People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University Of Ferhat Abbas Setif I

Departement Of Computer Sciences

Master Thesis in Computer Sciences

Option: Data Engenering and web technologies

Identification of Plant-Leaf Diseases Based Generative-Attention CNN and Transfer-Learning Approaches

---

By : Abdeldjalil Chougui

Achraf Moussaoui

Supervised By: Prof.Abdelouahab Moussaoui

JUNE 2022

Abstract

Detecting plant diseases is usually difficult without an experts knowledge. In this thesis we want to propose a new classification model based on deep learning that will be able to classify and identify different plant-leaf diseases with high accuracy that outperform the state of the art approaches, previous works and the diagnosis of experts in pathology. Using only training images, CNN can automatically extract features for classification, and achieve high classification performance. We used two datasets in this study, PlantVillage dataset containing 54,303 healthy and unhealthy leaf images divided into 38 categories by species and disease, and Tomato dataset containing 11,000 healthy and unhealthy tomato leaf images with nine diseases to train the models. We propose a deep convolutional neural network architecture, with and without attention mechanism, and we tuned 9 pretrained models that have been trained on large dataset such as Inception, Xception, MobileNet, DenseNet, VGG-16, VGG-19, EfficientNet B3, EfficientNet B5 and ResNET, we also tuned 4 ViT models that was powered by keras(vit b32), google (base patch 16), microsoft(BeiT) and facebook (DeiT). And we used YoLo V5 for plant leaf detection. Our porposed model obtained an accuracy up to 97.74%. The pretrained models gave an accuracy up to 99.86%. YoLo obtained a result of 65.4% recall, and a result of 65.3% precision. This study may aid in detecting the plant leaf diseases and improve life conditions to plants which will improve quality of humans life.

Détecter les maladies des plantes est généralement difficile sans les connaissances d'un expert. Dans cette thèse nous veulent proposer un nouveau modèle de classification basé sur le deep learning qui pourra classer et identifier différentes maladies des feuilles des plantes avec une grande précision qui surpasser l'état de l'art des approches, les travaux antérieurs et le diagnostic des experts en pathologie. En utilisant uniquement des images d'entraînement, CNN peut automatiquement extraire des caractéristiques pour la classification et atteindre des performances de classification élevées. Nous avons utilisé deux ensembles de données dans cette étude, l'ensemble de données PlantVillage contenant 54 303 images de feuilles saines et malsaines divisées en 38 catégories par espèce et maladie, et l'ensemble de données Tomato contenant 11 000 images de feuilles de tomates saines et malsaines avec neuf maladies pour entraîner les modèles. Nous proposons une architecture de réseau de neurones à convolution profonde, avec et sans mécanisme d'attention, et nous avons accordé 9 modèles préformés qui ont été formés sur de grands ensembles de données tels que Inception, Xception, MobileNet, DenseNet, VGG-16, VGG-19, EfficientNet B3, EfficientNet B5 et ResNET, nous avons également adapté 4 modèles ViT alimentés par keras (vit b32), google (base patch 16), microsoft (BeiT) et facebook (DeiT). Et nous avons utilisé YoLo V5 pour la détection des feuilles des plantes. Notre modèle proposé a obtenu une précision allant jusqu'à 97,74%. Les modèles pré-entraînés ont donné une précision allant jusqu'à 99,86%. YoLo a obtenu un résultat de 65,4% de rappel et un résultat de 65,3 % de précision. Cette étude peut aider à détecter les maladies des feuilles des plantes et à améliorer les conditions de vie des plantes, ce qui améliorera la qualité de vie des humains.

DEDICATION AND ACKNOWLEDGEMENTS

I would like to thank every one believed in me, helped me, pushed me forward to succeed, i'm really grateful to all of you. I would like to thank our supervisor Professor MOUSSAOUI Abdelouahab, I learned from you the art of Machine learning, you made it like a heaven for me, because of your passion that you pass it to me.
Mom and dad may Allah bless you ! you have been always proud and happy for me and more supportive than anyone else. Dear brothers thanks for your support, encouragements especially you my colleague Achraf, you are the best of all and the best thing that ever happened to me in this years.
The deepest appreciation to all SCA Club members and El Wiam Scout members .. I had a beautiful memories with everyone of you in every corner of the faculty and every event we did together.
Dear Hani, Islam, Zahrou, Seif, Zaki, Ayoub, Raouf ... we had fun, we studied together and know we are succeeding together, i'm lucky to have you ! Github, stackoverflow, Kaggle ,Geoffery Hinton, Andrew Ng, my first and second Laptop and every one contributed to make this work see the light, Thank you all ! I would also like to thank the members of the jury for accepting the examination of this modest work.
With Love **Abdeldjalil Chougui**

I am **Achraf Moussaoui** . . I would like to thank our supervisor Professor MOUSSAOUI Abdelouahab for his patience ,guidance and encouragement

I would like to extend our sincere gratitude to my mother for its patience and support since the first year until i have made this Master's dissertation.

