use by startups, is TensorFlow, whose development was initiated by Google. It works embedded as well as on servers as in the cloud. It is the framework with the functional spectrum which seems the widest, and which is easily deployed on parallel architectures, in particular those which are based on GPUs such as those of Nvidia. Tensorflow is the most appreciated by startups because it is the most generalist of the frameworks [5]. Table 3.1 presents several frameworks. We choose Tensorflow which is the most used software and has such rich documentation on the internet and a large community. We will use it to train the model not to be confused with the abstract model that we presented in the methodology. We have other languages and frameworks that allow us to deploy the solution:

- Tensorflow js: to convert the pretrained Tensorflow models and deploy the machine learning model in the browser;
- HTML5: used to design web pages;
- CSS3: for the presentation of HTML documents;
- JavaScript: for interactive web pages;
- PHP: used to produce dynamic web pages via an HTTP server;
- Laragon server or Xampp server: to deploy locally;
- Git: for decentralized version control;
- Flutter: which is Google's UI toolkit for creating beautiful, natively compiled apps for mobile, web, desktop, and embedded devices.

5 Model training

We will describe how we can train the model based on several phases, each phase contains several steps which we have represented in Figure 3.

5.1 Data collection and pre-processing phase

To train the model, we always need the data, and training of the model always begins with the data collection step, followed by the data labeling step, then the pre-processing step described as follows:

- Data collection: The data collection step is essential because without data we cannot form a model. The deep learning model needs a large amount of data to produce a good performance and avoid overfitting. Few data on plant diseases are publicly available these days. We will use the Google Plant Villages as indicated on the site https://www.crowdai.org/. We use a public and quite famous database, the PlantVillage Dataset. This dataset was released by crowdAI during the "PlantVillage Disease Classification Challenge". The dataset includes approximately 54,305 images of plant leaves collected under controlled environmental conditions. Plant images cover the following 14 species:

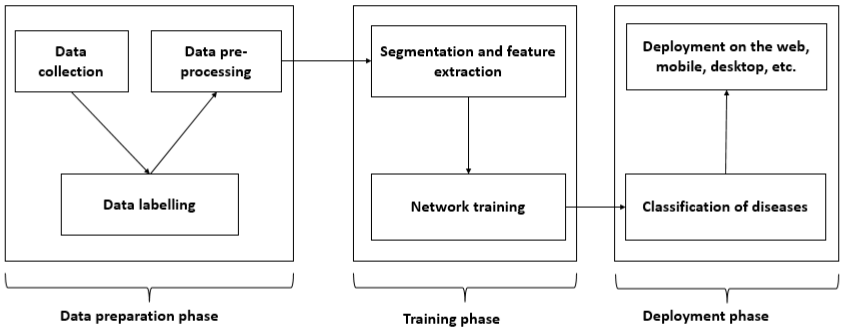


Fig. 3: General architecture for the detection and classification of plant diseases

apple, blueberry, cherry, corn, grape, orange, peach, pepper, potato, raspberry, soy, squash, strawberry and tomato. The dataset contains a total of 38 classes of plant diseases and one class of background images listed below: The dataset contains images of 17 basic diseases, 4 bacterial diseases, 2 diseases due to mold, 2 viral diseases, and a disease caused by a mite. Additionally, 12 of the cultivated species are healthy leaves that are not visibly affected.

Fig. 4: Leaf of sick and non-sick plants. source: https://www.crowdai.org/ consulted on 06/16/2021

– Data Labeling: the labeling or labeling stage a human expert labels the images collected, there are two types of labeling:
  • Weak labeling: which is done by experts in an agricultural field, identifies the disease without any addi-

