MACHINE LEARNING & CONTENT ANALYTICS

Mini Project:

Plant disease recognition and remedy proposal is just one click away

Team Members:

- Cipi Klenti – p2822101
- Kakavas Pantelis – p2822115
- Konstantinou Marina – p2822121
- Tsougias Dimos – p2822133

Source: Plant Disease Identification Using Machine Learning

28/08/2022

Contents

Figures

Tables

1. Introduction

The Agri-Food sector in numbers

The importance of the Agri-Food sector for the global and local economy is undeniable. In Greece, according to data for 2019, the food and beverage industry maintains the first place in the number of companies among the manufacturing sectors (16,263 companies out of a total of 57,014), and is the largest employer in domestic manufacturing, with a percentage of 39%.

At the same time, in 2020, the domestic agricultural sector contributed 4.7% of the total Gross Value Added (GVA), while employing more than 400 thousand people, or a percentage that exceeds 10% of the total employed potential. This number is significantly reduced compared to 2001, when the sector employed 666 thousand people.

In comparison, globally, the contribution of the agricultural sector to the global Gross Domestic Product (GDP) is estimated at 4.3%. It is indicative that the added value of the industry internationally increased by 73% between 2000 and 2019, to approximately $3.5 trillion. Alongside its contribution to economic growth and prosperity, the sector also ensures social cohesion, while also contributing to the eradication of poverty (EY, 2022).

Undeniably, the Agri-Food sector plays a vital role not only in a local but a global economy and any challenges that may occur need to be handled appropriately and efficiently.

Agri-food is facing enormous challenges

Today, the Agri-Food sector is faced with enormous challenges, but also opportunities.

By 2050, the sector should feed 40% more people and increase food production by 70%, while arable land will have increased by just 10%. By then, 68% of the world's population will live in urban areas, while an estimated 12% will be undernourished. In 1970, arable land per person, worldwide, was estimated at 3.8 hectares. In 2000, it decreased to 2.3, while the forecast for 2050 is 1.5 hectares.

At the same time, the climate crisis leads to the urgent need to take measures to address serious issues related to Agri-Food, such as food waste, reckless use of water resources, greenhouse gas emissions, soil degradation, plant diseases and the reduction of biodiversity. Today, the quality of 75% of the planet's terrestrial soils has been significantly degraded, and if this trend continues, this percentage is expected to increase to 95% by 2050 (EY, 2022).

In a nutshell, although the sector will need soon to increase the food production to cover the expected population growth, the planet’s terrestrial soils are consistently leading to the expansion of plant diseases and the limitation of their healthy crop production. As a result, it is of utmost importance to appropriately and timely identify diseases including the respective remedies which will contribute to alleviate this challenge.

The pain point of plant disease recognition

The problem of efficient plant disease protection is closely related to the problems of sustainable agriculture and climate change (K. A. Garrett, 2006). Research results indicate that climate change can alter stages and rates of pathogen development and modify host resistance (S. M. Coakley, 2009).

The situation is further complicated by the fact that, today, diseases are transferred globally more easily than ever before. New diseases can occur in places where they were previously unidentified and, inherently, where there is no local expertise to combat them.

Research indicates that plant diseases cause substantial management issues and economic losses in the agricultural industry. It has been reported that at least 10% of global food production is lost due to plant disease (Peter R.Scott, 2005). Therefore, early detection, timely mitigation, and disease management are essential for agriculture production.

As aforementioned, in this changing environment, appropriate and timely disease identification including early prevention has never been more important. There are several ways to detect plant pathologies. Some diseases do not have any visible symptoms, or the effect becomes noticeable too late to act, and in those situations, a sophisticated analysis is obligatory. However, most diseases generate manifestation in the visible spectrum, so the naked eye examination of a trained professional is the prime technique adopted in practice for plant disease detection (M. B. Riley, 2012).

Due to the cultivation of many crop products, even an agriculturist and pathologist may often fail to identify the diseases in plants by visualizing disease-affected leaves. However, in the rural areas of developing countries, visual observation is still the primary approach of disease identification. It also requires continuous monitoring by experts. In remote areas, farmers may need to travel far to consult an expert, which is time-consuming and expensive.

As a result, traditionally plant disease recognition is often biased, time-consuming, and laborious due to the human factor involved.

Methods that have been developed in assisting in disease recognition

In the past, many methods have been developed to assist disease recognition and management. Lab-based techniques have been developed and established the previous decades. The commonly used techniques for plant disease recognition include enzyme-linked immunosorbent assay (ELISA), polymerase chain reaction (PCR), immunofluorescence (IF), flow cytometry, fluorescence in situ hybridization (FISH), and DNA microarrays. However, these techniques require an elaborate procedure and consumable reagents (Sankaran, 2010).

Meantime, image-based machine learning methods for plant disease recognition, which identify plant diseases by training computers with labeled plant images, have become popular.

The introduction of the automated identification through machine learning

To overcome the above problems, researchers have thought of several solutions. Various types of feature sets can be used in machine learning for the classification of plant diseases.

An automated system designed to help identify plant diseases by the plant's appearance and visual symptoms could be of great help to amateurs in the gardening process and trained professionals as a verification system in disease diagnostics. Advances in computer vision present an opportunity to expand and enhance the practice of precise plant protection and extend the market of computer vision applications in the field of precision agriculture.

Machine learning methods based on plant leaf images have been proposed to improve the disease recognition process. The latest generation of convolutional neural networks

(CNNs) has achieved impressive results in the field of image classification and are proven to be very effective (Lu, 2017).

The trend of Agri-technology and precision farming

Taking all the above into consideration, Agri-technology, and precision farming, also known as digital farming, have emerged as modern scientific disciplines that employ data-intensive methods to improve agricultural production while reducing its environmental impact. Data generated in intensive farming practices are supported by a variety of different sensors resulting in more accurate and rapid decision-making.

2. Our project

The challenge

Undeniably, the importance of the agriculture sector both for the global and local economies is enormous. That is the reason why, any challenges that appear and may influence the agriculture production need to be handled efficiently and timely.

One of the biggest pain points that have been identified concerning the agriculture sector is the plant diseases that are evolving as the climate crisis is flourishing in our days. Not to mention that diseases are transferred globally more easily than ever before.

Given that disease identification is mainly performed with visual methods, even today, from trained professionals and taking into consideration the spread of the diseases throughout the globe, it is impossible to handle all the separate circumstances arising at every corner of the earth.

The solution – Our project

Our project has the goal to overcome this obstacle by providing an easy tool with high accuracy that can be used throughout the world by anyone (from amateur gardeners to trained professional and farmers) and identify healthy from unhealthy plants and classify them based on their disease. By quickly and appropriately identifying the plant diseases throughout the world, substantial management issues and economic losses in the agricultural industry will be alleviated. Given that at least 10% of global food production is lost due to plant disease, the early identification and correct identification is of utmost importance.

Taking it a bit further

But identifying and classifying the disease of a plant is only the one side of the coin, or part of the solution. Our project aims at providing an end-to-end solution. Therefore, our tool aims at also providing which is the remedy for every separate disease identified. As a result, having only an image of a diseased plant, with the help of our project, the disease will be accurately identified, and the respective remedy will be proposed.

In a nutshell

This project is concerned with a new approach to the development of plant disease recognition model, based on leaf image classification using deep convolutional networks. The aim is to develop an easy but accurate system implementation in practice. All essential steps required for implementing this disease recognition model are fully described throughout the paper, starting from the dataset itself, up to the machine learning framework developed.

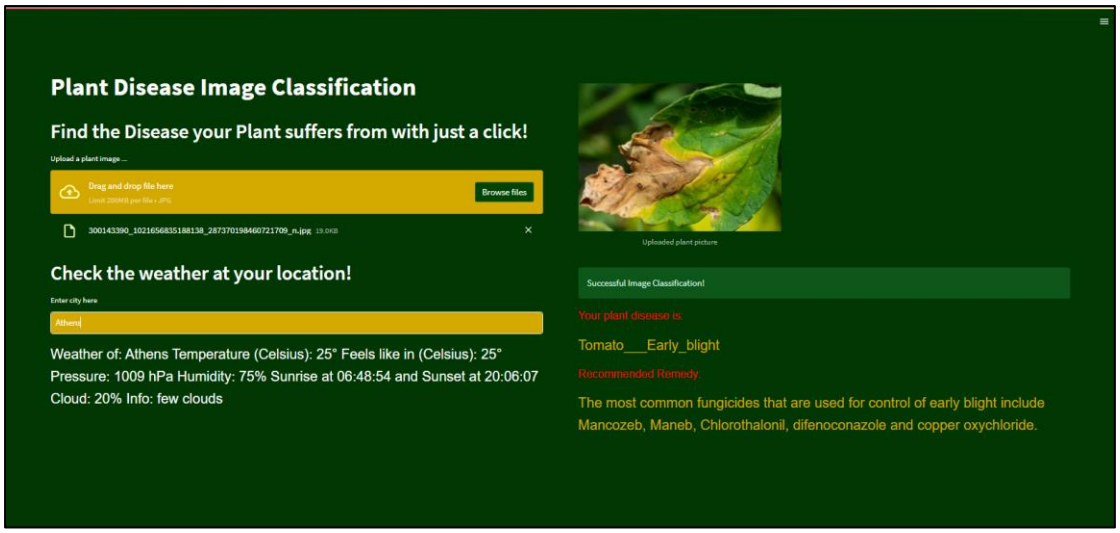
Our application

Our team has developed a User Interface (UI) to make the experience more interactive fast and efficient. With the designed UI, the user has just to upload the picture with the leaf whose disease needs to be identified and the rest will be made automatically. Specifically, the steps are listed below:

1. Upload the respective picture
2. The UI will output both the disease and the remedy proposal

An overview of the UI is presented below.