Dear brothers and my sister thanks for your support and encouragments.

I would to salute my new niece Taline may god blees her and thank my partner Abdeldjalil for his work.

I would also like to thank the members of the jury for accepting the examination of this modest work.

TABLE OF CONTENTS

List of Tables

List of Figures

Abbreviation Table

Abbreviation	Meaning
SOTA	State Of The Art
CNN	convolution neural network
SVM	Support Vector Machine
KNN	K-Nearest Neighbor
Adam	adaptive moment estimation
ReLu	rectified linear unit
CBAM	Block Attention Module
CAM	Channel Attention Module
SAM	Spatial Attention Module
Grad-CAM	Gradient-weighted Class Activation Mapping
VGG	Visual Geometry Group
YOLO	You Only Look Once
ViT	Vision Transformer
BeIT	Bert Image Transformer
DeIt	Data Efficient Image Transformer
BERT	Bidirectional Encoder Representations from Transformers
TMV	Tobacco mosaic virus
TSWV	Tomato spotted wilt virus
TYLCV	Tomato yellow leaf curl virus
CMV	Cucumber mosaic virus
PVY	Potato virus Y
MSP	Multi-Head Self Attention Layer
MLP	Multi-Layer Perceptrons
LN	Layer Normalisation

Table 0.1: Abbreviation Table

CHAPTER 1

PLANT LEAF DISEASES

Agriculture is the oldest profession. Humans started cultivation science even there was no civilization. plant is a living thing that grows in earth, in water, or on other plants, usually has a stem, leaves, roots, and flowers, and produces seeds[Cambridge dictionary] They provide the foundation of many food webs and animal life would not exist if plants were not around. With the development of science, plants were identified as living-thing. It could also respire, reproduce, and even get prone to various diseases. These are different types of diseases by various microorganisms may it be bacteria, viruses, or fungi. Plant diseases can damage crops to a great extent Causing losses in terms of production, both quantitative and qualitative, that clearly affect the farmer's revenues. Generally, in some countries, a large team of experts will classify the disease with the naked eye. In some countries, Farmers are unaware of the disease and lack facilities to contact specialists. Under these conditions, they face many problems, including timing and access to high operating costs. Automatic detection of disease symptoms will improve adequate and cheaper than expert advice.

1.1 Plant leaf diseases

The severity of plant disease ranges from minor damage of plant parts (leaves, stems, fruits) to total plant extinction. They affect plants by reducing their vigor, causing Reduced production quality and quantity, resulting in lower farmer income Reduced availability and higher prices of low-quality food for final consumers. Agricultural production, even leading to famine when attacking important large crops. Famine then leads to human suffering and diseases. People who are undernourished and weak from hunger are easy prey to cholera, pneumonia, stomach disorders, and other infectious diseases and parasites.[1]

Three components are absolutely necessary in order for a disease to occur in any plant system. The three components are:

1. a susceptible host plant
2. a virulent pathogen
3. a favorable environment

If any of the 3 factors is missing disease will not develop.

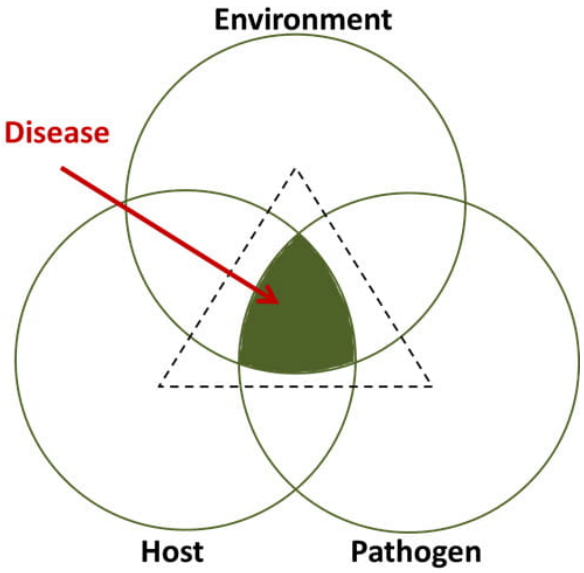

Figure 1.1: Caption

Plant diseases are classified as infectious and non-infectious depending on the nature of a causative agent.

1.1.1 Infectious plant diseases

Infectious plant diseases can be figure as physiological or structural unrest caused by living (biotic) agents called "pathogens". These pathogens can be spread from an infected plant to a healthy plant. The microorganisms that cause plant disease include **nematodes, fungi, bacteria and viruses** . We also classify viruses as biologic because they must There are living cells that can reproduce and are composed of nucleic acids and proteins.[1]

1.1.1.1 Nematodes

nematodes are small worm lives in the soil and feed on plant roots, although some species attacks aboveground plant parts and cause damage to plants by injuring cells, removing cell contents, or

(a)

(b)

