University of Manouba
National School of Computer Sciences

Internship Report of Enterprise Immersion

Subject

Plant Disease Detection Using Deep Learning

Author :

Ahmed Gharbi

at

I.T.GRAPES
ADVANCED INTEGRATED SOLUTIONS

Host Organism **: I.T GRAPES**

Responsible **: Mr Mestiri Taher**

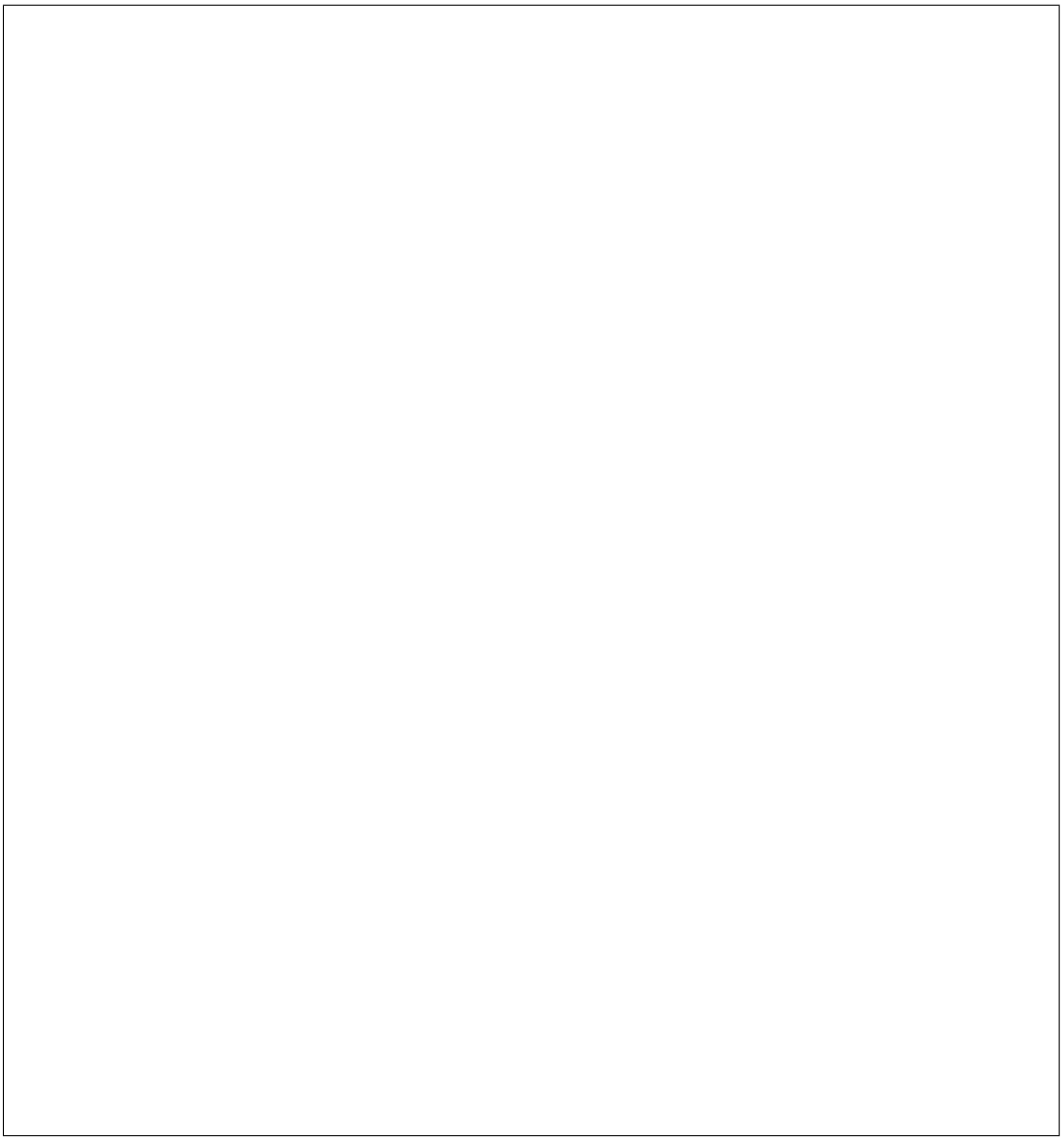
Company Supervisor: **: Mr Lahouar Hedi**

Address: **: North Star, Block A, A26 App, North Urban Center, Tunis 1003**

Phone: **: (+216) 36 363 030**

Academic Year : 2018 /2019

Signature



Acknowledgement

In conducting this report, we have received meaningful assistance from many people which we would like to put on record here with deep gratitude and great pleasure.

First and foremost, we express our sincere gratitude to our supervisor Mr Lahouar Hedi, who extended his complete support and helped to make us deliver our best, as well as our friends for their guidance and valuable pieces of advice during the different phases of the elaboration of the project. Simply put, we could not have done this work without the amount of help and trust we received from them. We appreciate their patience in checking and reviewing our work. We would also like to thank all of our teachers at the National School of Computer Science for their continuous help and treasurable training during our study years.

Finally, special thanks to the jury members who honored us by examining and evaluating this modest contribution.

Contents

List of Figures

List of Tables

List of Acronyms

CNN *Convolutional Neural Network*

AI *Artificial Intelligence*

CAM *Class Activation Map*

General Introduction

According to the Food and Agriculture organization of the United Nations (UN), transboundary plant pests and diseases affect food crops, causing significant losses to farmers and threatening food security. The spread of transboundary plant pests and diseases has increased dramatically in recent years. Globalization, trade and climate change, as well as reduced resilience in production systems due to decades of agricultural intensification, have all played a part. Transboundary plant pests and diseases can easily spread to several countries and reach epidemic proportions. Outbreaks and upsurges can cause huge losses to crops and pastures, threatening the livelihoods of vulnerable farmers and the food and nutrition security of millions at a time.

The Tunisian farmers are no exception. Hundreds of thousands of Tunisian dinars are spent every year by the Tunisian farmers for plant diseases diagnosis and treatment. Experts must move to farms to examine the crops. This process is not only a very costly process for the farmers, but also a time consuming one. Hence, it seemed essential to take a revolutionary step forward in the automation of this process.

Today, AI is at the forefront in changing the world and the way we live. It is impacting positively nearly all aspects of society: the labor market, transportation, healthcare, education and banking. AI did not fail to improve the sector of agriculture as well. It has succeeded to progressively move forward toward the automation of the means to control and monitor the environmental parameters of a farm and to react automatically and appropriately to any variation of these parameters with a minimum level of human intervention.

Within this context, our project focuses on one of the most important applications of deep learning, which is image recognition used to detect plant diseases.

In the first chapter of this report, we will describe the general context of our project. Throughout the second chapter, we will analyse the requirements of this project. In the third chapter, we describe briefly the design of the application. In the fourth chapter, we will define some key concepts and we will establish a preliminary study. The fifth chapter is dedicated to expose the theory behind the algorithms we are using, while in the sixth chapter we discuss the results of our work. Finally, we conclude our work by giving future

prospects and potential upgrades for our project.

Chapter 1

Context of The Project

Introduction

In this chapter, we introduce the circumstances in which our project has been carried out. The first section is a presentation of the host organization. In the second section, we state the problem and the project's goals.

1.1 Presentation of the Hosting Company

This section is dedicated to present our internship hosting company.