Figure 1: Identifying the disease and the respective remedy is just one click away



In the upper left part of the screen, the user must upload the image with the diseased leaf. After the file has been browsed, at the right part of the screen the application outputs the following in the order that are listed:

1. Your plant disease is: …
2. Recommended Remedy: …

Finally, at the bottom left part of the screen the user can also populate the field with their location to have an overview about:

- Temperature
- Pressure
- Humidity etc.

This information may be of use, since a lot of diseases are linked with the weather conditions.

3. Our Vision/Goals

Short Term Goals

Given that most diseases generate manifestation in the visible spectrum, the naked eye examination of a trained professional is the primary technique adopted for plant disease detection even today. Undeniably, there are some diseases with no visible symptoms that need a more sophisticated analysis, but our team's vision is to help and optimize the detection and decision-making progress in most of the diseases which are also visible to the naked eye.

The goal of the specific project is twofold. The plant disease identification is a process that concerns two main steps that are consecutive. Only after successfully performing the two steps that are being listed below, the last step of overcoming the disease with the appropriate pesticides can take place:

1. Identify the exact plant that is being examined
2. Identify the disease of the plant

Given, that as a team we want to provide an end-to-end solution, the vision does not stop to the identification step. Having identified both the plant and the disease, the trained model will be able to also provide which are the most appropriate remedies (pesticides) to overcome the specific disease.

In a nutshell, what we wish to accomplish is to provide an automated system designed to help identify plan from amateurs in the gardening process to trained professionals as a verification system in disease diagnostics. With a simple photo taken a three-way output will be produced:

- Plant identification
- Disease identification
- Appropriate remedies

Future Work

The main goal for the future work will be developing a complete system containing a trained model and an application for smart phone devices with features such as displaying recognized diseases in fruits, vegetables, and other plants, based on leaf images captured by the phone camera.

This application will serve as an aid to farmers regardless of the level of experience, enabling fast and efficient recognition of plant diseases and facilitating the decision-making process when it comes to the use of chemical pesticides.

Furthermore, future work will involve spreading the usage of the model by training it for plant disease recognition on wider land areas, where it would be possible to even identify the main cause of the plant disease such as worms, insufficient arable land properties etc.

By extending this research, we hope to achieve a valuable impact on sustainable development, affecting crop quality for future generations and optimizing the decision-making process given that both the disease identification and the route cause will be the combined output from a single solution.

4. Methodology

Overview

To have a holistic view on the subject and be sure that we are providing the best possible model with the best results, different methodologies were followed as listed below:

1. Convolutional Neural Networks (CNN)
2. Data Augmentation
3. Transfer Learning
4. Vision Transformers (ViT)

The reason why these methodologies were chosen and how they were applied to optimize the results in our project is presented below in detail.

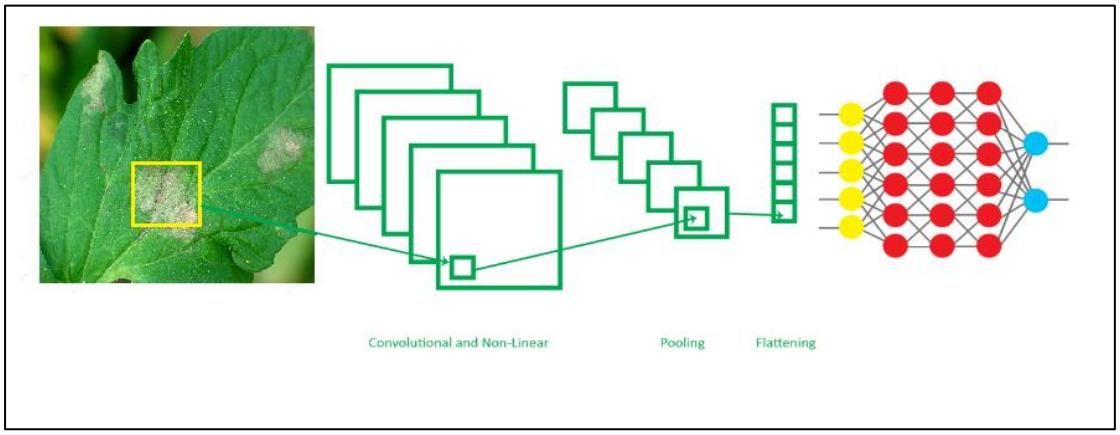
Convolutional Neural Networks (CNN)

Convolutional Neural Networks are used as the supporting framework of our method. CNN is a class of deep, feed-forward artificial neural networks. It was adopted widely for its fast deployment and high performance on image classification tasks. CNNs are usually composed of convolutional layers, pooling layers, batch normalization layers and fully connected layers.

The convolutional layers extract features from the input images whose dimensionality is then reduced by the pooling layers. Batch normalization is a technique used to normalize the previous layer, which can increase the stability and improve the computation speed of the neural networks. The fully connected layers are placed near the output of the model. They act as classifiers to learn the non-linear combination of the high-level features and to make numerical predictions.

It should be noted that CNN requires a large training dataset, which is typically not the case for plant disease recognitions. When the number of model parameters is greater than the number of data samples, a small training dataset will lead to the overfitting problem, which results from a model that responds too closely to a training dataset and fails to fit additional data or predict future observations reliably. One of the commonly adopted methods to address this problem is data augmentation (Guo J., 2015).

Figure 2: Convolutional Neural Networks in Plant Disease Identification



Source: CNN

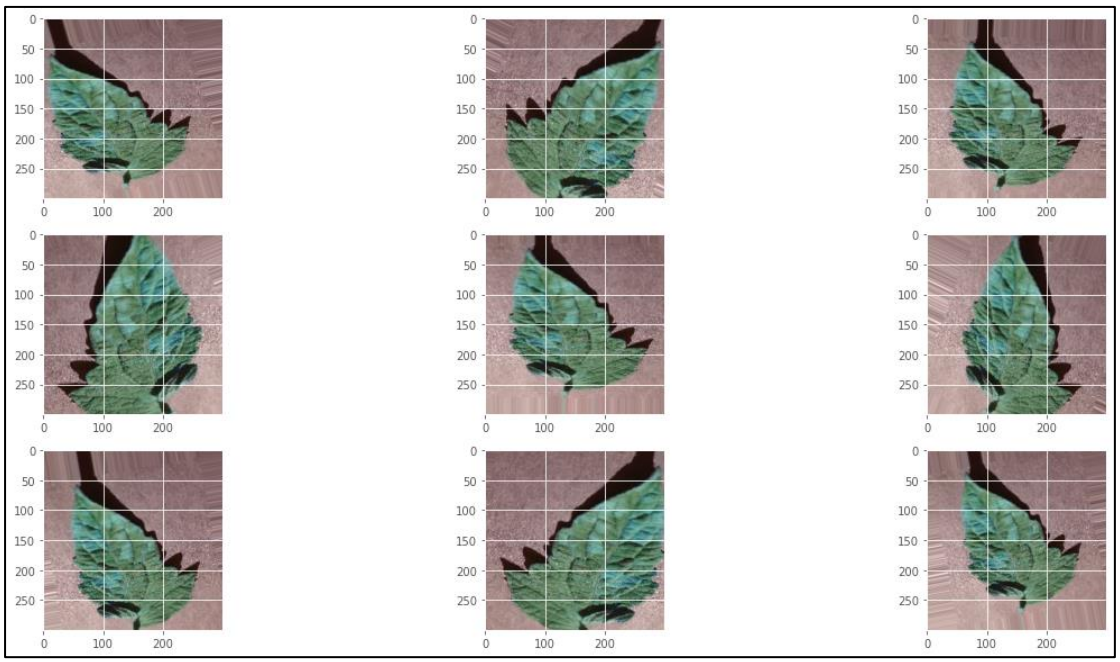
Data Augmentation

Data augmentation is a method to increase the number of labeled images. The classic data augmentation methods include vertical flipping, horizontal flipping, 90° counterclockwise rotation, 180° rotation, 90° clockwise rotation, random brightness decrease, random brightness increase, contrast enhancement, contrast reduction and sharpness enhancement.

The main purpose of applying augmentation is to increase the dataset and introduce slight distortion to the images which helps in reducing overfitting during the training stage. In machine learning, as well as in statistics, overfitting appears when a statistical model describes random noise or error rather than underlying relationship (Hawkins).