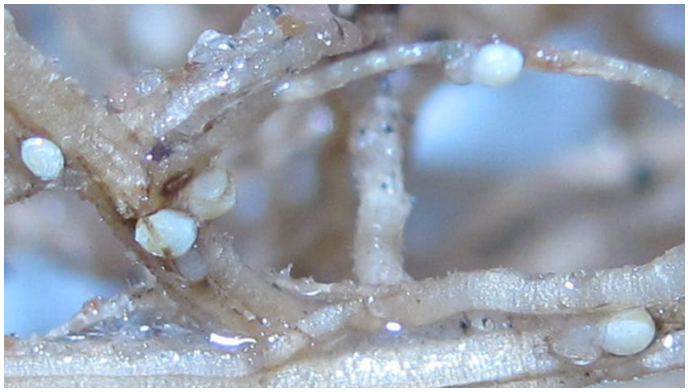

Figure 1.2: Nematodes

changing normal plant growth processes, Their annual global losses are estimated at 100 billion dollars.[1]

1.1.1.2 Fungal Diseases

Fungi are small threadlike organisms composed of tiny filaments They cannot be seen without a microscope. Fungi may evolve into colonies similar to what we see on bread or vegetables, which improperly store for a long time ,examples :(downy and powdery mildew, early and late blight ,Botrytis rots, Anthracnose, and Fusarium). Some fungi grow in plant tissues and destroy it by producing toxins and enzymes ,others develop into large Structures such as mushrooms.[1]

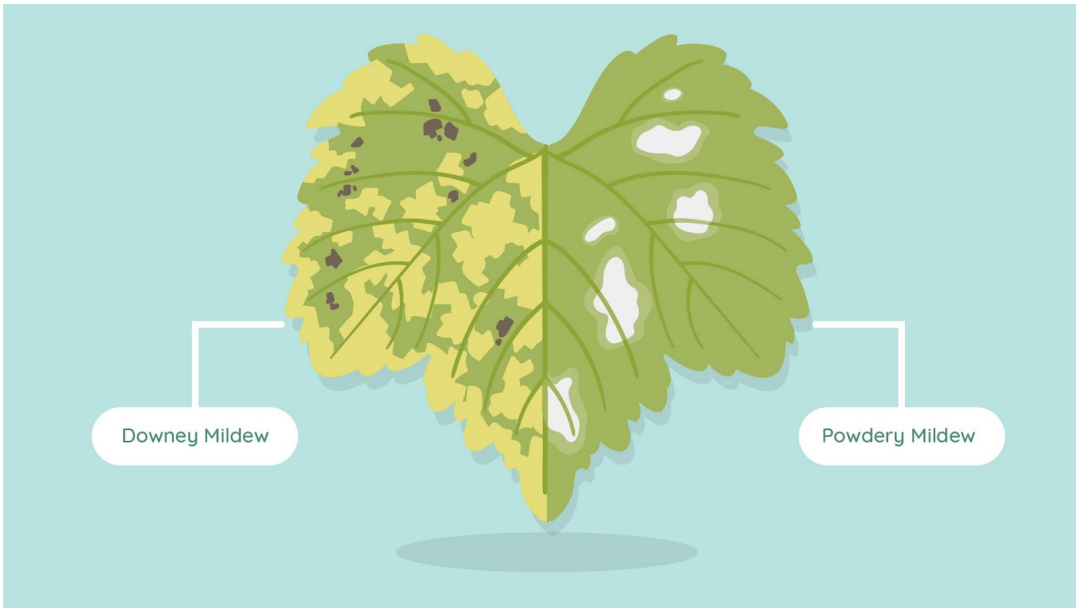

Figure 1.3: Downey Mildew VS Powdery Mildew

Most fungi produce microscopic spores that spread frequently From one plant to another by wind, water or people. The main symptoms of fungal diseases include wilting, spotting, mold, overgrowth, deformations, mummification (shrinkage, darkening, and compaction of the infected tissue), and rot [2]

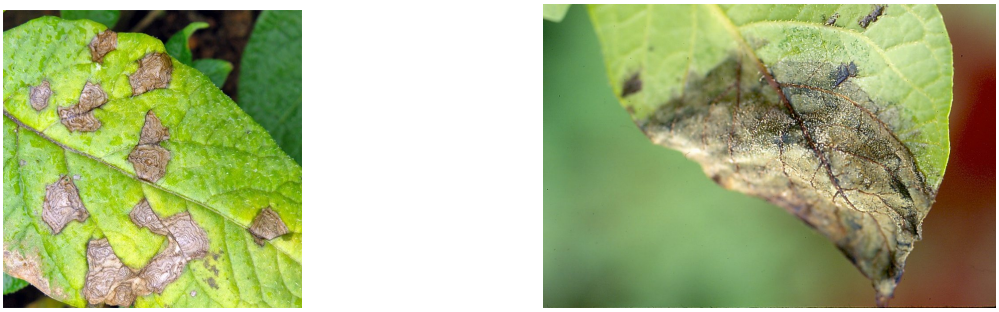

Figure 1.4: Early blight VS Late blight

1.1.1.3 Bacterial diseases

Bacteria are small, single-celled organisms that possess rigid cell walls. Bacterial cells It must be zoomed approximately 400 times to be seen through a light microscope.
Bacteria reproduce by a process called binary fission in which one cell divides into two cells, Under suitable conditions some bacteria can divide about every 30 min, The rapid multiplication of bacteria and the production of toxins and enzymes that destroy plant tissues contribute to the

damage caused by bacteria.[1]
Examples of bacterial diseases : Granville wilt , fire blight, blight of beans, soft rot crown gall, aster yellows and others.