1.1.1 I.T GRAPES

I.T.GRAPES is a startup, based in Tunisia, with a number of teams working mainly on developing integrated systems that connects real world to web and mobile.

1.1.2 The SEABEX Project

SEABEX is an e-monitoring and smart automation system that helps farmers find the right balance of water consumption needed to get the better quality and quantity production. SEABEX is able to monitor and control in real time the key environmental parameters of a farm, interacts and reacts autonomously and appropriately to any environmental parameter variation with minimum level of human intervention. SEABEX's ambition is to rationalize water resource expenditures for agriculture and improve farming profitability. The list of international awards for this project is already very impressive: Let's mention, for example, the prize for technological innovation obtained in January 2017 at the MBC AL AMAL event in Dubai or the Ye! Star - Female Entrepreneur Award for Amira Cheniour in May 2017 in Germany at the ceremony The Global Inclusion Awards 2017. And on September 6, 2018, the SEABEX project was rewarded with the 1st Orange prize of

the 2018 Social Entrepreneur in Tunisia and thus won its place for the international final which took place on November 14th in South Africa.

1.2 Project Overview

In this section we are going to present the internship context, the problem description and the goal of our project.

1.2.1 Scope

This document is the internship report of enterprise immersion project entitled " Plant Disease Detection using Deep Learning". Throughout this document, we detail the process we went through to achieve our purpose. This project is a requirement for the second year in the National School of Computer Science. It has been achieved during an internship at I.T GRAPES that lasted 2 months starting from 24 June 2019 to 17 August 2019.

1.2.2 Problematic

The Seabex project seeks to expand the services it offers. One major issue that farmers face in all over the world, is plant diseases and their identification. When a plant get infected with a certain disease, farmers tend to call experts to come and identify the disease. This procedure can be expensive and time-consuming for farmers, thus there is a growing need to make the disease detection automatic.

1.2.3 Objectives and Contributions

It's almost impossible to design a fully working application that offers high accuracy and to prepare it for production in our two-months internship, especially with the lack of data and resources we're facing. Thus, this internship project is more of a prototype designed to demonstrate the feasibility of the desired goal. Our main goals during this internship are:

- designing and implementing a model capable of detecting the type of the plant and its disease when given an image of a plant leaf.
- Deploy the model in a web application

Conclusion

Throughout this chapter, we put under the spotlight our project's context. We have presented the host company and we have clarified the problematic and goals behind this

work. In the next chapter we will be analyzing the application's requirements.

Chapter 2

Requirements Analysis and Specification

Introduction

In this chapter we analyse and specify the functional and non-functional requirements.

2.1 Requirements Analysis

In this section we present, in the first place, the actors, and in the second place the functional and non-functional requirements.

2.1.1 Actors

The main actors of the system are the farmers. The farmers have to upload an image of the diseased leaf to our system so that they get the result of the prediction.

2.1.2 Functional Requirements

The system must enable the user to:

- upload an image of a leaf
- predict the type of the plant
- predict the disease of the plant
- indicate that the plant is healthy if that's the case

2.1.3 Non functional requirements

Non-functional requirements cover all the remaining requirements which are not covered by the functional requirements. They specify criteria that judge the operation of a system, rather than specific behaviours

- Performance: The app must provide the services in a reasonable amount of time and the results of the prediction must be accurate.
- Reliability: The application must be functional regardless of any circumstances.

2.2 Requirement Specification

In this section we present the use case diagram.

2.2.1 Use Case Diagrams

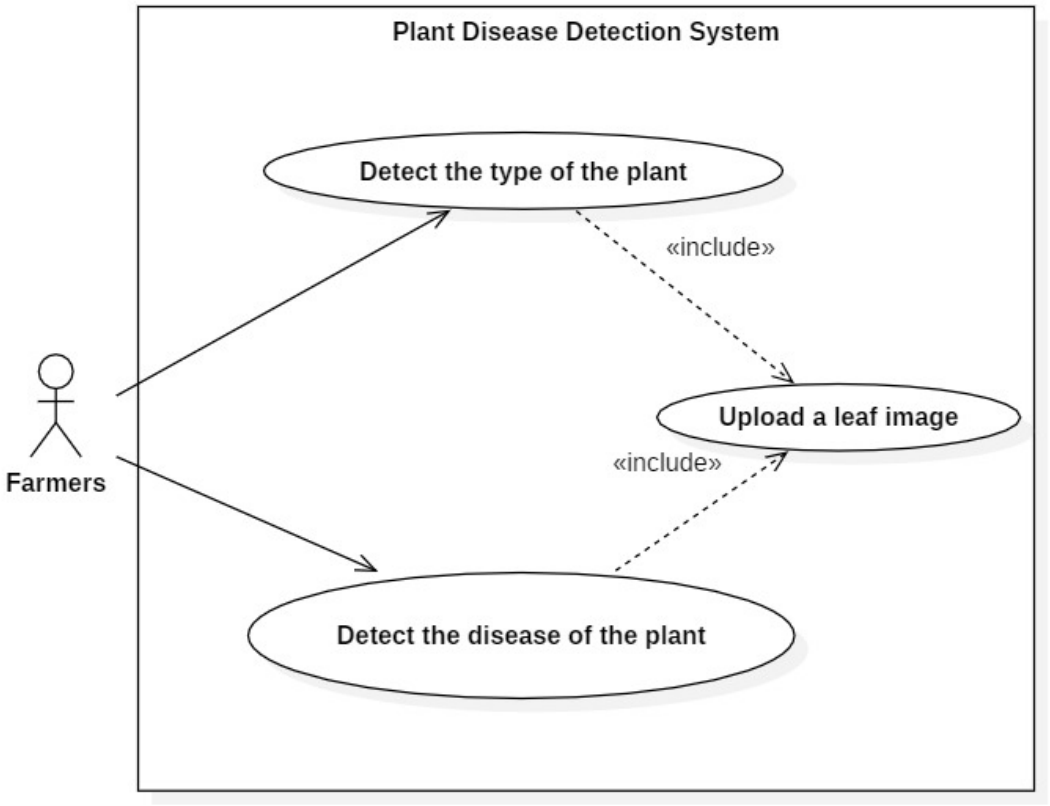


Figure 2.1: Use case diagram

Conclusion

Throughout this chapter, we identified the actors, specified the functional and non functional requirements of our application and we presented the use cases. In the next chapter, we will present briefly the design of the application.

Chapter 3

Design

Introduction

To achieve the desired results described in the requirements analysis and specification, we will discuss the global design of the application.

3.1 Global Design

In this section we will only discuss the physical architecture of the application because the web application we've built is so basic and no particular logical architecture was used.

3.1.1 Physical Architecture

During this project, we used the classical 3-tier architecture, consisting of the client tier, the application tier and the database tier. Our work consisted only of setting the application server. As for the database server, the engineers who will take over the project will take care of it. The idea is to build a database server that will store all the leaf images sent by the users. Once the amount of images stored becomes reasonably large, the images will be examined by an expert who will classify them into their right category, and the training of the model will be launched again, enabling us to improve the accuracy of the model. Figure 3.1 summarizes and explains the architecture we used.