Number	Class	Training datasets	Validation datasets	Test datasets
01	Apple Scab	504	114	12
02	Apple Black Rot	496	113	12
03	Apple Cedar Rust	220	50	5
04	Apple healthy	1316	297	32
05	Blueberry healthy	1202	270	30
06	Cherry healthy	684	153	17
07	Cherry Powdery Mildew	842	189	21
08	Corn Gray Leaf Spot	410	93	10
09	Corn Common Rust	953	216	23
10	Corn healthy	929	210	23
11	Corn Northern Leaf Blight	788	178	19
12	Grape Black Rot, Guignardia bidwellii	944	213	23
13	Grape Black Measles	1107	249	27
14	Grape Healthy	339	76	8
15	Grape Leaf Blight	861	188	21
16	Orange Huanglongbing	4405	992	110
17	Peach Bacterial Spot	1838	409	45
18	Peach healthy	288	65	7
19	Bell Pepper Bacterial Spot	797	190	10
20	Bell Pepper healthy	1183	266	29
21	Potato Early Blight	800	190	10
22	Potato healthy	121	21	10
23	Potato Late Blight	800	190	10
24	Raspberry healthy	297	67	7
25	Soybean healthy	4072	917	101
26	Squash Powdery Mildew	1468	331	36
27	Strawberry Healthy	297	67	7
28	Strawberry Leaf Scorch	887	200	22
29	Tomato Bacterial Spot	1702	383	42
30	Tomato Early Blight	800	199	10
31	Tomato Late Blight	1273	287	31
32	Tomato Leaf Mold	762	171	19
33	Tomato Septoria Leaf Spot	1527	353	39
34	Tomato Two Spotted Spider Mite	1341	302	33
35	Tomato Target Spot	1132	253	28
36	Tomato Mosaic Virus	298	68	7
37	Tomato Yellow Leaf Curl Virus	4286	964	107
38	Tomato healty	1273	287	31

Table 2: PlantVillage Dataset [1]

tional information on the disease of the plant.

- Strong labeling: the expert in the field identifies the disease and the infected regions of the plant. This method requires a high cost, and a lot of time, the expert uses the labeling software to annotate the data from where the labeled data is not available in large quantities.

– The preprocessing: Is used to normalize the images of the data set. The most used techniques are resizing and average subtraction. Resizing is used to convert the input images to the normal network input layer size. However, mean subtraction is used to center data which speeds up optimization through optimization algorithms. We also segregate training data (80% data), validation data (20% data), and test data (10% validations data) like in www.tensorflow.org where they have used 10% validation data to do image classification.

5.2 Training phase

It requires features which are able to train the convolutional neural network, we retrieve data labeled by domain experts. We go through the convolutional neural network which extracts the features and performs features flattening, then we train the convolutional neural network with its characteristics (we note that these different operations are done automatically). To get these features, we pass images to the feature extraction layer, the segmentation and feature extraction steps are often very difficult to explain in neural networks because it is done in hidden layers. As said YOSHUA Bengio [2] [3] August/22/2017, at the summer school in deep learning «because the meaning is hidden something happens which is left free to learning, so the learning decides what suits it to learn the function given to it, and it discovers intermediate functions which allow going from input to output and perform a calculation that can be complex». We will describe the segmentation, feature extraction:

[2] https://www.youtube.com/watch?v=R-TZPo$_X Zoo$

[3] Accessed 05/30/2021

– **Segmentation:** Segmentation consists of the localization of infected regions in the leaf in the context of the DL. To solve this problem, DL segmentation algorithms can be used to split the image into several parts in an unsupervised way. The segmentation of the regions is generally done using an activation function or thresholds or even a transfer function which introduces a non-linearity.

– **Feature extraction:** This step is very important since we could not perform a good classification if the features are not well extracted. The extraction of features in a convolutional neural network is done automatically. The neural network uses filters to be able to extract them and these filters move over the images that constitute our input data and make a point product with its subregion.

– **Training:** the training step allows the convolutional neural network to learn good representations. For this, we apply supervised learning algorithms which consist of using training data and validation, which is labeled. After extracting the characteristics of the training data automatically, we flatten and pass them to the classification layer. Moreover, we find out and use validation data if the convolutional neural network has learned well if not, we go directly to optimization.
– **Optimization:** Optimization allows us to improve the performance of the model during the training stage of the convolutional neural network. We apply the optimization algorithms to improve the performance of the network when the model has learned poorly or learned too much. The most popular algorithms actively used include SGD, SGD+momentum, RMSprop, RMSprop+momentum, AdaDelta, and Adam as listed in [3]. When our network has learned by heart, we say that it has learned too much and it predicts with noise, hence over-fitting. If the network has learned badly, it cannot predict certain values, hence the use of optimization algorithms.

5.3 Deployment phase

Before deploying the model on several platforms (Windows, Linux, MacOs, website, . . . etc.) we will first perform tests with test data.

– **Classification:** At the classification stage, we will test our model using test images that we will pass as input to the model and at the output, the model will tell us to which disease class each image belongs.
– **deployment on different platforms :)** To deploy we will use Tensorflow js to convert and deploy the preformed models in the browser . HTML5 we will design web pages, CSS3 we will present HTML documents. JavaScript we will build interactive web pages, and PHP to produce dynamic web pages. Laragon server or Xampp server to deploy locally to see if the software works well. Git is going to allow us to manage decentralized builds and Flutter, which is Google's UI toolkit, is going to allow us to build beautiful, natively compiled apps for mobile devices, web, and desktops. integrated devices.