Figure 1.5: Granville Wilt

Figure 1.6: Fire blight

1.1.1.4 Viral Diseases

Viruses are composed of a nucleic acid molecule and a protective protein coat (capsid). The typical size of a plant virus is 30 nm They can be seen only under an electron microscope at Magnifications nearly to 2500 times [3]

The RNA in viruses directs certain enzyme systems in plant cells to manufacture components used to produce more virus particles. In order to diverting energy and structural components, normally used in plant growth, into reproductive processes for the virus. Viruses are spread from one plant to another by insect or nematode vectors or by humans. [1]

The symptoms of viral diseases can be divided as follows:

- deformations (leaf wrinkling, threadlike leaves)
- growth suppression (reduced growth of the entire plant or its leading shoots)
- discoloration (mosaic, leaf chlorosis, variegation)
- necrosis
- impaired reproduction (flower sterility, parthenocarpy, shedding of flowers and ovaries)

In the majority of subtropical and tropical regions, a viral infection can lead to a loss of up to 98% of the crop. [4]

There are many viral diseases among them Tobacco mosaic virus (TMV), Tomato spotted wilt virus (TSWV), Tomato yellow leaf curl virus (TYLCV), Cucumber mosaic virus (CMV), Potato virus Y (PVY) and others.

Figure 1.7: Tomato spotted wilt virus

Figure 1.8: Potato virus Y

1.1.2 Non Infectious diseases

Non Infectious plant diseases are caused by non living (abiotic) agents that cannot be transmitted from one plant to another. [1] These agents are usually:

- Environmental(moisture, soil compaction, ice, and sun scorch)
- nutritional(Calcium deficiency, nitrogen and iron deficiency)
- chemical factors (Chemical injuries due to misuse of insecticides, fungicides, and plant-growth regulators or their use in high or wrong rates).

Figure 1.9: Calcium deficiency

Probably the major causes of poor health in plants are noninfectious diseases and his Symptoms appear suddenly, almost at once ,to their full intensity usually not progressive.

Figure 1.10: Sun scorch

1.2 Tomato production and statistics

World tomato production is 120 Mt, of which one third in Asia, one third in Europe, one third in North America. There are now more than 500 varieties of tomatoes.

World tomato production is steadily increasing, rising from 64 million tonnes in 1988 to more than 100 million today, 30 million of which are intended for processing.

World production has increased by 35% over the last ten years and is distributed as follows: Asia 45%, Europe 22%, Africa 12%, North America 11%, South America South and Central 8%

According to statistics from the Food and Agriculture Organization of the United Nations, world tomato production in 2007 amounted to 126.2 million tonnes for an area of 4.63 million hectares, i.e. a average yield of 27.3 tonnes per hectare.

CHAPTER 2

STATE OF THE ART AND RELATED WORKS

This chapter summarizes the current state of modern machine and deep learning methods as they are applied to practical applications such as classifying various plant diseases. We will begin by explaining many points in this field that aid in the development of machine learning and deep learning models, as well as how to evaluate them. We will then proceed to present and summarize the state of the art by highlighting many works in the same our search.

2.1 Machine Learning

Arthur Samuel defines Machine learning as follows : " A field of study that gives computers the ability to learn without being explicitly programmed."[5]

Machine learning is a scientific field, and more specifically a subcategory of artificial intelligence. It consists of letting algorithms discover “patterns”, namely recurring patterns give computers the ability to learn from data sets, i.e. improve their performance to be able to solve tasks without being programmed already. a data can be numbers, words, images, statistics, Machine learning also helps us find solutions to many problems in vision, speech recognition, and robotics[6]

2.1.1 Machine learning concepts

Some concepts of machine learning needs to be explained first:

2.1.1.1 Splitting data

While constructing the model, the data needs to be splited to avoid overfitting into two three parts, training test,validation and test set

- raining set The portion of data used for training the model, and need to take big part of the split. In this part of the split, the class labels are known for the data objects.
- Validation set The validation set is a set of data, separate from the training set, that is used to validate our model during training. This validation process helps give information that may assist us with adjusting our hyperparameters.
- Test set The test set is a set of data that is used to test the model after the model has already been trained. The test set is separate from both the training set and validation set. The test set should not be labeled.

2.1.1.2 Underfitting and overfitting

A model that would just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on yet-unseen data, this called overfitting. If the model fits perfectly the training data, the predictive accuracy on new data must not necessarily be good. One major problem of machine learning is overfitting , is one the modal perfectly knows how to class the data points during the training, but the mis-classified data during the test is very high. which means that the model is suitable only for the data he was trained on.