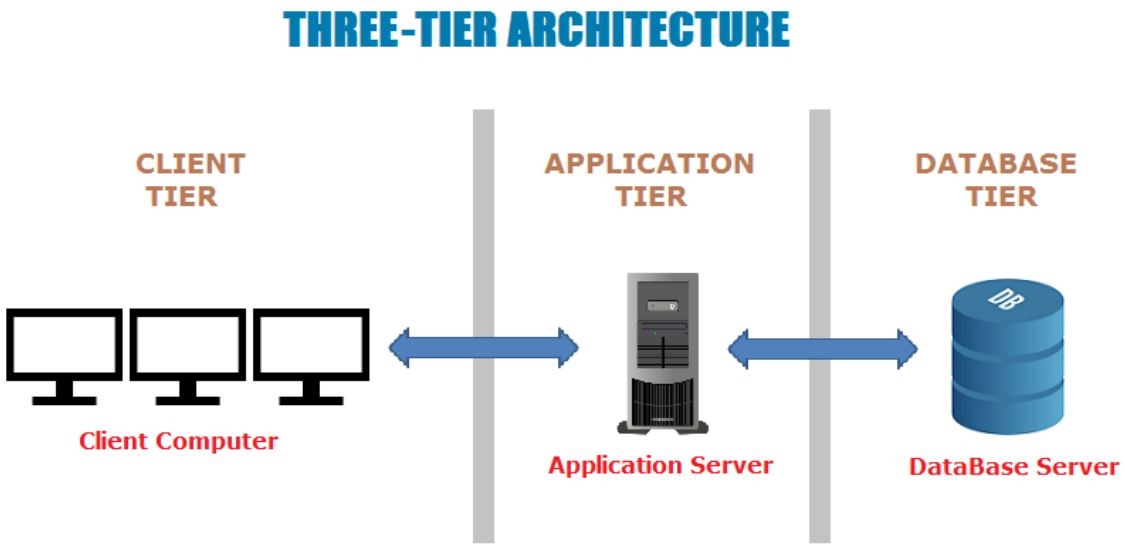


Figure 3.1: Physical architecture [URL1]

Conclusion

Throughout this chapter, we have described the physical architecture. In the next chapter, we will present a preliminary study to understand the key concepts of our project.

Chapter 4

Preliminary Study

Introduction

In this chapter, in a first time, we introduce the main concepts and their definitions, in a second time, we clarify and define the adopted approach we followed.

4.1 General Concepts

In this section, we define the main terms we are going to be using through the whole repport.

4.1.1 Image Recognition

Image recognition is a term for computer technologies that can recognize certain people, animals, objects or other targeted subjects through the use of algorithms and machine learning concepts. The term "image recognition" is connected to "computer vision," which is an overarching label for the process of training computers to "see" like humans, and "image processing," which is a catch-all term for computers doing intensive work on image data.

4.1.2 Deep Learning

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks. Deep neural network is a neural network having more than one block of hidden layers which makes the algorithm more robust and complex capable of making accurate predictions.

4.2 Adopted Approach

This section is dedicated to explain briefly the approach we adopted to achieve our goal.

4.2.1 Convolutional Neural Network

To tackle the classification problem we're facing, we used Convolutional Neural Networks (CNN) which is basically a particular architecture of an artificial neural network. In the recent years, CNNs have revolutionised the computer vision field, proving to be one of the best models currently used, offering outstanding results in many problems. We are going to present the basic building blocks of CNNs. This will allow us to understand the complex model we used that we will discuss later in the Theoretical Study chapter. A classic CNN architecture is usually composed of three types of layers:

- Convolutional layer: The convolution layer comprises of a set of independent filters. A filter can be considered as a matrix of values, that when convolved with the image, we end up with feature maps, which is basically a new matrix whose values indicate features that had been detected. Such features can be vertical lines, horizontal lines, colors... the deeper the convolutional layer is, the more is it capable of detecting more complex features.

- Pooling layer: A pooling layer is another building block of a CNN. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. Pooling layer operates on each feature map independently.

- Fully connected layer: A fully connected layer is usually found at the end of the model. Its function is to use features learnt by the convolutional layers to make predictions.

4.2.2 Transfer Learning

Transfer learning is a deep learning technique often used to make use of pre-trained, state of the art, deep learning models trained by experts, big corporations and researchers. The training of models having a complex architecture is computationally very intensive, and usually take multiple weeks when training on huge datasets. But the features learnt by the earlier layers turn out to be pretty generic in nature for multiple image classification problems, and hence in principle can be easily reused when attempting an image classification problem on a different dataset of images spread over completely different classes. As shown in Figure 2.1, the early layers of a deep convolutional network are responsible for extracting low level features (ex: vertical lines), the deeper the layer is, the more complex

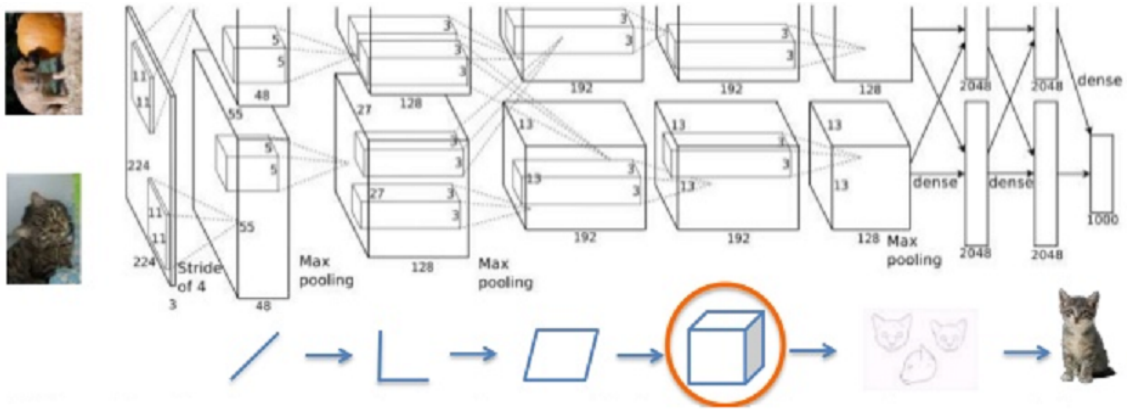


Figure 4.1: Transfer Learning

the extracted features get. The last few layers, commonly referred to as Dense layers, are responsible for recognizing the target object. The idea behind transfer learning is to load a pre-trained model that had achieved excellent performance in a classification task, we freeze the weights of the early layers, we delete the last layers responsible for the object recognition, replace them with new layers, and train the model on our dataset, updating only the weights of the added layers. This technique allows us to gain in terms of computational power and to reduce the number of parameters to update.

Conclusion

In this chapter, we defined the main concepts we're using in deep learning. In the next chapter, we will discuss the details of the implemented algorithms.

Chapter 5

Theoretical study

Introduction

In this chapter, first, we will introduce the model we used, its architecture and the theory behind it. Second, we will explain the theory underlying some tools we implemented to interpret the neural network.

5.1 GoogLeNet Network

GoogLeNet (also called the Inception Network) is a deep convolutional neural network developed by google. It achieved the new state of the art in the ImageNet Large-Scale Visual Recognition Challenge 2014.

5.1.1 The Inception Network motivation

When designing a CNN, we face many choices. We have to decide whether are we going to use 3x3 or 5x5 conv layers, or maybe a maxpooling layer. The inception module suggests that we should use them all and let the neural network learn and decide which ones it wants to use most. Intuitively, we may think that using all sorts of layers will undoubtedly increase the number of parameters, which it will. However, the Inception module uses what we call a 1x1 convolutional layers that helps decreasing the number of parameters significantly without hurting the performance. The figure 5.1 explains how a 1x1 conv layer could reduce the shape of the volume of filters from 56x56x64 to 56x56x32.