For the augmentation process, simple image rotations were applied, as well as rotations on the different axis by various degrees. An example of an augmented image from the training dataset is presented below.

Figure 3: Data Augmentation Example from the PlantVillage Dataset



Transfer Learning

Transfer learning helps developers take a blended approach from different models to fine-tune a solution to a specific problem. The sharing of knowledge between different models can result in a much more accurate and powerful model. The approach allows for the building models in an iterative way.

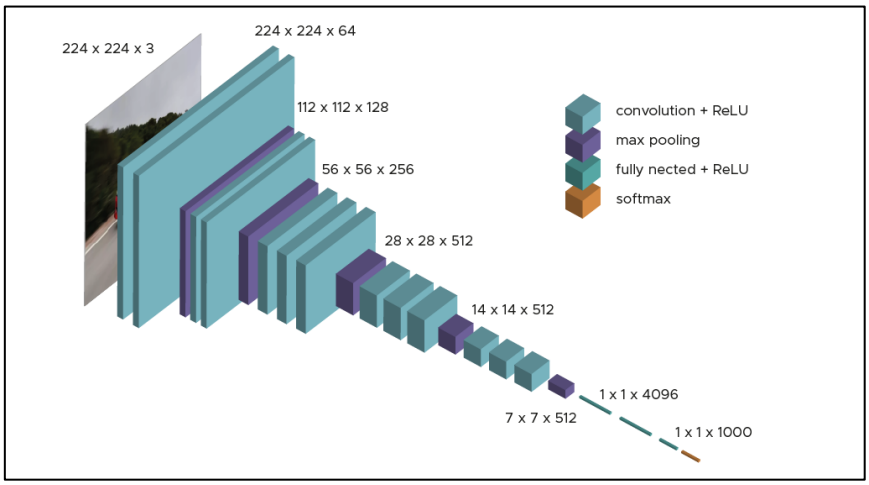
As a result, the concept of transfer learning has been used in our project for the classification. The main advantage in using transfer learning is that instead of starting the learning process from scratch, the model starts from patterns that have been learned when solving a different problem which is similar in nature to the one being solved (A., 2018).

This way the model leverages previous learnings. In image classification, transfer learning is usually expressed using pre-trained models. A pre-trained model is a model that was trained on a large benchmark dataset to solve a similar problem. In our project we have used two different pre-trained models:

1. VGG19
2. ResNet50

In a nutshell, transfer learning brings a range of benefits to the development process of machine learning models. The main benefits include the saving of resources and improved efficiency when training new models. Basically, transfer learning provides training speed since most of the model parameters are taken as they are from the pretrained model and only some of the parameters from the top layers need to be trained specifically for our task.

Figure 4: Transfer Learning

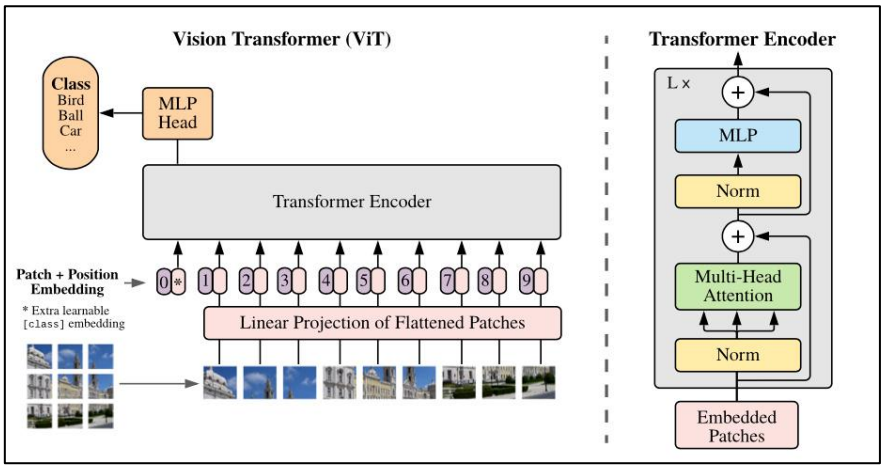


Source : VGG19 architecture

Vision Transformers (ViT)

ViT is a visual model based on the architecture of a transformer originally designed for text-based tasks. The ViT model represents an input image as a series of image patches, and directly predicts class labels for the image. ViT exhibits an extraordinary performance when trained on enough data, breaking the performance of a similar state-of-art CNN with 4x fewer computational resources. These transformers have high success rates when it comes to NLP models and are now also applied to images for image recognition tasks (viso.ai, 2022).

Figure 5: ViT for Image Classification with Transformers



Source: ViT - huggingface

The overall architecture of the vision transformer model is given as follows:

1. Split an image into patches
2. Flatten the image patches
3. Create lower-dimensional linear embeddings from these flattened image patches
4. Include positional embeddings
5. Feed the sequence as an input to a state-of-the-art transformer encoder
6. Pre-train the ViT with image labels, which is then fully supervised on a big dataset
7. Fine-tune the downstream dataset for image classification

5. Data Collection

Project Goal Leading to Dataset Selection

The goal of this project is to develop an automated system designed to help identify plant diseases by the plant's appearance and visual symptoms that could be of great help from amateurs in the gardening process to trained professionals as a verification system in disease diagnostics.

The situation is further complicated by the fact that, today, diseases are transferred globally more easily than ever before. New diseases can occur in places where they were previously unidentified and, inherently, where there is no local expertise to combat them.

Dataset Selection

Taking into consideration all the above, to develop a tool which will be useful for amateurs and trained professionals throughout the world, we as a team needed to include as many plant species as possible and their respective diseases.

Given the limited time of the specific project, the choice to create our own dataset seemed impossible and therefore we had to search and use an existing dataset from the internet. For us as a team, the dataset to be appropriate for use needed to meet specific requirements which are listed below:

1. Be large enough to include many plant species and diseases
2. Have a valid source to be sure about the information that it contains
3. Being used in the past for other projects to be able to have reference

After searching in various valid sites such as:

- www.kaggle.com
- www.github.com etc.

we found many different datasets that are referring to plant species and their respective diseases. Since the subject of plant disease detection though image classification is very popular these days, many datasets seem to meet our requirements.

Selected Dataset

At last, we found a large dataset which have been used in similar projects in the past with goods reviews as far as the validity is concerned. Some basic characteristics of the dataset are presented below:

- **Name**: Plant Village Dataset
- **Link**: Dataset of diseased plant leaf images and corresponding labels
- **Raw Directory**: 3 different versions of imagers
    - Color
    - Grayscale
    - Segmented

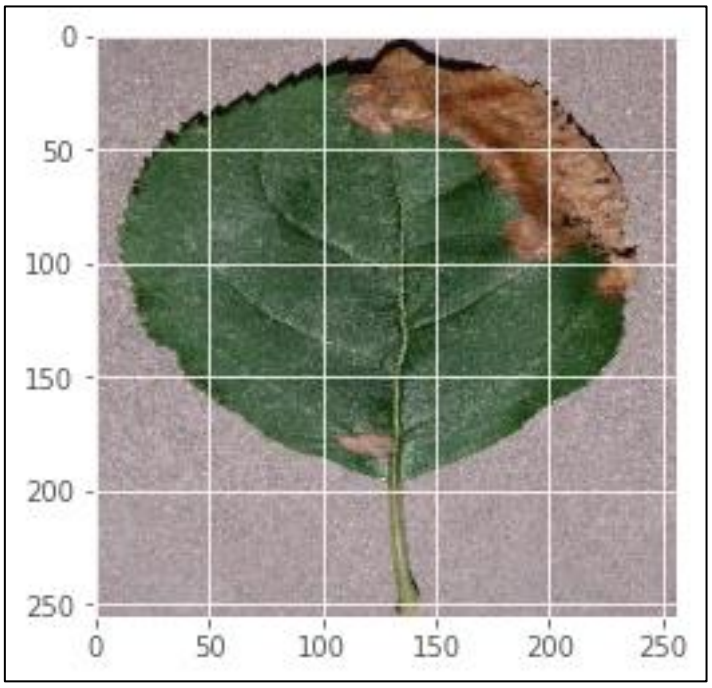
More detailed information concerning the dataset can be found in the next chapter.

6. Dataset Overview

Dataset

For our project we used the standard open-access PlantVillage dataset, which consists of 54,305 images of healthy and infected plant leaves. The database contains 38 different classes of 14 different plant species. The below figure shows a single leaf image from the PlantVillage dataset.