2.1.1.3 Underfitting

in the other side when the model isn't complex enough to be able to map and represent the data while training, and can't generalize the new data one solution is proposed to solve this problem is regularization.[7]

2.1.1.4 Regularization

Regularization is the process of generalizing the model and make it more representive to the data, used to prevent overfitting and co-adaptations of model's connection. each algorithm of machine learning should have a regularization parameters, many ways to do regularization will be described the next chapter.[8]

2.1.1.5 Model validation

one more way to prevent overfitting , validation is the task of demonstrating that the model is a reasonable representation of the actual system: that it reproduces system behavior with enough fidelity to satisfy analysis [9] there are many ways to validate the model, here's one of them:

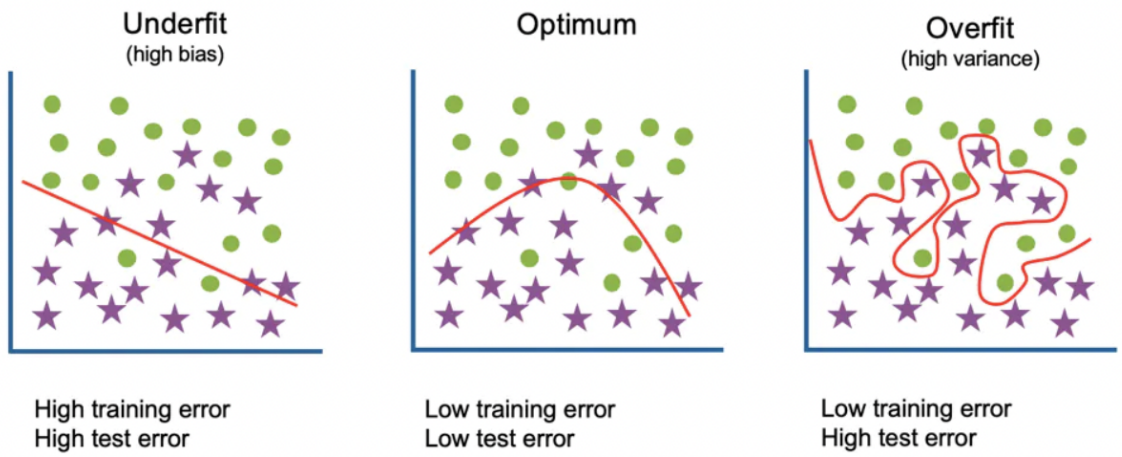

Figure 2.1: Underfitting and overfitting in classification , the third plot in the left is using all the parameters of the model, and this will lead to adapt the exact parameters of the the exact data point and won't be able to generalize new data

---

Algorithm 1: K-fold cross validation algorithm

---

1. Split the dataset into training set and test set
2. The training set is split into k smaller sets
3. A model is trained using K-1 of the folds as training data
4. The remain part of data is used for validation
5. This operation is repeated K times.
6. The performence is measured by the average of values returned in the loop

---

2.1.2 Supervised learning

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example (xi,yi), where xi is the input and yi the corresponding target class [9]

2.1.2.1 SVM

Stands for Support Vector Machines was first proposed by V.Vapnik [10],is one of the most powerful and popular machine learning methods that can be used in either classifications and

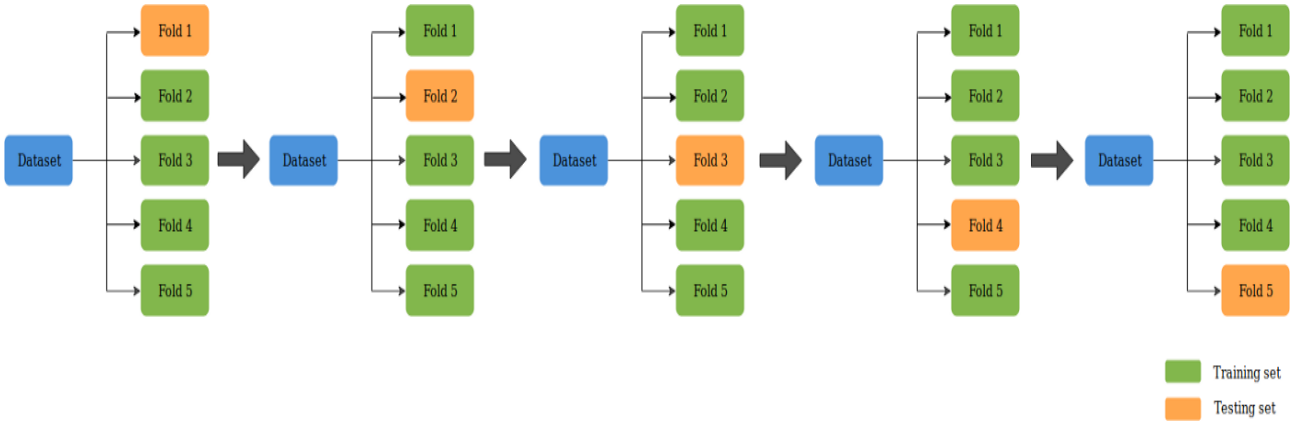

Figure 2.2: Caption

regression tasks, it's commonly used in classification problems. SVM can perform in both linear separable and non separable data , in the second case, SVM uses Kernels to transfer the data-points into a higher dimensional space where they can be linearly separable. SVM simultaneously