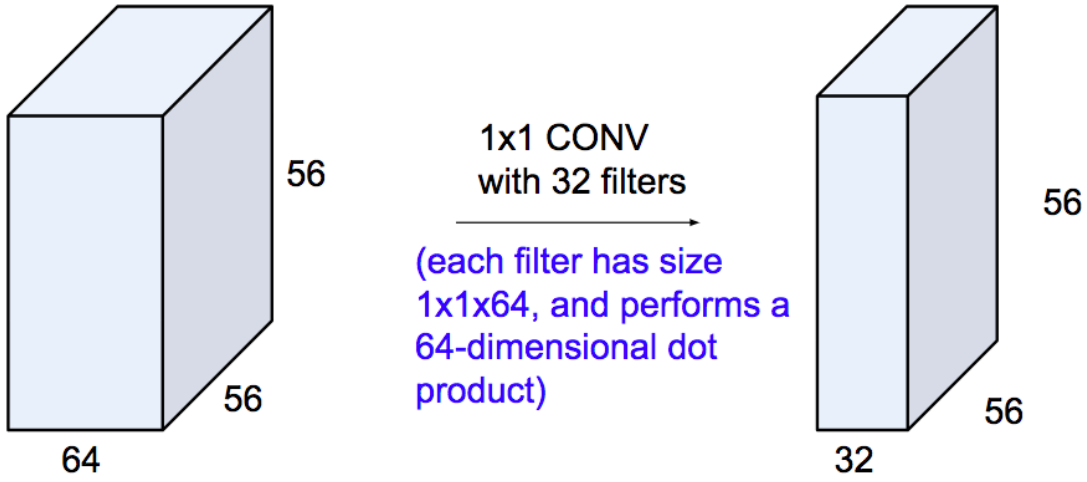


Figure 5.1: 1x1 Convolution [URL2]

5.1.2 The Inception Module

The Inception module focuses on the idea of using all sorts of filters and then concatenate them as shown below :

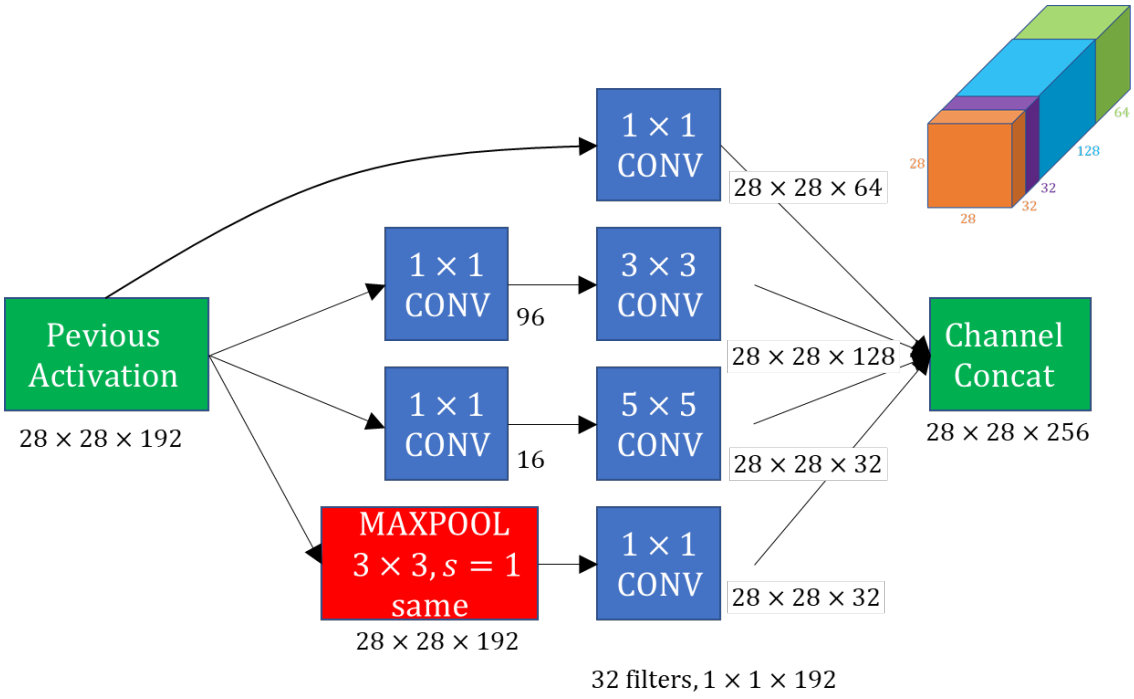


Figure 5.2: The Inception module [URL3]

In the figure 5.2 above, we should note the use of 1x1 convolution layers before feeding the output of the "Previous Activation" to the nxn convolutional layers. For instance, before feeding the output of "Previous Activation" to the 3x3 convolutional layers, we apply 96 filters 1x1 (96 filters, each one of these filters is of dimension 1x1x192) . The output of the 1x1 convolution is of dimension 28x28x96, thus reducing significantly the number of parameters before going through the 3x3 conv (from 28x28x192 to 28x28x96). The module combine the use of different filters sizes (1x1, 3x3 and 5x5) and uses Maxpooling layers as well. In the end, all of the volumes are concatenated and fed to the next Inception module.

5.1.3 The Inception Network Architecture

The inception model is composed mainly of multiple inception modules. The figure below displays the whole architecture of the Inception Network :

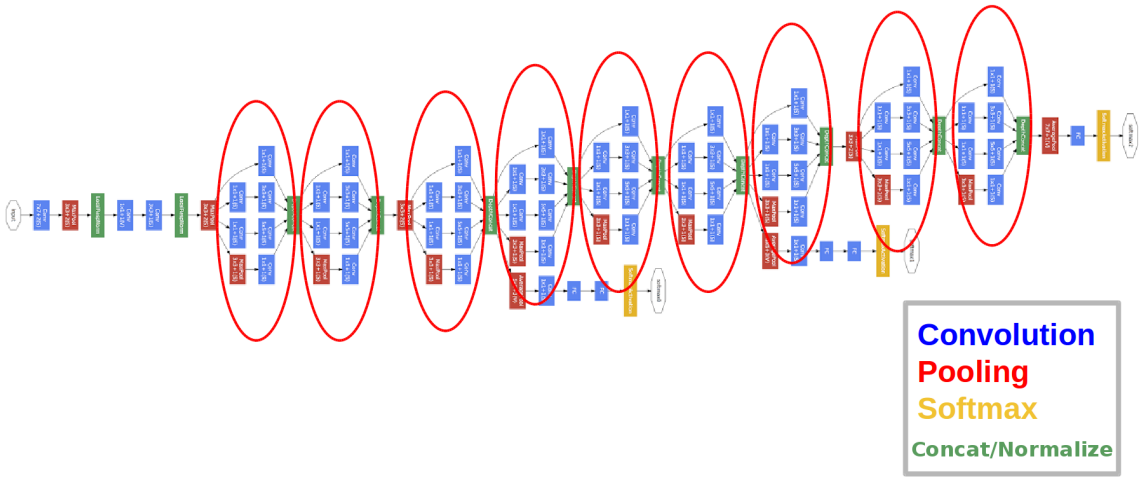


Figure 5.3: The Inception network [URL4]

- The circled areas represent Inception modules.
- Sometimes a Max-Pool block is used before the inception module to reduce the dimensions of the inputs.
- There are 3 Softmax branches at different positions to push the network toward its goal and helps to ensure that the intermediate features are good enough to the network to learn and it turns out that softmax0 and sofmax1 gives regularization effect.
- Another change that GoogLeNet made, was to replace the fully-connected layers at the end with a simple global average pooling which averages out the channel values across the 2D feature map, after the last convolutional layer. This drastically

reduces the total number of parameters. This can be understood from AlexNet (a convolutional neural network, winner of the ImageNet challenge in 2012) , where the fully connected layers contain approx. 90% of parameters. Use of a large network width and depth allows GoogLeNet to remove the FC layers without affecting the accuracy.