Figure 6: Sample Image from the PlantVillage Dataset



The raw data of the dataset included three different formats:

1. Color scale images
2. Grayscale images
3. Segmented images

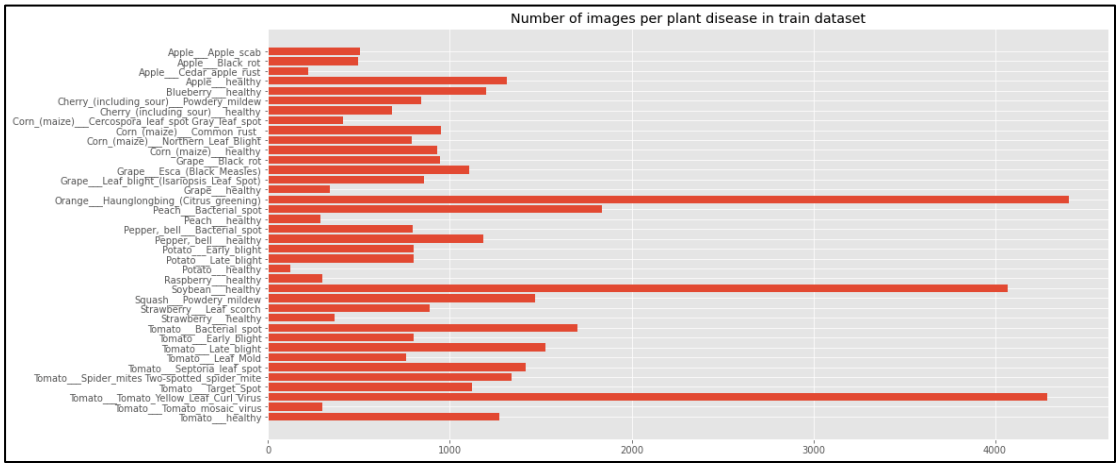
Since the most common practice is to have colored images, we decided as a team to keep and work on the colored scale format of the provided images from the selected dataset. The collected dataset uses images from databases of multiple countries for global wide acceptance of proposed framework. The images consist of laboratory images to develop a robust framework.

Image distribution between classes

For the colored images that were taken into consideration in our analysis, the total number of images per class (38 total classes) is presented below.

It is clear from the graph, that some diseases have far more images than other diseases. At first, this seemed like a problem of bias for our project, but though the use of the computations of the data loader, this issue was overcome.

Figure 7: Number of Images per Plant Disease on PlantVillage Train Dataset



The dataset was divided into training, validation and testing set for training and examining the framework. The percentages of the three different sets are listed below:

- Training set – 80% (presented in Figure 7)
- Validation set – 10%
- Testing set – 10%

7. Data Processing/Annotation/Normalization

Raw Data

Having carefully researched the internet for appropriate datasets, as explained beforehand the dataset that we finally selected met all the requirements that were set by our team. The raw data initially included three different formats:

1. Color scale images
2. Grayscale images
3. Segmented images

and as explained in the previous chapters, our team worked with the color scaled images. The dataset had the images in different directories and folders based on the plant species and their respective diseases.

A very important attribute of the selected dataset was that all the images were in the same size of 256 x 256 pixels x 3 channels. Therefore, resizing was not needed for any of the images that were included in our analysis. Moreover, the quality of all the images was sufficient and therefore no issues were expected to arise from the quality, or the sizes of the images provided by the dataset itself.

Therefore, no extra pre-processing steps were needed concerning the quality and the sizes of the images.

Data Processing

Building an effective neural network model requires careful consideration of the network architecture as well as the input data format. Concerning the raw data, two different techniques were applied:

- Scaling

Being ensured that all images are square, we scaled each image appropriately by dividing with 255. Scaling 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 augmentation

The main purpose of applying augmentation was to increase the dataset and introduce slight distortion to the images which helps in reducing overfitting during the training stage.

For the augmentation process, simple image rotations, zooms and shears were applied, as well as rotations on the different axis by various degrees. Better accuracy than that of a traditional machine-learning-based approach is expected due to data augmentation.

In a nutshell

The raw data were in a very good shape with no need for cleansing. Some techniques were applied to be sure that an effective neural network model will be built with the appropriated input data format.

8. Algorithms, NLP architectures/systems

Overview

Having developed a proper dataset with the pre-processing techniques that were described in detail in the previous chapter (data augmentation, scaling etc.), the finalized dataset was divided into:

- 80% training set
- 10% validation set
- 10% training set

Then, a deep learning architecture was trained for 40 epochs using the finalized data. The performance of the trained model was evaluated on various parameters which were fine tuned to optimize the performance. At the end, the trained model was tested on the unseen images to countify the out of sample ability of the model.

To have a holistic view on the subject and be sure that we are providing the best possible model with the best results, different techniques were followed as listed below:

5. We created our **own model from scratch,** and we fine-tuned the hyperparameters to optimize the performance
6. We used **transfer learning** to utilize patterns that have been learned when solving a different problem from other already developed models
7. We used **Vision Transformers** which exhibit an extraordinary performance when trained on enough data, breaking the performance of a similar state-of-art CNN

Own model from scratch

Deep learning has revolutionized the field of image classification and computer vision. The general architecture of convolutional neural network composed of multiple numbers of layers e.g., convolutional layer, pooling layers, fully connected layers with other activation functions.

The input data is molded into the dimensions of the input layer of the convolutional neural network. Then the input data passes through the multiple layers and features are extracted layer by layer to learn the object class and to differentiate well from the other classes.

We finetuned the network with different parameters to achieve optimal results. We performed extensive testing by adjusting the different parameters. We used different batch sizes in the range of 64 – 256, and different dropout values in the range of 0.1 – 0.4.

To optimize the model, we tested it with different learning rates.

We compared the performance of the implemented model with that of other transfer learning-based CNN deep-learning models and state-of-the-art machine-learning techniques. Results showed that the implemented model performed worse in terms of both accuracy and required training time. Detailed analysis concerning the results is included in Chapter 10. Results & Quantitative Analysis (incl. visualizations).

Transfer Learning

A transfer-learning-based CNN was also applied. In each model, we froze the layer weight before the first fully connected layer and removed all layers after that. We have also used dropout layers with different dropout values, which prevents the architecture from overfitting. Specifically, two different transfer learning models were applied which are described in detail below.

Results showed that the implemented models performed better in terms of both accuracy and required training time from the model that we developed from scratch. Detailed analysis concerning the results is included in Chapter 10. Results & Quantitative Analysis (incl. visualizations).

- **VGG19**

VGG-19 was trained on ImageNet Competition Data with purpose to identify the main object in an image. It is noteworthy for its extremely simple structure, being a simple linear chain of layers, with all the convolutional layers having a kernel size of 3x3. Despite this simple structure, it achieves competitive classification accuracy although at the cost of slower evaluation speed and much larger net size (Repository, 2017).

The validation and test data consisted of 150,000 photographs, collected from flickr and other search engines, hand labeled with the presence or absence of 1000 object categories. The 1000 object categories contain both internal nodes and leaf nodes of ImageNet, but do not overlap with each other (Net, 2014).

- **ResNet50**

ResNet50 was trained on ImageNet Competition Data with purpose to identify the main object in an image. ResNet-50 is a convolutional neural network that is 50 layers deep (48 Convolution layers along with 1 MaxPool and 1 Average Pool layer). A residual neural network (ResNet) is an artificial neural network (ANN) of a kind that stacks residual blocks on top of each other to form a network (Medium, 2021).

The validation and test data consisted of 150,000 photographs, collected from flickr and other search engines, hand labeled with the presence or absence of 1000 object categories. The 1000 object categories contain both internal nodes and leaf nodes of ImageNet, but do not overlap with each other (Net, 2014).

Vision Transformers (ViT)

Having developed both the model from scratch and having used transfer-learning based CNNs, the final step was to check whether the Vision Transformers perform better than the aforementioned.

Vision Transformers is a quite new methodology which started in 2017. Concerning the architecture and the application of ViT in our project we took into consideration the official documentation from Keras.

Some of the basic steps that were applied for the implementation of the Vision Transformers architecture in our project are presented below:

- set hyperparameters
- define function to create MLPs
- construct patches class and PatchEncoder
- perform data augmentation

- build ViT classifier
- set callbacks

Results

Having developed the three different methodologies that were described above:

1. our own model from scratch
2. Transfer Learning CNNs
3. Vision Transformers (ViT)

The results concerning the performance of the trained model, indicated that Vision Transformers brought the optimal results as it was also anticipated since from the bibliography ViT seems to exhibit an extraordinary performance when trained on enough data, breaking the performance of a similar state-of-art CNN. Nevertheless, once we embedded the ViT model in our UI and tested it on random images from google and some images that we took on our own, it did not perform well.

More detailed information about the setup and configuration of the three different methodologies is included in chapter 9. Experiments – Setup, Configuration and concerning their respective results everything is clearly stated in chapter 10. Results & Quantitative Analysis (incl. visualizations).

9. Experiments – Setup, Configuration

To have a deep dive view on the subject, the setup and configuration of the three different methodologies applied are presented below in detail.

Callbacks

First, we set some callbacks for the models that were going to be trained. Specifically, we monitored the validation loss and put early stopping in case a model did not improve for 8 epochs. We saved each time the best model based on the validation loss and if for 3 epochs it did not improve, the learning rate was reduced with a factor of 0.1.