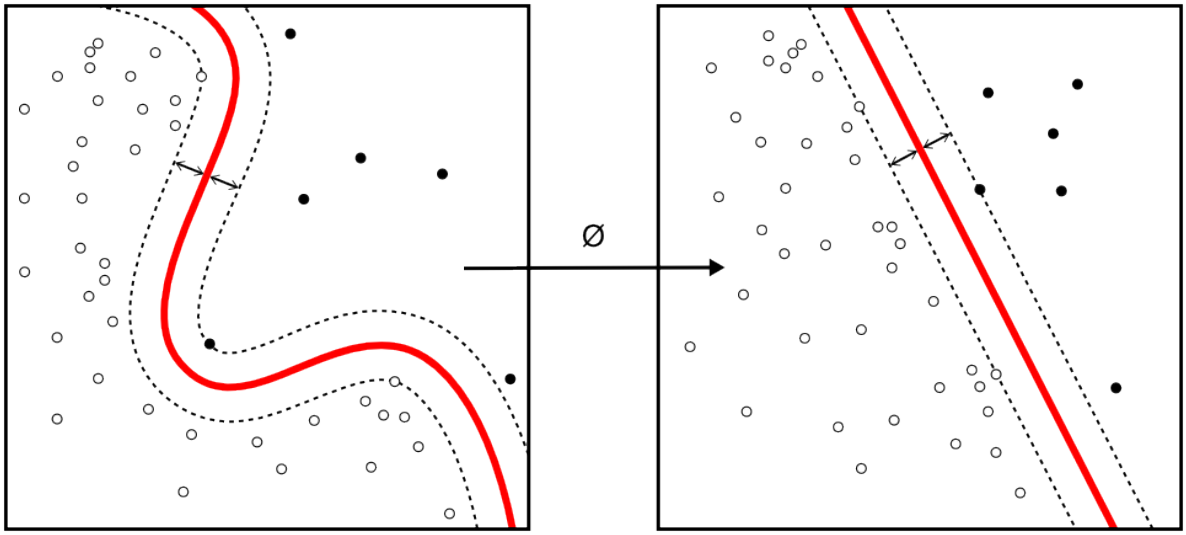

Figure 2.3: SVM in linearly seperable data on the right, the discantious lines are the margins, the data ponints on these lines are called the support vectors, the goal is to maximize these margins, on the left side, the kernel trick of SVM when the data in non-linear seperable.

minimize the empirical classification error and maximize the geometric margin. So SVM called Maximum Margin Classifiers [11] SVM parameters such as kernel parameters and penalty parameter have a great influence on the complexity and performance of predicting models.

Related works using SVM

- Padol, P. B *et al* [12] proposed a system to aid in the detection and classification leaf dis-

eases of grape using SVM classification technique. First the diseased region is found using segmentation by K-means clustering, then both color and texture features are extracted. Finally classification technique(svm) is used to detect the type of leaf disease.

The proposed work focused on recognition and classification of fungal disease like Downy Mildew and Powdery Mildew , total 137 grape leaf images (containing both initial stage as well as final stage images) are used out of which 75 images are Downy leaf images and 62 are Powderly leaf images. For training phase 60 Downy and 50 Powderly images are used and 15 Downy and 12 Powderly are used for testing.
The proposed system can successfully detect and classify the examined disease with accuracy of 88.89

- S. Arivazhagan *et al* [13] proposed an application of texture analysis in detecting and classifying the plant leaf diseases.
  About 500 plant leaves of 30 different native plant species of Tamil Nadu(india) have been collected for this work. the proposed algorithm was tested on ten species of plants namely banana, beans, jackfruit, lemon, mango, potato, tomato, and sapota
  The classification gain obtained by Minimum Distance Criterion is 86.77%. The detection accuracy is improved to 94.74% by SVM classifier

2.1.2.2 Random Forest

Random Forest or random decision forests, it is a supervised classification algorithm consisting of many decisions trees, meaning the approach generate a random forest which is a combination of decision trees models.

Related work using random forest

- S Ramesh *et al* [14] propose a methodology based on the algorithm Random Forest in identifying between healthy and diseased leaf ,this algorithm are flexible in nature and can be used for both classification and regression techniques.
  The algorithm was contrasted with other machine learning models for accuracy. Using Random forest classifier, the model was trained using 160 images of papaya leaves. The model could classify with approximate 70 percent accuracy. The accuracy can be increased when trained with vast number of images Compared to other machine learning techniques like SVM (Accuracy 40.33 %), Naïve bayes (Accuracy 57.61 %), logistic regression(Accuracy 65.33 %), Random forests gave more accuracy (70.14 %), with less number of image data set.