5.2 Interpreting the Neural Network

To interpret the neural network, we implemented some visualization tools that allow us to understand more in depth how the learned filters work.

5.2.1 Class Activation Map

Class Activation Map (CAM) is a visualization technique that allows us to find out what regions in an image the filters are looking at when processing an image for prediction. Using CAM requires that the model disposes of a Global Average Pooling layer at its end. The output of the last convolutional layers is a 3-dimensional tensor often called feature map. This tensor is a representation of the features learned. The Global Average Pooling layer reduces each matrix of this tensor to a single value. The idea behind CAM is to get weighted features, which means, to get a value of how important a feature was in the process of predicting one class. The output of the Global Average Pooling layer is a set of values, each value represents a certain feature. Each feature is connected to the next dense layer responsible for making predictions through a weight matrix. Let's say for instance that we want to know what regions of the input image the filters are focusing on when calculating the prediction for a class A. Each feature (the output of the Global Average Pooling) has a weight used to calculate the prediction for the class A. By multiplying this weight by the value of the feature, we get an idea of how useful that particular feature was in making the prediction. By using the shape of the original image, the feature map and the obtained values of this multiplication, we can get what we call an activation map, which is an image that describes what features in an input image are the most relevant in making the prediction.
Figure 5.4 might help illustrate CAM:

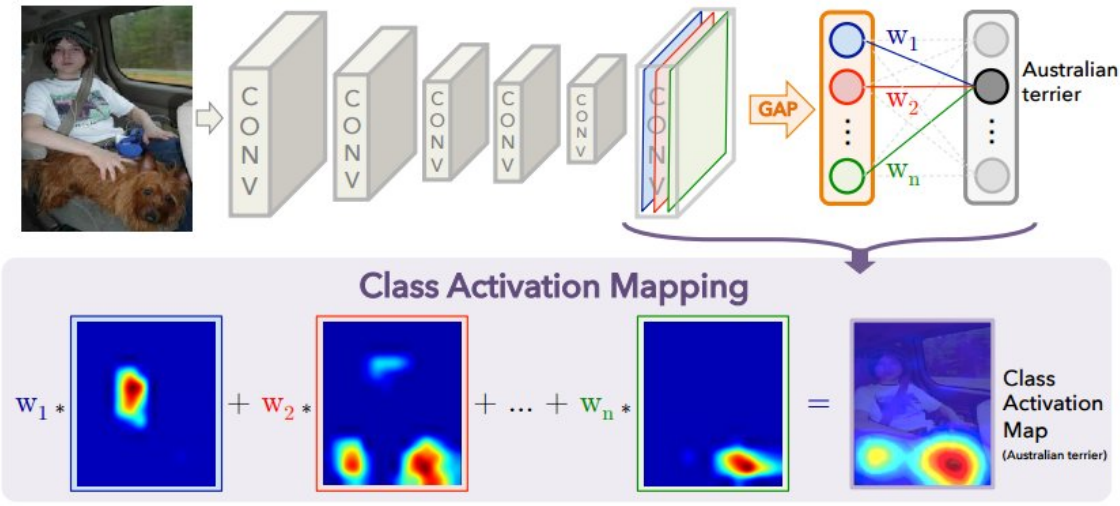


Figure 5.4: Class Activation Mapping [URL5]

5.2.2 Exploration of Convolutional Network Filters: Activation Maximization

The idea behind activation maximization is simple in hindsight : Generate an input image that maximizes the filter output activations. We compute the gradient of the activation maximization loss with respect to the input. Activation maximization loss simply outputs high values for large filter activations (we are maximizing losses during gradient ascent iterations). This allows us to understand what sort of input patterns activate a particular filter. For example, there could be an eye filter that activates for the presence of eye within the input image. The first layers basically just encode direction and color. These direction and color filters then get combined into basic grid and spot textures. These textures gradually get combined into increasingly complex patterns. Our convnet filters are uninterpretable to the human eye. In the next chapter we will display the results of the activation maximization technique and try to interpret them.

Conclusion

Throughout the whole chapter we tried to explain in details the architecture of the model we used and the theory underlying it, as well as some techniques we used to interpret the implemented neural network. In the next chapter, we will discuss the implementation details and the achieved results.

Chapter 6

Implementation and Results

Introduction

This chapter allows us to present the achieved work. We begin by listing the technologies and the software environment used throughout the project life cycle. Afterwards, we describe briefly the data, then, we present an overview of our algorithms implementations and discuss later the obtained results. At last, we present the deployment of the model in a web application and show some interfaces.

6.1 Work Environment

In the following sections we describe the hardware and the software environments as well as the technological choices we made to achieve this work.

6.1.1 Hardware Tools

A personal computer with the following specs was used to achieve this work:

- Brand: msi GT62VR 6RD
- Processor: Intel(R) Core(TM) i7-6700HQ CPU @ 2.60 GHz
- GPU: NVIDIA GeForce GTX 1060 10 GO of memory
- Memory: 8 GO of RAM
- Operating system: Windows 10
- Hard Disk : 1 TO

6.1.2 Software Environment

◇ Anaconda : Anaconda is a free and open source distribution of the Python and R programming languages for scientific computing and predictive analytics, that aims

to simplify package management and deployment.

◇ Spyder: is an open source cross-platform integrated development environment (IDE) for scientific programming in the python language.

◇ Python : Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.

◇ TensorFlow: TensorFlow is an open source software library for numerical computation using data-flow graphs.

◇ Keras: Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow

◇ Flask: Flask is an open-source web development framework in Python. Its main purpose is to be lightweight, to keep the flexibility of Python programming, associated with a template system

6.2 Data Overview

In this section,, we are going to give an overview of the data we have used to train and test our neural network, then, we will highlight the main steps we go through to prepare this data.

6.2.1 Data Description

The dataset used in this project is an open-source dataset that falls under the Creative Commons 3.0 Share and Share Alike. (CC BY-SA 3.0) licence.
The data records contain 54,309 images. The images span 14 crop species: Apple, Blueberry, Cherry, Corn, Grape, Orange, Peach, Bell Pepper, Potato, Raspberry, Soybean, Squash, Strawberry, Tomato. It contains images of 17 fungal diseases, 4 bacterial diseases, 2 mold (oomycete) diseases, 2 viral disease, and 1 disease caused by a mite. 12 crop species also have images of healthy leaves that are not visibly affected by a disease.

6.2.2 Data Preprocessing

in this section, we describe the transformations we applied to the data in order to make it usable for the training.