To implement the model, we chose the material and flexible tools for the implementation of the model using the collected data (PlantVillage). We carried out the data preparation phase, the training phase and finally the deployment phase. We then deployed the model on several platforms such as Windows, Linux, mobile and on the web. We appreciate the results of the implementation in the next chapter where we justified the choice of the best architecture for the deployment of the model.

6 Results

We used **80%** (43,444 images) of the model training data, 20 percent (10,861 images) of the validation data. **20%** (2,172 images) of this validation data was used as test data. We used the GoogleNet and AlexNet architectures and added the GAP in GoogleNet and the LRN in AlexNet to train the model by training over several epochs. In each epoch, we find that the learning curve increases as the validation rate, precision, recall, and F1-measure increase and the loss rate decreases (precision, recall and F1-measure generalize all the model performance measures). We will detail the results in the following. We note that :
- tp = number of true positives
- fp = number of false positives
- fn = number of false negatives
- Precision = tp / (tp + fp) is intuitively the ability of the classifier not to label as positive a sample that is negative (model accuracy measure), $multiclassprecision = \sum_{i=1}^{n} précision_i/n$;

- Recall = tp / (tp + fn) is intuitively the ability of the classifier to find all positive samples (an exhaustive measure of the model), $multiclassrecall = \sum_{i=1}^{n} rappel_i/n$;

- F1-measure = 2*(precision*recall) / (precision + recall) can be interpreted as a weighted harmonic mean of precision and recall, where an F-measure score reaches its best value at 1 and the worst score at 0, $F1 - mesuremulti - class = 2 * (multiclassprecision * multiclassrecall)/(multiclassprecision + multiclassrecall)$,
- Accuracy = (Correct predictions / Total predictions) = (tp+tn) /(tp+tn+fp+fn) is a metric that summarizes the performance of a classification model.

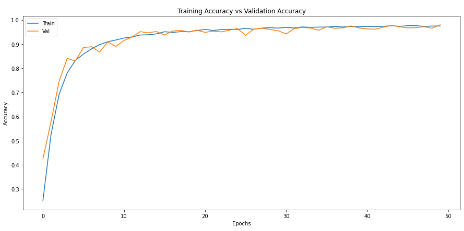
- Error rate (losses) = (Incorrect predictions / Total predictions) = (fp + fn)/(tp + fp + fn + tn)

We used Global Average Pooling (GAP) in the GooGleNet architecture which was proposed to replace the multi-layer perceptron part. The idea is to generate a feature map for each corresponding category. Instead of adding a perceptron after the feature map, we take the average of each one and the result is fitted into the softmax function. An advantage of GAP over perceptron layers is that there are no parameters to train, therefore overlearning is avoided.

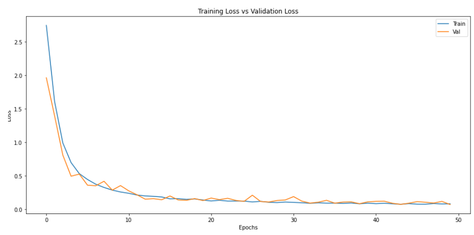
With the AlexNet architecture, we have used another new concept, the normalization of the local response, which is a new trick to make training smooth and stable. Unlike sigmoid and tanh activations, ReLU has an unlimited response in the positive edge. Thus, activations can explode and interfere with the training procedure as the training continues. This is why some kind of normalization is necessary. Local response normalization attempts to balance the activation values in a pixel among neighboring channels. Each activation is divided by the sum of the squares of the activations of the neighboring channels at the same pixel location. This idea is borrowed from the neurobiological term lateral inhibition and was first described in detail in the original AlexNet article. Table 3 summarises the results.

Model	**Optimizer**	**epoch**	**rate of classification**	**Precision**	**Rappel**	**F1 - measure**	**loss**	**Training time**
GoogleNet	Adam	50	0,9797	0,9821	0,9780	0,9800	0,0203	7h
AlexNet	Adam	50	0.9465	0.9577	0.9359	0.9466	0.1799	6h

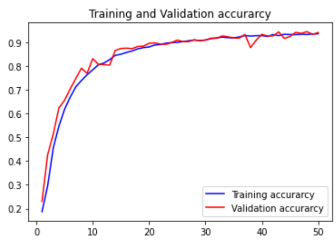
Table 3: Implementation results of the GoogleNet and AlexNet architectures