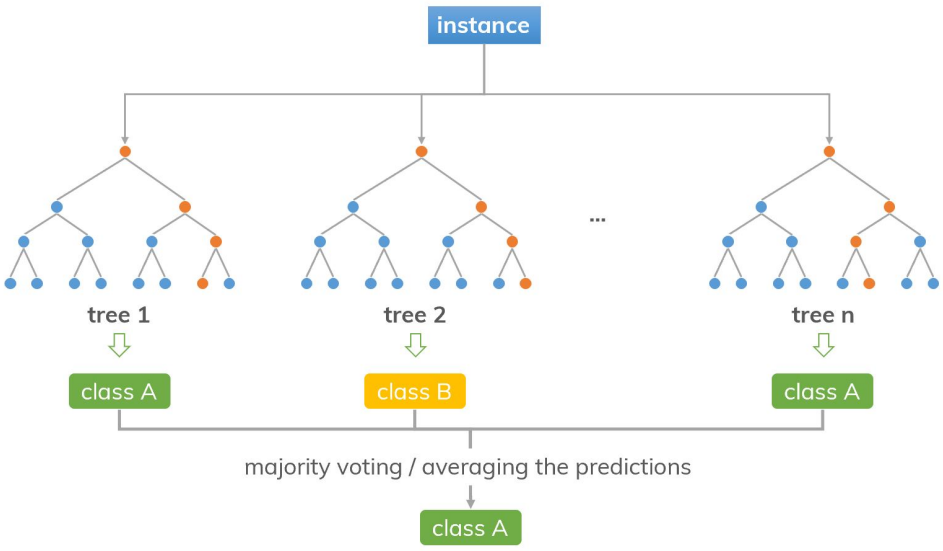

Figure 2.4: Random forest classifier

- CHAUHAN, Ms Deepika, *et al.* [15].The proposed method was applied using image data with a label to train the separation model with various techniques such as KNN(Accuracy 72.16%),NB(Accuracy 74.35%) , DT(Accuracy 73.35%), SVM(Accuracy 76.16%), and RF in the maize disease detection database and finds that the RF(Accuracy 80.68%) classification process is better than other classification methods.
  maize data sets are divided into training data (90%) and test data (10%). The maize plant

  disease database contains a total of 3,823 photographs andfour category labels. The details of the section label information about maize disease data are as follows: gray area, common rust, damage to thenorthern leaves, and healthy 513, 1192, 985, and 1162, respectively.

2.1.2.3 Logistic regression

Is supervised classification algorithm used to classify data to a discrete set of classes(classes are already determined), by using sigmoid function to return a probability value (between 0 and 1) which can be used to map the instance into its corresponding class. If there are many classes to assign, Logistic regression applies One Vs All approach.

One Vs All classification : It is a technique applied by Logistic regression to classify the data ,in the case of many classes should be assign. One-Vs-All is a strategy that involves training a single classifier per class, with the samples of that class as positive samples and all other samples as negatives.