6.2.2.1 Data Normalization

Data normalization is an important step which ensures that each input parameter (pixel, in this case) has a similar data distribution. This makes convergence faster while training the network. Data normalization is done by subtracting the mean from each pixel and then dividing the result by the standard deviation. The distribution of such data would resemble a Gaussian curve centered at zero. For image inputs we need the pixel numbers to be positive, so we chose to scale the normalized data in the range [0,1].

6.2.2.2 Data Augmentation

No data augmentation was applied during training, because it turned out that in many cases, our dataset has multiple images of the same leaf (taken from different orientations). Also, when we applied some minor data augmentation, we have seen a decrease in the testing accuracy, thus we chose not to apply any data augmentation.

6.3 Model Training

In this section, we describe the training process:

- We divided to dataset into 2 sets, a training set containing 80% of the images, and a testing set containing 20% of the images.
- We used the Stochastic Gradient Descent optimizer with the following hyperparameters:
  - Base learning rate: 0.005
  - Momentum: 0.9
  - Weight decay: 0.0005
  - Learning rate policy: Decrease the learning rate by a factor of 10 every 10 epochs
  - Number of epochs: 30
  - Batch size: 24
- In the first phase of training, we only trained the dense layer. The training was quite rapid since the number of trainable parameters is low.
- In a second phase, we fine-tuned the model to augment the testing accuracy. We trained some of the last convolutional layers. The training of the convolutional layers was done after the training of the newly added fully connected layers because if we had trained the convolutional layers and the fully connected layers at the same time, the obtained gradients would have had high values (because the fully connected layers

were not trained at all) and thus we would be violently updating the convolutional layers parameters.

- It should be noted that during the whole project, we trained different models with different architectures. The report only describes the training of the best performing model, in this case, the GoogLeNet network.

6.4 Results

In this section, we describe the achieved results.

6.4.1 Accuracy

The accuracy metric we used is the categorical accuracy. The algorithm Calculates the mean accuracy rate across all predictions for the multiclass classification problem. By the end of the training, we achieved 97.5% accuracy rate.

6.4.2 CAM Results

The images below show some results of applying CAM on our dataset.

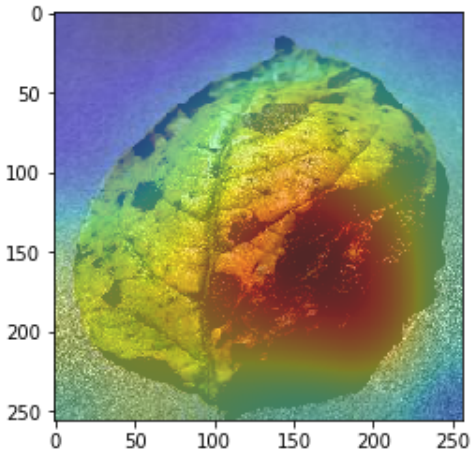


Figure 6.1: CAM

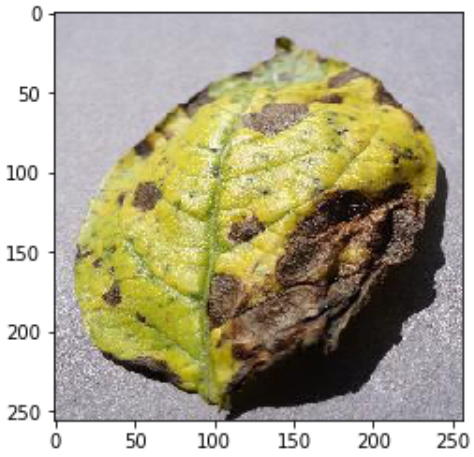


Figure 6.2: Ground Truth

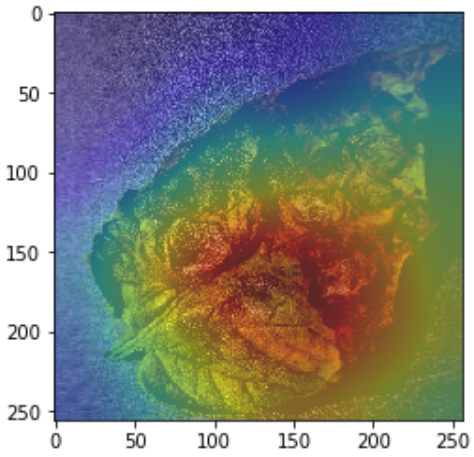


Figure 6.3: CAM

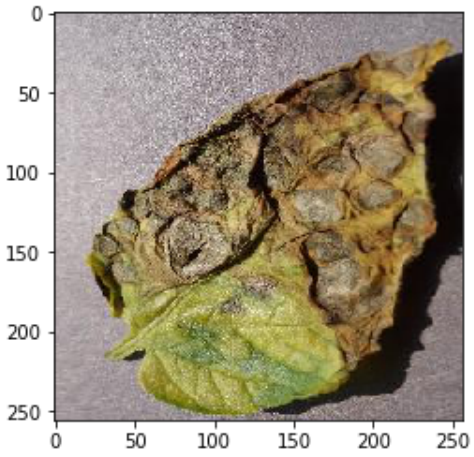


Figure 6.4: Ground Truth

6.5 Activation Maximization Results

The images below are the results of applying the aforementioned activation maximization approach. These images represent the input images that maximizes the activation of the filters.

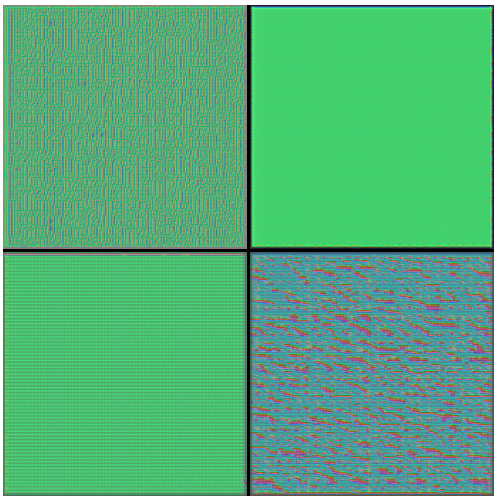


Figure 6.5: Input that maximizes the activation of the first convolutional layer

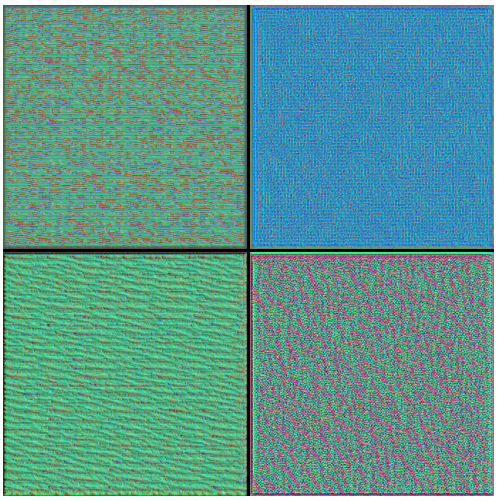


Figure 6.6: Input that maximizes the activation of the second convolutional layer