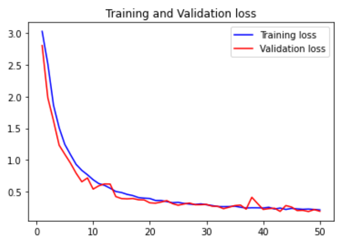
(a) Validation and classification rate of GoogleNet



(b) Loss of validation and training of GoogleNet



(c) Validation and classification rate of AlexNet



(d) Loss of validation and training of AlexNet

We observe the results of the application after deploying the model locally, on mobile, Windows, Linux, and the web through tests(https://pants-deseases.herokuapp.com/). The following figure 6 shows an example of the deployment of the model trained using Tensorflow js. We notice that this model is trained with Tensorflow. We observe that this potato leaf is diseased. We used plants images that we trained the model with in order to understand, otherwise it might have a problem with detection outside the distribution of convolutional neural networks.

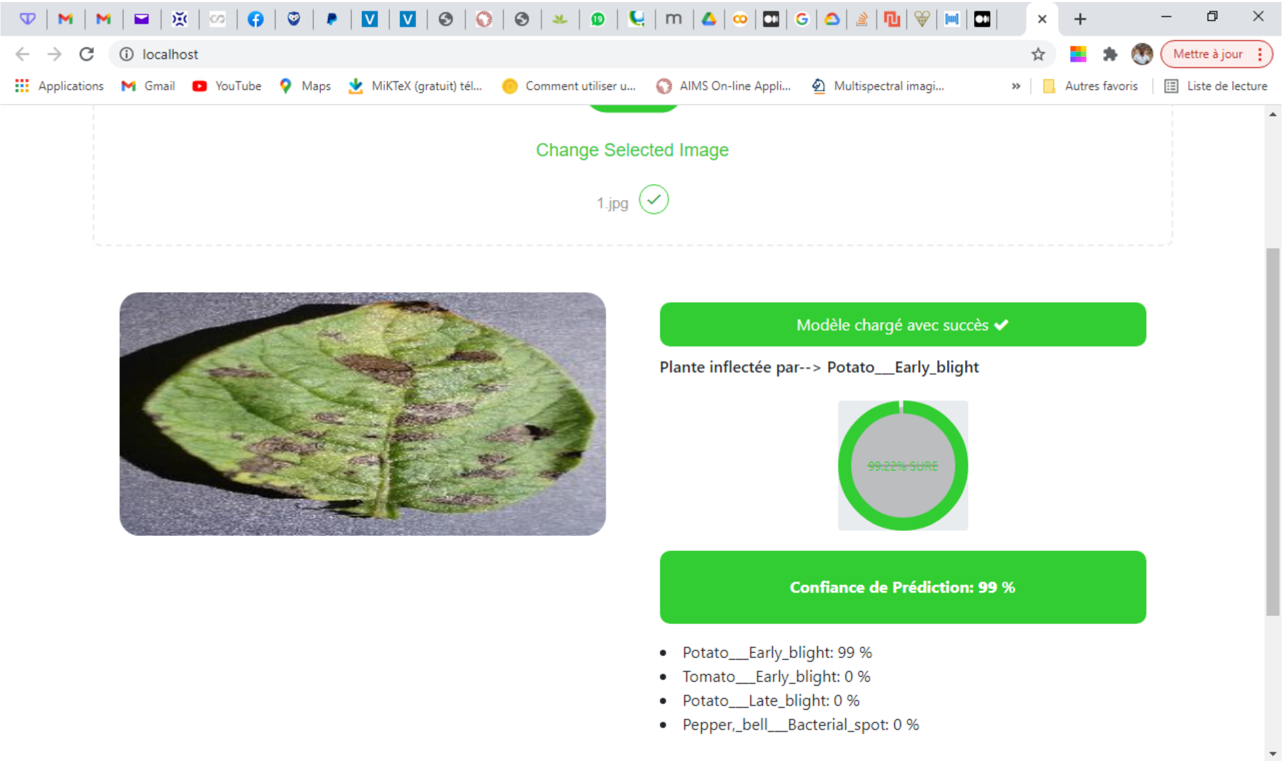


Fig. 6: Example of a test

6.1 Comparison of results

We chose GoogeNet to deploy the solution because it produces better result compared to AlexNet, and we compared it to the models used in the state of the art. We note that with the LRN and the GAP, we had smooth and stable training compared to [6] and a model that learns well (see table 4). Based on these documents, we have chosen approach and technique.