The aforementioned was applied in all the models that were trained with the 3 different methodologies selected:

- Own model from scratch
- Transfer Learning
- Vision Transformers (ViT)

Own model from scratch

The general architecture of convolutional neural network is composed of multiple numbers of layers e.g., convolutional layer, pooling layers, fully connected layers with other activation functions.

In our model built from scratch the following layers were applied:

- 2D convolutional (64 units) followed by MaxPooling and a Dropout of 0.1
- 2D convolutional (128 units) followed by MaxPooling and a Dropout of 0.25
- 2D convolutional (256 units) followed by MaxPooling and a Dropout of 0.25
- A layer to flatten the outcome
- A final MLP with 1024 units and the 38 classes

The optimizer used for the model is Adam (suggested by official Keras implementation).

Concerning callbacks as mentioned above EarlyStopping of eight epochs patience was set, reducing our learning rate by 0.1 factor with three epochs patience and saving our best model. The monitoring metric was validation loss.

Transfer Learning

A transfer-learning-based CNN was also applied with VGG19 and ResNet50 architecture. In each model, we froze the layer weight before the first fully connected layer and removed all layers after that. We have also used dropout layers with different dropout values. Specifically:

- VGG19
    - Added dense layers along with activation and batch normalization
    - Added dropout layers with different values (0.2, 0.4)
    - Used Adam optimizer
- ResNet50
    - A layer to flatten the outcome
    - A dropout of 0.2
    - A final MLP with 1024 units and the 38 classes

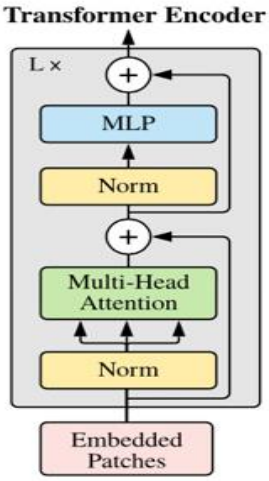
Basically, transfer learning provides training speed since most of the model parameters are taken as they are from the pretrained model and only some of the parameters from the top layers need to be trained specifically for our task.

Concerning callbacks as mentioned above EarlyStopping of eight epochs patience was set, reducing our learning rate by 0.1 factor with three epochs patience and saving our best model. The monitoring metric was validation loss.

Vision Transformers (ViT)

The steps that were applied for the Vision Transformers architecture is presented below in detail. The official Keras documentation for ViT was taken into consideration:

- Set our hyperparameters. Our image size is 72x72 where our patch size is 6x6 resulting in 144 patches per image. The projected dimensions of the patches were 64 and the number of heads of the multi head self-attention four. The layers of the feed forward of the encoder have been set to two with 128 and 64 units respectively. The transformer layers of the encoder were eight and lastly, we added our top MLP with 2048 & 1024 units before our final classifier with our 38 classes.
- Define a function to create MLPs with gelu activation function and a certain dropout rate. While relu activation also used in this implementation gelu function, being the one suggested in Keras, indeed performed better.
- Construct a Patches class where the image patches are been created with sizes and strides in respect to the patch size in order not to miss any pixels of our image but also to not have overlapping patches.
- Construct PatchEncoder class where this layer will linearly transform a patch by projecting it into a vector of size 64 according to our initial training arguments. In addition, it adds a learnable position embedding to the projected vector.
- Perform data augmentation where we normalize, resize, flip, zoom and rotate the images
- Build our ViT classifier according to the transformer encoder of the ViT architecture. We take our input images, with the help of our Patches and PatchEncoder class the Embedded Patches are being created and fed to the encoder where they are being normalized. Then the multi-Head Attention layer takes over to find the necessary relevancies and then the outcome is being added to the original embedded patches and after a second normalization are being fed to our MLP. The outcome of the MLP is again added to our second skip connection. This process is being repeated eight times and finally after the encoder there's another normalization layer, a layer to flatten the outcome, a dropout layer and our final MLP with 2048 & 1024 units and the 38 classes classifier.



The optimizer used for the model is AdamW (suggested by official Keras implementation) with weight decay of 0.0001. AdamW is a variant of the optimizer Adam that has an improved implementation of weight decay to further lower our chance of overfitting.

Concerning callbacks as mentioned above EarlyStopping of eight epochs patience was set, reducing our learning rate by 0.1 factor with three epochs patience and saving our best model. The monitoring metric was validation loss.

10. Results & Quantitative Analysis (incl. visualizations)

Performance Metrics

To quantify the performance of the trained models, different metrics were set.

Concerning the validation and test set (10% of the raw data each) the following were applied and plotted to visualize how the performance is evolved as the epochs are growing:

- Accuracy diagnostic plots
- Loss diagnostic plots
- Top-5 accuracy

Additionally, for the model with the optimal performance which came from the ViT architecture more measures were taken into consideration. The following were checked:

- Confusion matrix
- Classification report

The results for both the validation and test set are presented below in detail alongside their respective plots.

Train & Validation Set

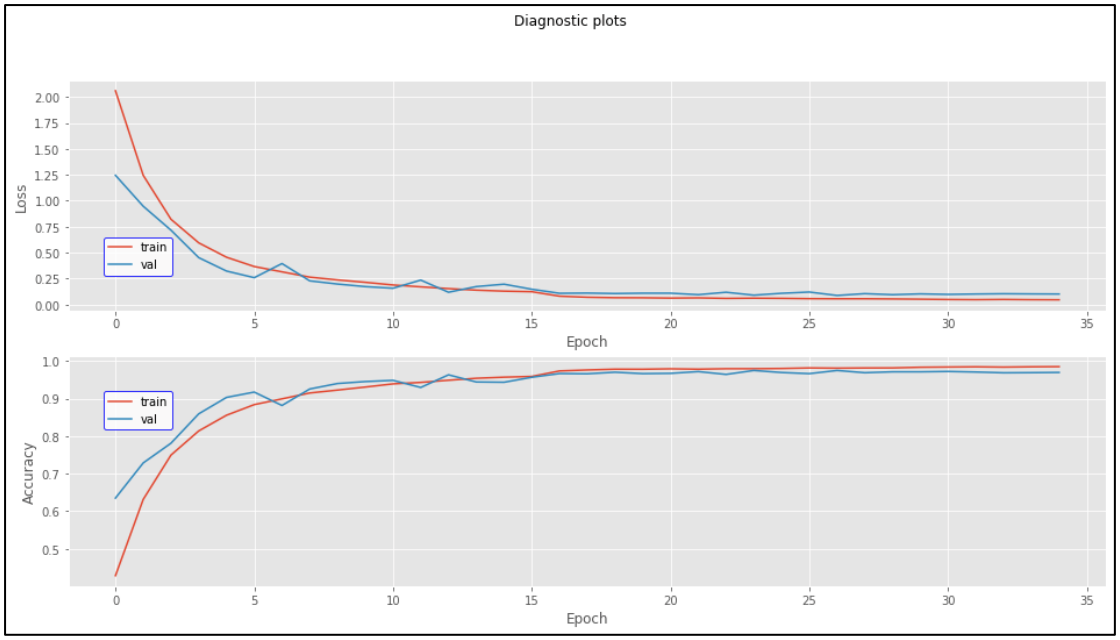
Own model from scratch

For the train & validation set the results are the following:

Table 1: Results for the Own Model from Scratch

Validation Set Results	
Validation accuracy	0.9746
Validation loss	0.0877

Figure 8: Diagnostic Plots for the Own Model from Scratch



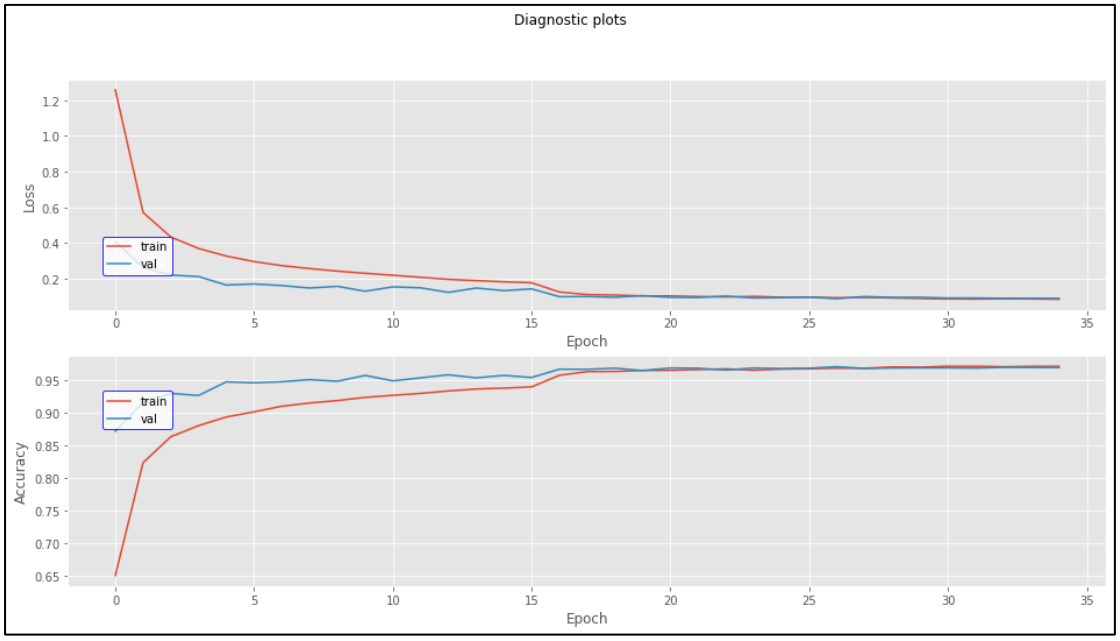
VGG19

For the train & validation set the results are the following:

Table 2: Results for VGG19

Validation Set Results	
Validation accuracy	0.9707
Validation loss	0.0875

Figure 9: Diagnostic Plots for VGG19



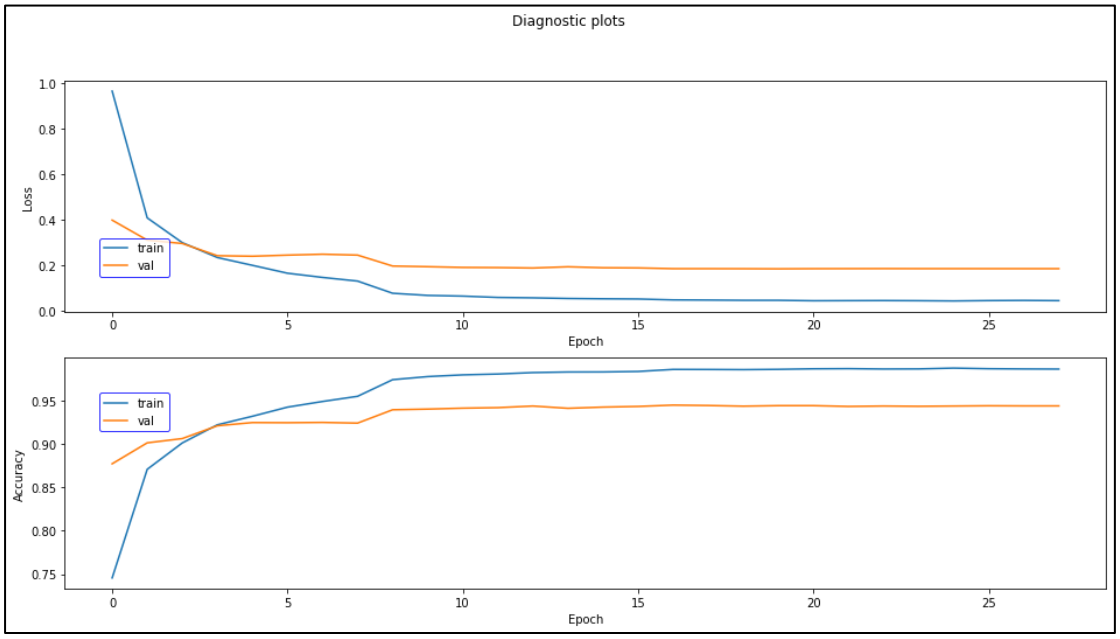
ResNet50

For the train & validation set the results are the following:

Table 3: Results for ResNet50

Validation Set Results	
Validation accuracy	0.9438
Validation loss	0.1839

Figure 10: Diagnostic Plots for ResNet50



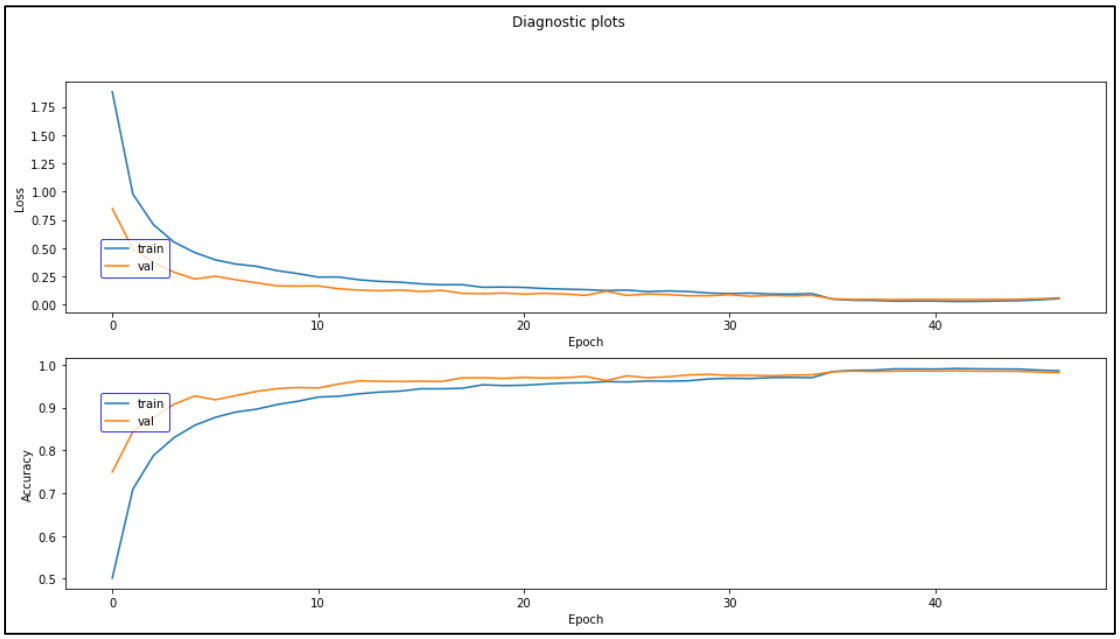
Vision Transformers (ViT)

For the train & validation set the results are the following:

Table 4: Results for ViT

Validation Set Results	
Validation accuracy	0.9850
Validation loss	0.0451

Figure 11: Diagnostic Plots for ViT



Top-5 Accuracy

On the validation set, an extra metric that has been applied to all the models is the Top-5 Accuracy.

For the Top-5 accuracy which means any of our model's top 5 highest probability answers match with the expected answer, the performance of each model is presented below.

Table 5: Top-5 Accuracy on the Validation Set

Model	Top-5 Accuracy
Own model from scratch	0.9991
VGG19	0.9993
ResNet50	0.9979
ViT	0.9995

Test Set

For the test set the accuracy and loss of all the models was quantified to identify the one with the optimal results.

Table 6: Test Set Accuracy & Loss for all models

Model	Test Accuracy	Test Loss
Own model from scratch	0.9769	0.0956
VGG19	0.9706	0.0893
ResNet50	0.9328	0.2156
ViT	0.9866	0.0527

As indicated in Table *6*, ViT had the better results from all the models that were trained. Nevertheless, once we embedded the ViT model in our UI and tested it on random images from google and some images that we took on our own, it did not perform well. More detailed information about this issue is included in chapter 11. Qualitative & Error Analysis. As a result, we tried to do the same thing with the second-best model concerning the test set (VGG19) which performed really close to the percentages from the test set.

Therefore VGG19 being the optimal methodology, more metrics were taken into consideration for its performance. These are presented below.

Classification Report for VGG19

Evaluation is done in different parameters such as true positive (TP), true negative (TN), false positive (FP), false negative (FN), precision, recall, and F1-score.

TP are correct data labels which were predicted correctly with respect to the ground truth. FP are those negative data labels which have been predicted wrong and categorized into a different category of image label. TN are those negative data samples which have been accurately predicted. FN are those positive data labels which were predicted wrong.

- Precision = TP / (TP + FP)
- Recall = TP / (TP + FN)
- F1 Score = [2 × (Precision × Recall)] / (Precision + Recall)

The results for the 38 different classes are presented in the table below. Concerning support, support is the number of actual occurrences of the class in the specified dataset.