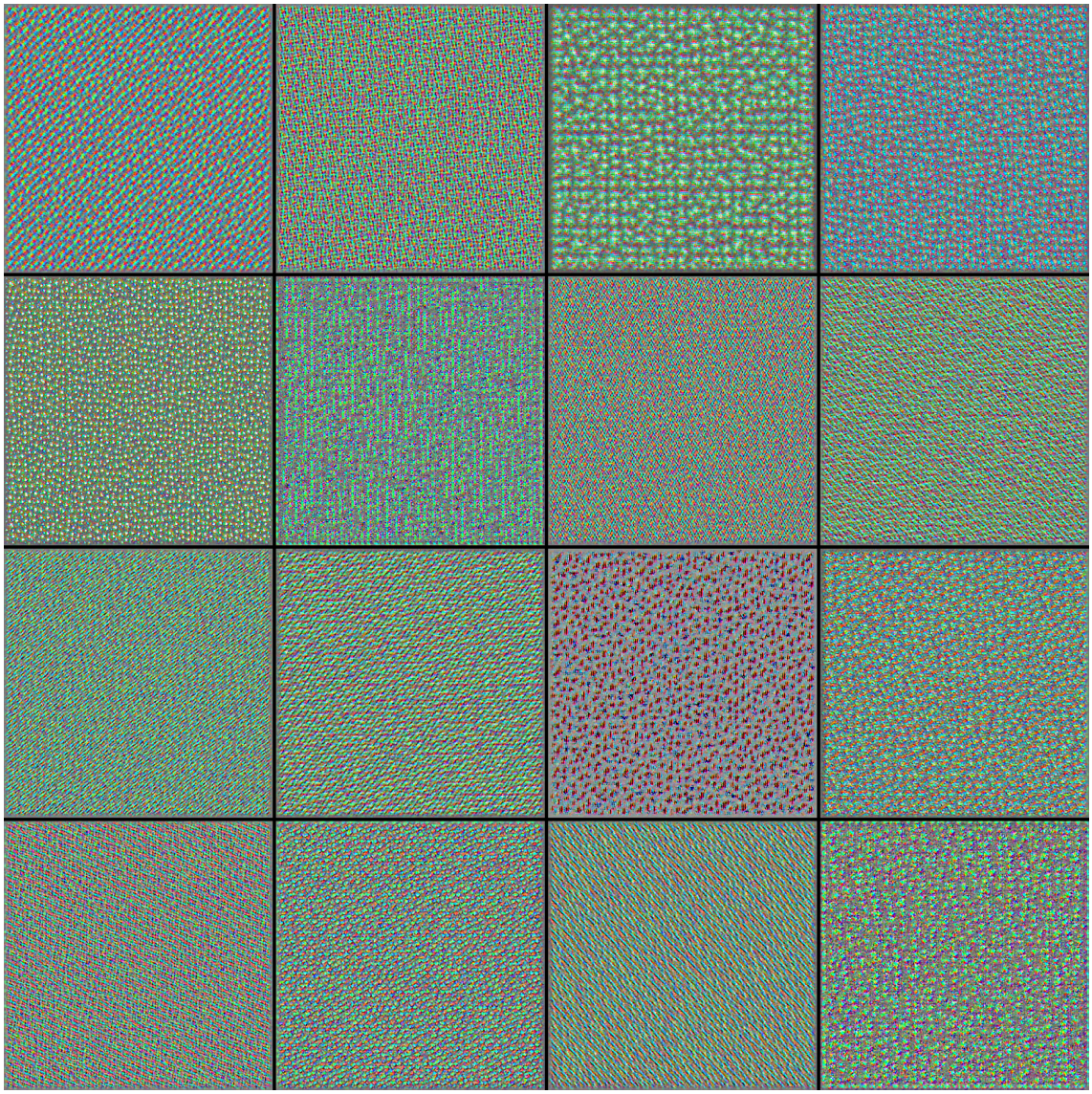


Figure 6.7: Input that maximizes the activation of the third convolutional layer

Our convnet filters are obviously uninterpretable to the human eye. However, that does not mean that convolutional neural networks are a bad tool. They serve their purpose just fine. What it means is that we should refrain from our natural tendency to believe that they "understand", say, the concept of leaf, just because they are able to classify these objects with high accuracy.

6.6 Model Deployment

We built a very basic web application to highlight the value of the work achieved. This web application is intended to be improved by the engineers who will take over the project.

We used Flask to deploy the model into a web application. The following images are some interfaces showing the process the user goes through to make a prediction.