Model	**Method extraction characteristics**	**rate of classification**
ANN(artificial neural network)[2]	artisanal (GLCM)	94,67%
KNN [7]	artisanal (GLCM)	96,76%
CNN[4][6]	automatic (Convolution + pooling)	96,70 %/ 99, 89%
GoogleNet	automatic (Convolution + pooling)	97,97%
AlexNet	automatic(Convolution + pooling)	94,65 %

Table 4: Comparison of results

We analyzed our results based on the AlexNet architecture where we added the LRN and the GoogleNet architecture. We saw that GoogleNet has a good precision compared to AlexNet. We compared our results with the state of the art after the comparison and validated the choice of our architecture (GoogleNet) which is better compared to others because the training is smooth and stable. It trains well the model when using GAP, and GoogleNet architecture already has LRN. So we choose GoogleNet for the deployment of the model as it has shown us a good performance.

CONCLUSION AND PERSPECTIVES

Global balance sheet

We used Global Average Pooling (GAP) and Local Response Normalization (LRN) to avoid overlearning in GoogleNet and AlexNet architectures to make training smooth and stable. We built a plant disease detection and classification model based on Deep Learning; a tool easy to use by families. We analyzed and compared existing work for automatic plant disease detection, some of which previously used traditional methods to extract features. We used a method that allows us to extract automatically using these architectures (AlexNet and GoogleNet) and we used a public dataset (PlantVillage) of plant diseases to train the model. The results clearly shown that we have achieved an precision of 98.21% with the GoogleNet architecture and 95.77% with the AlexNet architecture. The GoogleNet architecture allowed us to deploy the trained model on several platforms. As a limitation of the system, we have taken into account some aspects whose cultures vary according to the regions. The problem of detection out of distribution in the neural networks remains to be solved and the model has not been sufficiently trained because the time allocated in the Google colab platform did not allow us to do so.

perspectives

Our future work will focus on Transfer Learning. This will allow us to use the model we have built on plants coming from different regions. In addition, we will solve the problem of out-of-distribution detection in neural networks for out-of-distribution images.

7 Conflicts of interests

The authors declare that they have no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

8 DAS

Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

References

1. Valantine Achancho. Revue et analyse des stratégies nationales d'investissements et des politiques agricoles en afrique du centre: Cas du cameroun. *Reconstruire le potentiel alimentaire de l'Afrique de l'Ouest, FAO/FIDA. Disponible en ligne à l'adresse suivante: http://www. fao. org/docrep/018/i3222f/i3222f04. pdf.(Consulté le 15/02/2018)*, 2013.
2. Heba Al-Hiary, Sulieman Bani-Ahmad, M Reyalat, Malik Braik, and Zainab Alrahamneh. Fast and accurate detection and classification of plant diseases. *International Journal of Computer Applications*, 17(1):31–38, 2011.
3. C Aggarwal Charu. *Neural Networks and Deep Learning: A Textbook.* Spinger, 2018.
4. Chad DeChant, Tyr Wiesner-Hanks, Siyuan Chen, Ethan L Stewart, Jason Yosinski, Michael A Gore, Rebecca J Nelson, and Hod Lipson. Automated identification of northern leaf blight-infected maize plants from field imagery using deep learning. *Phytopathology*, 107(11):1426–1432, 2017.
5. Olivier Ezratty. *Les usages de l'intelligence artificielle.* 2017.
6. Mercelin Francis and C Deisy. Disease detection and classification in agricultural plants using convolutional neural networks—a visual understanding. In *2019 6th International Conference on Signal Processing and Integrated Networks (SPIN)*, pages 1063–1068. IEEE, 2019.
7. Eftekhar Hossain, Md Farhad Hossain, and Mohammad Anisur Rahaman. A color and texture based approach for the detection and classification of plant leaf disease using knn classifier. In *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, pages 1–6. IEEE, 2019.
8. Andreas Kamilaris and Francesc X Prenafeta-Boldú. Deep learning in agriculture: A survey. *Computers and electronics in agriculture*, 147:70–90, 2018.
9. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.
10. Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.
11. Muhammad Hammad Saleem, Johan Potgieter, and Khalid Mahmood Arif. Plant disease detection and classification by deep learning. *Plants*, 8(11):468, 2019.
12. Abebe Shimeles, Audrey Verdier-Chouchane, and Amadou Boly. *Building a resilient and sustainable agriculture in sub-Saharan Africa.* Springer Nature, 2018.
13. and . *Contributions aux méthodes d'extraction de caractéristiques discriminantes pour la classification des images médicales.* PhD thesis, Ziane Achour University of Djelfa, 2019.