Table 7: Classification Report Results for VGG19

Classification Report

	precision	recall	f1-score	support
0	0.97	0.97	0.97	63
1	0.98	1.00	0.99	63
2	0.96	0.96	0.96	28
3	0.97	0.99	0.98	165
4	0.98	1.00	0.99	151
5	1.00	0.98	0.99	106
6	1.00	1.00	1.00	86
7	0.84	0.88	0.86	52
8	0.99	1.00	1.00	120
9	0.95	0.92	0.93	99
10	1.00	1.00	1.00	117
11	0.97	0.98	0.98	118
12	0.98	0.98	0.98	139
13	1.00	1.00	1.00	109
14	1.00	0.98	0.99	43
15	1.00	1.00	1.00	552
16	1.00	0.98	0.99	231
17	0.92	0.97	0.95	36
18	0.98	0.97	0.98	101
19	0.97	1.00	0.99	149
20	1.00	0.99	0.99	100
21	0.97	0.87	0.92	100
22	0.71	0.94	0.81	16
23	1.00	0.97	0.99	38
24	1.00	1.00	1.00	509
25	1.00	0.99	1.00	184
26	0.99	0.98	0.99	112
27	0.98	1.00	0.99	47
28	0.98	0.97	0.98	214
29	0.96	0.71	0.82	100
30	0.93	0.92	0.92	192
31	0.91	0.92	0.91	96
32	0.94	0.95	0.95	178
33	0.86	0.98	0.91	169
34	0.84	0.90	0.87	141
35	0.99	0.98	0.98	537
36	0.80	0.95	0.87	38
37	0.98	0.97	0.98	160
accuracy			0.97	5459
macro avg	0.96	0.96	0.96	5459
weighted avg	0.97	0.97	0.97	5459

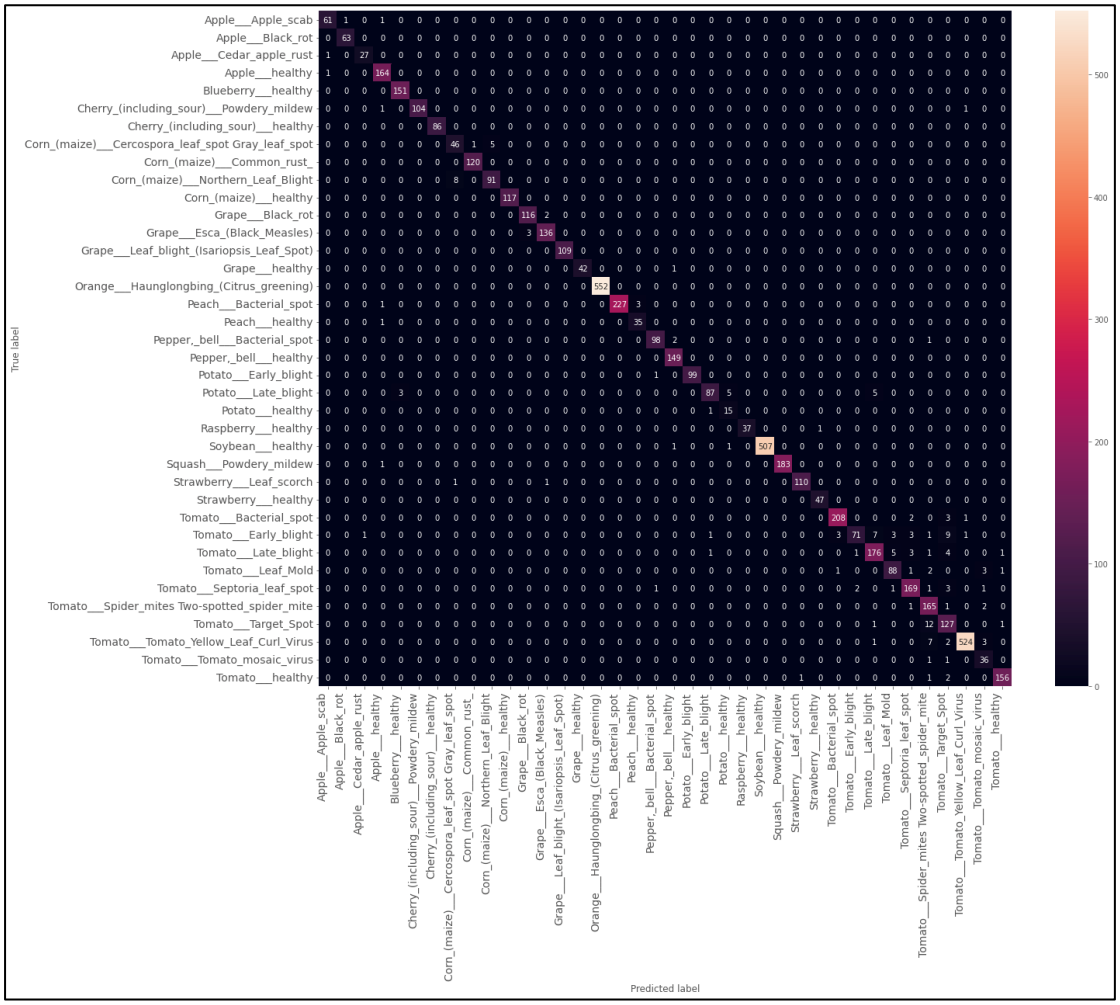
The results of the classification report indicate that the model with the ViT architecture is performing very well in all the 38 different classes. For example, the precision does not drop to any class below 0.84.

Confusion Matrix for VGG19

Confusion matrix is used for the proper evaluation of the data in which ground truth and predicted labels are represented. The diagonal elements are representing the true positives and are directly proportional to the overall accuracy of the trained model.

The confusion matrix is given below for all 38 different classes. It is shown that the proposed model has shown highly accurate results. The confusion matrix illustrates that the model is performing very well in the test set as well.

Figure 12: Confusion Matrix for VGG19



11. Qualitative & Error Analysis

Overview

Having analyzed the performance of the three different methodologies with quantitative metrics such as accuracy and loss, classification report and confusion matrices, the next step is to go through the qualitative and error analysis.

Specifically, this chapter tries to understand in which classes the optimal VGG19 model did not perform well and identify the causality behind it. Therefore, both the quantitative results will be needed, and especially the confusion matrix, as well as the information about the dataset that has been used for the model.

Identifying the weak points

Based on the confusion matrix, the classes in which the model has not performed well, are the ones which have positive numbers in their rows in places other than in the diagonal of the matrix. As indicated by *Figure 12*, these classes are the following (a minimum threshold of 3 wrongly classified images is set) :

- Apple scab
- Potato late blight
- Tomato late blight
- Tomato spider mites

Classes with few images in the training set

For some of the classes listed above, the images in the train dataset presented in Figure *7* are relatively few. Some classes appear to have over 4000 images and others just around 500. Classes such as, ***"Apple scab"*** appear to have a small number of images in the raw dataset and it may be a cause for which the model is misclassifying some of the images.

Although there are many classes with few images, these classes are not listed above as the ones that the model is not performing well. This happens due to the data augmentation that has been applied which basically multiplies the number of trained images per class. Therefore, in one or two cases the few images on the train set may have led to worse results in the classification.

Common symptoms between diseases

After having a deep dive of the images from the classes that most of the misclassifications were made, we have identified some reasons that may have led to this result:

1. Different diseases of a same plant leaf such as the "tomato" have relatively close symptoms making it hard to be distinguished
2. Some diseases appear to have mild symptoms which due to the image quality are not easily identifiable

As a result, taking also into consideration the quantitative results and the extremely high accuracy of the VGG19 model, there are some misclassifications which are mainly caused due to relatively close symptoms between diseases or mild symptoms.

Testing our UI application with random images

As mentioned in chapter 10. Results & Quantitative Analysis (incl. visualizations) the VGG19 model performed better than the ViT one in random out of sample images. Specifically, as a last step for the error and qualitative analysis, we decided to test our model in real life conditions. At first, we tested the ViT model which had the best performance in the test set, with images that we downloaded from google but also images that we took ourselves.

Having embedded the model with the ViT architecture in the UI that we have developed, the model did not seem to perform as well as it did in the test set (98% test accuracy) which was odd at first since the test set is basically out of sample.

Therefore, we read different articles and analysis concerning the ViT architecture as it is relatively new. After cross checking the different sources most of them led to the same conclusion. ViT to perform well in out of sample data needs to be trained with dataset amounting to millions of different images which have different:

- angles
- quality
- sizes etc.

Our dataset does not meet some of these requirements since it includes around 35 thousand images from the same distribution. Specifically, all images are representing standalone leaves with a dark grey background while images from google or our own are either in brighter background or the leaves are not standalone. As a result, the ViT architecture would probably not perform greatly in out of sample data as it did.

Taking into consideration the above, we decided to embed the next best model to our UI application which came from transfer learning and specifically VGG19 with 97% test accuracy. Indeed, the VGG19 model performed very well both in random images downloaded from google and our own.

As a result, we decided to keep the VGG19 model as the optimal one since:

1. it performs approximately the same in the test set with the ViT one (97% & 98% respectively)
2. it performs way better that the ViT in out of sample random images downloaded from google

Finally, as a further improvement to overcome the issues presented with the ViT architecture in the out of sample data in the long term, we will implement the pre-trained google/vit-base-patch16-224 that is being implemented through pytorch framework. This model has been trained in millions of images with a much wider distribution.

12. Discussion, Comments/Notes and Future Work

The task

The early detection and identification of plant diseases using deep-learning techniques has recently made tremendous progress. Identification using traditional approaches heavily depends on some factors such as image enhancement, the segmentation of disease regions, and feature extraction.

Our approach

Our approach was based on three different techniques:

- CNN model from scratch
- Transfer Learning
- Vision Transformers

Although the models achieved high success rates in the detection of plant diseases, some limitations appeared, which are going to be the ignite for the future work. Specifically, a little noise in the sample images led to misclassification by the deep-learning models in some cases.

Issues to be resolved

Therefore, future work includes evaluating performance on noisy images and improving it. The dataset that we used to evaluate performance included 38 different classes (diseases and healthy leaves). However, there is a need for the expansion of the dataset with wider land areas and more varieties of disease images.