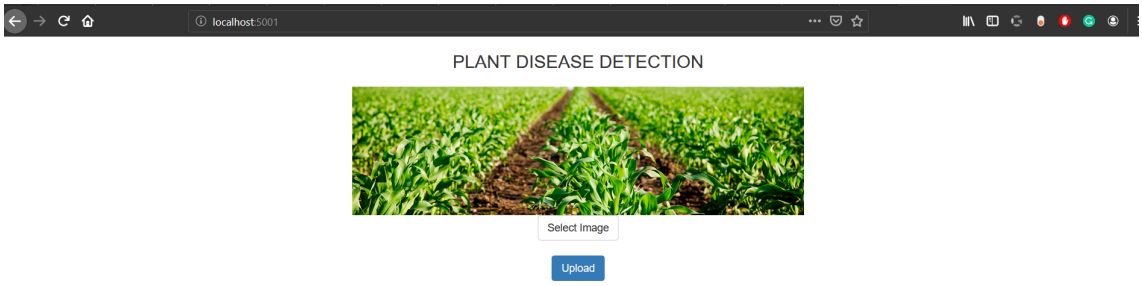


Figure 6.8: The basic interface

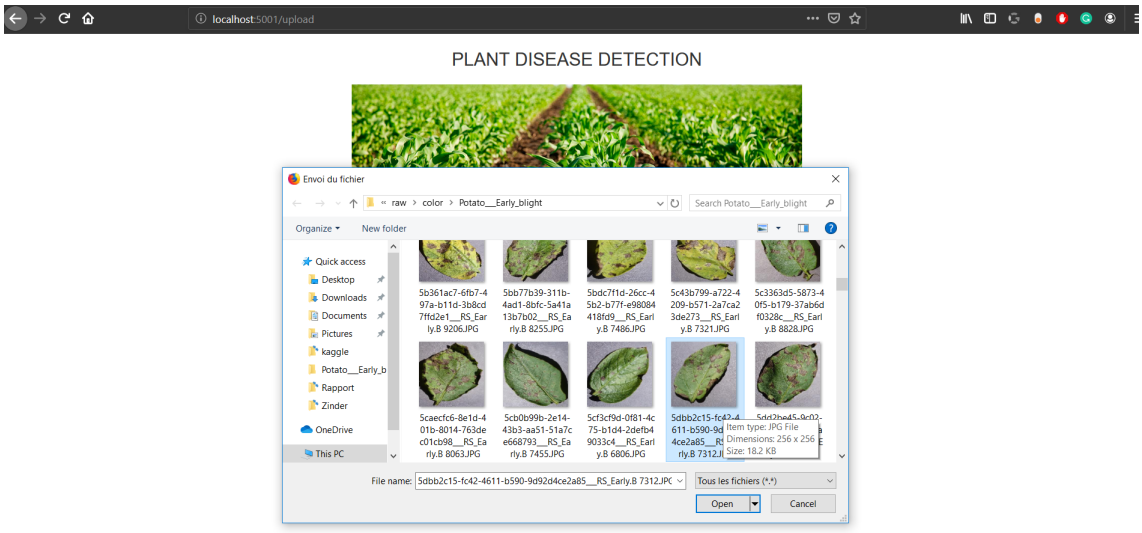


Figure 6.9: Uploading an image

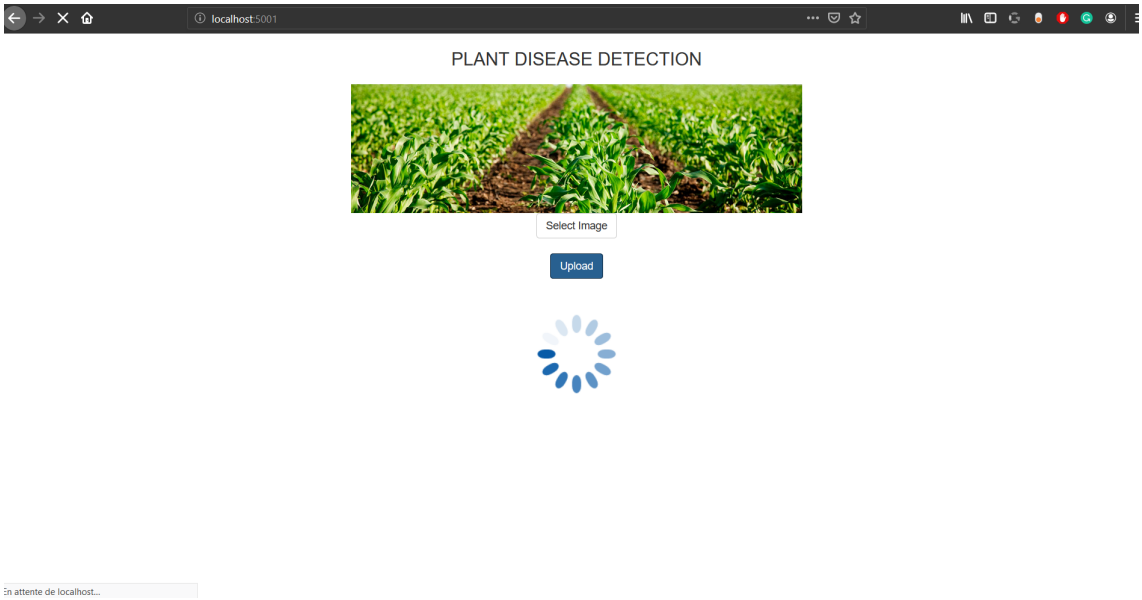


Figure 6.10: Making a prediction

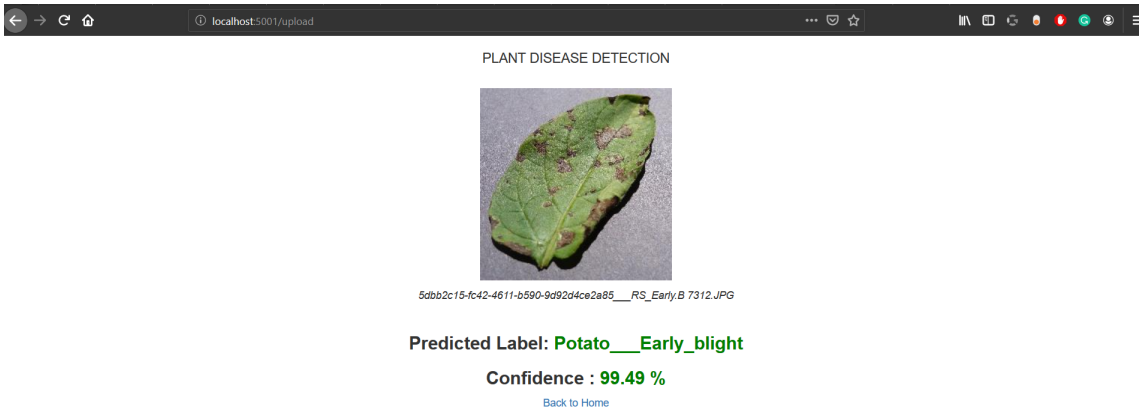


Figure 6.11: The result

Conclusion

This chapter was devoted, in the first part, for the description of the software and hardware environment. In the second part, we described the data and the preprocessing steps. In the third part, we detailed the model training process. In the fourth part, we showed some of the results we obtained and as for the last part, we discussed the deployment

of the model into a web application and showed some interfaces.

Conclusion and Prospects

The purpose of this project was to design a model capable of detecting plant diseases when given images of the plant leaves. After conducting a study, we found that Convolutional Neural Networks is the most efficient approach used in image recognition. Thus, we decided to adopt this revolutionary algorithm by searching and finding the suitable data to achieve our goal, and train a state-of-the-art model.

The first chapter was an introductory chapter in which we presented the hosting company, as well as, the problematic and the objectives we fixed to work on during our internship. After getting through this introductory chapter, we are going to analyse further the objectives of the project. The second chapter, titled, requirements analysis and specification, was dedicated to pinpoint the potential users of our application and to present the functional requirements and the non functional requirements. In the third chapter, we took care of explaining the global design of the application. The preliminary study chapter was dedicated, in a first part, to the definition of general concepts needed to better understand the fundamentals of our project. Then, an explanation of the adopted methodology and its projection in the context of our project. In the theoretical study chapter, we explained further the algorithms we used, the architecture of the adopted model, as well as the theory behind some techniques we used to interpret the results we obtained by our implemented algorithm. In the last chapter, titled implementation and results, first, we described the work environment, the data we used and the preprocessing steps. Second, we described briefly the steps we took to train the model. Third, we presented the results we obtained and at last, we introduced the web application and displayed some interfaces.

This internship allowed us to improve our soft skills and to enhance our sense of criticism and capability to develop a suitable solution. Besides, it was an opportunity for us to get into the professional life and develop our communication skills. Nevertheless, we faced considerable challenges while realizing this project. Image processing is a computationally intensive task, thus, a lot of time was required to see the results of the decisions we were making.

Last but not least, our model can be extended and improved by gathering more data. As expected, when testing the model on some images we gathered using Google search

engine, our model did not perform as well as on the test data. This is mainly because our model is trained on a single data distribution that is not as varied as it needs to be. Moreover, our application could be drastically improved by offering accurate solutions after the plant disease diagnosis.

Bibliography

Netography

[URL1] https://www.softwaretestingmaterial.com/software-architecture/, accessed 8/28/2019.

[URL2] https://tejaslodaya.me/blog/Primer-on-Convolution-Neural-Networks/, accessed 8/28/2019.

[URL3] http://datahacker.rs/018-cnn-inception-network/, accessed 8/28/2019.

[URL4] https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/googlenet.html, accessed 8/28/2019.

[URL5] https://github.com/metalbubble/CAM, accessed 8/28/2019.

Abstract

ملخص - هذا المشروع هو شرط للسنة الثانية في المدرسة الوطنية لعلوم الحاسوب.
يتكون مشروعنا من إنشاء تطبيق قادر على اكتشاف الأمراض النباتية من الصور
باستخدام التعلم العميق.
الكلمات الأساسية- التعرف على الصور، تعلم عميق، الشبكات العصبية التلافيفية

Résumé — Ce projet de stage est obligatoire pour la deuxième année à l'Ecole Nationale des Sciences de l'Informatique. Notre projet consiste à créer une application capable de détecter les maladies des plantes à partir d'images de feuilles en utilisant l'apprentissage approfondi .
Mots clés : Reconnaissance d'images, apprentissage approfondi, réseaux de neurones convolutionnels.

abstract— This internship project is a requirement for the second year in the National School of Computer Science. Our project consists of creating an application capable of detecting plant diseases from leaf images using deep learning
Key words : Image Recognition, Deep Learning, Convolutional Neural Networks.