Another important issue is that the testing images are all from the same image dataset. Testing the network with real-time field images is an important challenging issue. There is a need for the development of an efficient machine-learning system that could identify diseases in real-time scenarios and from collected data from different datasets.

Future Work

Furthermore, future work will involve spreading the usage of the model by training it for plant disease recognition on wider land areas, where it would be possible to even identify the main cause of the plant disease such as worms, insufficient arable land properties etc.

By extending this research, we hope to achieve a valuable impact on sustainable development, affecting crop quality for future generations and optimizing the decision-making process given that both the disease identification and the route cause will be the combined output from a single solution.

Finally, given that at this point one single classifier is working to identify both the plant as well as the respective disease, for future reference to be more productive as the list of the plant species and the diseases of our application will grow two single classifiers will be developed:

1. 1 classifier with the plant species
2. 1 classifier with the plant diseases

This method will be more productive, and a greater list of plant species and diseases will be handled efficiently and will be appropriately identified through our application.

13. Members/Roles

Our Team

Our team consists of 4 members which are alphabetically listed below:

- Cipi Klenti – p2822101
- Kakavas Pantelis – p2822115
- Konstantinou Marina – p2822121
- Tsougias Dimos – p2822133

The selection was based on two main reasons.

First, we have all worked with each other in previous assignments and therefore we know what we can bring to the table, we have a common way to operate and troubleshoot and in general we have a good interaction.

The second main reason has to do with our professions. In our team we have a developer, two engineers and a consultant. Each one of us has strengths in different subjects which can be combined and get good results.

Our Roles

For the role per member, we tried to keep a balance since it is important for all the members to be involved in all the possible responsibilities of the mini project.

First, for the initiation of the project which included the search of the project idea as well as the appropriated dataset, we all brainstormed and made a research to finalize our decision.

Then, concerning both the search of related work and the coding we tried to split the different steps that needed to be performed. In this way we were progressing faster, and all the members had the chance to take part in the coding. To make this work, we set a framework with some basic steps that need to be completed to finalize the coding such as:

- ETL
- Data augmentation
- Transfer learning
- Modelling etc.

and whilst some members of the team for example were loading and transforming the data, the other two members were trying to understand how the data augmentation is working and what needed to be done for our case.

Lastly, in the report based on the work that each one of us has performed during the mini project, we split the respective charters to be more efficient and detailed in our project presentation.

14. Time Plan

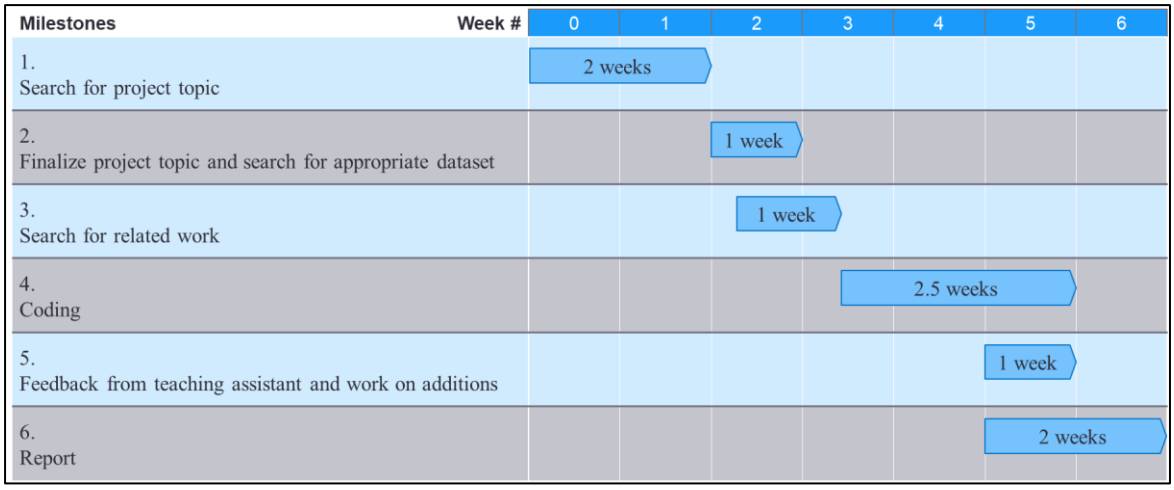
Mini Project Outline

The initial timeplan that was designed by the team, was fine-tuned during the 6-week period that was given as a deadline for the delivery of the assignment. The basic milestones that were set by the team were the following:

1. Search for project topic
2. Finalize project topic and search for appropriate dataset
3. Search for related work
4. Coding
5. Feedback from teaching assistant and perform additional work
6. Report

Timeplan

The milestones were distributed in the 6 given weeks as presented below.



It is clear for the timeplan, that the more time-consuming milestones had to do with:

- Search for topic

Although at the beginning, the selection of the topic did not seem very time consuming, after brainstorming ideas alongside the internet search, a big number of different and interesting projects was initially taken into consideration. But searching for a specific project with appropriate datasets helped our team to finalize our decision of the ***Plant Leaf Detection***.

- Coding

Undeniably, the most difficult and time-consuming milestone was the one for coding. A lot of different sources were used, but mainly through the notes from the lesson ***Machine Learning & Content Analytics*** guided us appropriately.

- *Report*

The report also took almost 2 weeks to be finalized, since we wanted to be sure that every step was explained in detail and nothing was left out.

15. Other related work

Plantix - your crop doctor

Link: Plantix - your crop doctor - Apps on Google Play

Plantix turns your Android phone into a mobile crop doctor with which you can accurately detect pests and diseases on crops within seconds. Plantix serves as a complete solution for crop production and management.

The Plantix app covers 30 major crops and detects 400+ plant damages — just by taking a photo of a sick crop. It's available in 18 languages and has been downloaded more than 10 million times. This makes Plantix the #1 agricultural app for damage detection, pest and disease control, and yield improvement for farmers worldwide.

Plant Disease Identifier

Link: Plant Disease Identifier - App Store

Plant disease identification is an app based on computer vision and machine learning tools. It helps to identify the disease by taking a photo of damaged leaf. The app has an information about 30+ plant diseases and possible options of treatment.

Pantech Solutions

Link: Leaf disease detection using CNN-Deep learning project - Pantech Solutions

In this image processing project, a deep learning-based model is proposed to solve the problem of leaf disease detection. The deep neural layer is trained using a public dataset containing images of healthy and diseased crop leaves.

The model serves its objective by classifying images of leaves into diseased categories based on the pattern of the defect. The leaves have texture and visual similarities which are attributes for the identification of disease type. Hence, computer vision employed with deep learning provides the way to solve this problem.

Department of Computer Engineering, Vidyavardhini's College, Maharashtra, India

Link: Application for detection of plant disease

Here, in this project, they are going to build an android application to detect the diseases in plants. Initially, they need to click a picture of a diseased leaf from an android device. Then the captured image will be passed on to a database server for further analysis and feature extraction. The server will receive a picture on which it will perform the steps of image processing to extract the important features of the diseased leaf.

In a nutshell

There are various applications and tools which are made for plant disease identification based on deep learning techniques.

What we wish to accomplish as a team and differentiate, is to spread the usage of the model by training it for plant disease recognition on wider land areas, where it would be possible to even identify the main cause of the plant disease such as worms, insufficient arable land properties which is not currently performed by no one. Finally, in our short-term plans we wish to expand the identification into more varieties of disease images and plant species, since most of the applications are peaking to around 30 plant species.

16. Bibliography

1. A., B. J. (2018). Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification.
2. EY. (2022). How can the Agri-Food sector face the challenges of tomorrow, today? Athens.
3. Guo J., G. S. (2015). Deep CNN Ensemble with Data Augmentation for Object Detection.
4. Hawkins, D. M. (n.d.). The problem of over-fitting. Journal of Chemical Information and Computer Sciences.
5. K. A. Garrett, S. P. (2006). Climate change effects on plant disease: genomes to ecosystems.
6. Lu, Y. Y. (2017). Identification of rice diseases using deep convolutional neural networks.
7. M. B. Riley, M. R. (2012). Plant disease diagnosis. The Plant Health Instructor.
8. Medium. (2021). Image Classification With ResNet50 Model. https://medium.com/@nutanbhogendrasharma/image-classification-with-resnet50-model-12f4c79c216b.
9. Net, I. (2014). https://image-net.org/challenges/LSVRC/2012/.
10. Peter R.Scott, R. N. (2005). PLANT DISEASE: A Threat to Global Food Security. London.
11. Repository, W. N. (2017). https://resources.wolframcloud.com/NeuralNetRepository/resources/VGG-19-Trained-on-ImageNet-Competition-Data.
12. S. M. Coakley, H. S. (2009). Climate change and plant disease management.
13. Sankaran, S. M. (2010). A review of advanced techniques for detecting plant diseases.
14. viso.ai. (2022). Vision Transformers (ViT) in Image Recognition – 2022 Guide.