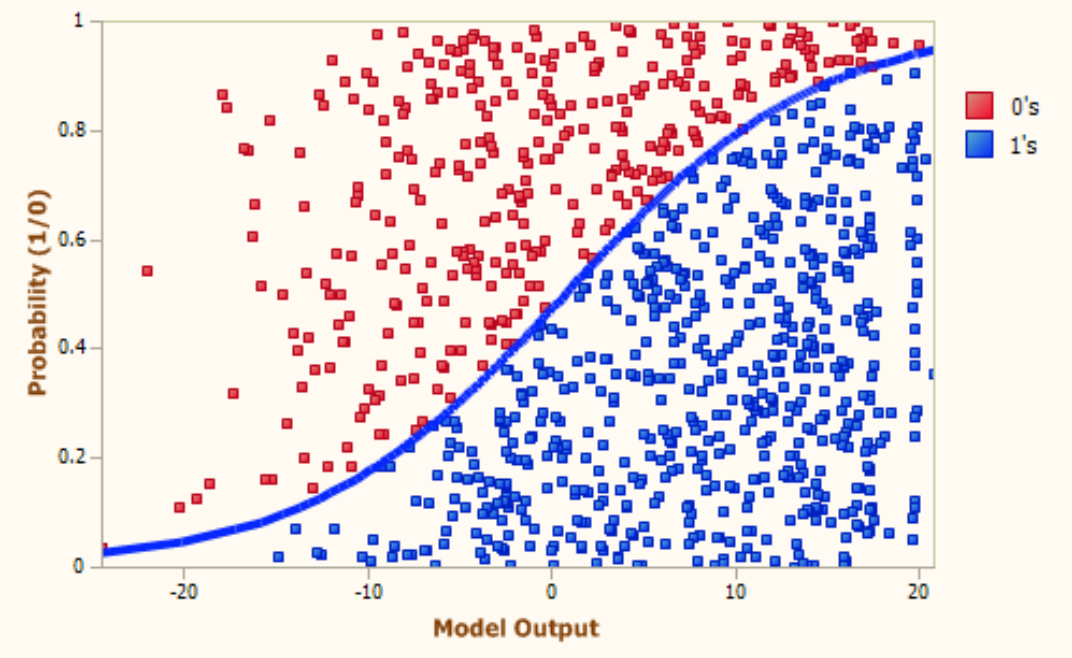

Figure 2.5: Logistic regression example

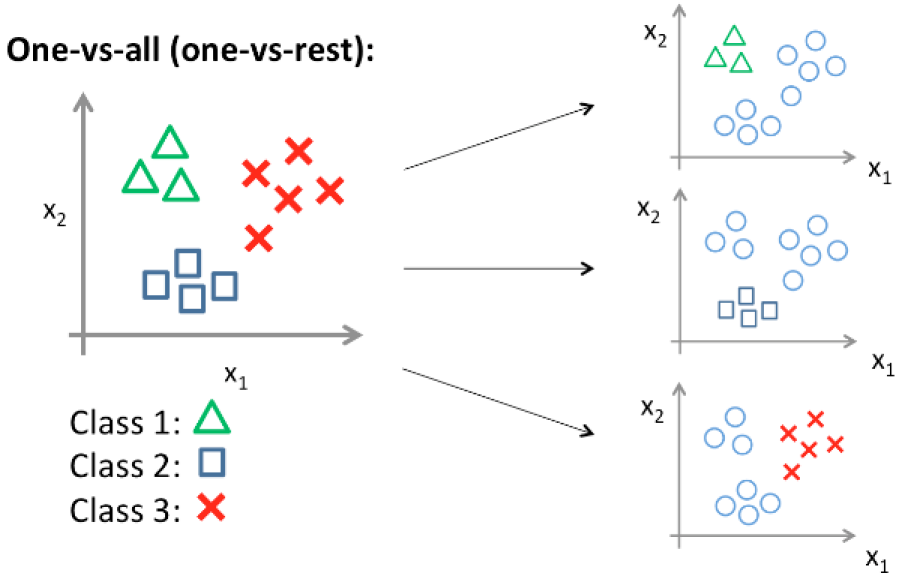

Figure 2.6: One Vs All classification

Related works using Logistic regression :

- According to Mila, A. L *et al* [16] Logistic regression is used widely in epidemiological research, where the binary response usually is the presence or absence of a disease, In this case it was used to estimate the probability of soybean stem rot prevalence with historical disease data from four states of the north-central region of the United States.

  Two models were developed: model I used spring (April) weather conditions and model II used summer (July and August) weather conditions as input variables. Both models had high explanatory power (78.5 and 77.8% for models I and II, respectively).

- D Tiwari *et. al* [17] have used the logistic regression as classifier for a proposed model , 2152 images of potato leaves were taken from a plant village dataset which comprises 1000 images of early blight,1000 images of late blight, and 152 of healthy images of potato leaves. Dataset is divided into two parts: the training part comprises 1700 images(70%) and the test part contains 452 images(30%). Various pre-trained models like inceptionV3 , VGG16 , and VGG19 are used for feature extraction among which VGG19 gave the optimal result.

  Multiple classifiers namely KNN , SVM , Neural Network , and logistic regression are used for classification. Among which Logistic Regression gives the state-of-the-art solution with a classification accuracy of 97.7%

2.1.2.4 KNN

kNN Stands for k Nearest Neighbors, is a popular classification approach. The simple, the new data point is classified according to the classes of the k nearest neighbors to it. The K nearest refers to the similar or close (in term of distance) points to that data, there are many methods to calculate this distance, Euclidian distance , Jaccard distance, Manhattan .. and K refers to the number of neighbors. After calculating the distance from this point to its neighbors, the label of this data point is maintained corresponding to the labels of the k nearest neighbors.[11]

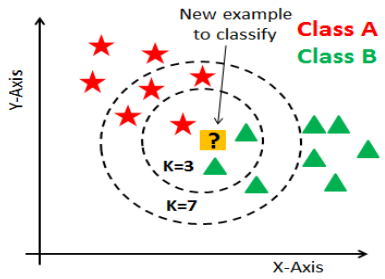

Figure 2.7: Knn example with k=3, k=7

Related work using KNN

- Ramcharan, Amanda, *et al* [18] proposed a method to aid in the detection and classification leaf diseases using KNN classification technique. The method is divided into two phases: the training phase and the testing phase. The training and testing phases comprise of five fundamental stages which are image acquisition, color conversion, color segmentation, morphological operation, and feature extraction. K-nearest neighbor classifier is taken out for color image segmentation.

  The proposed work focused on recognition and classification of five disease like Alternaria alternata, anthracnose, bacterial blight, leaf spot, and citrus canker leaf, total 237 leaf images are used out of which Among these images, total 177 leaf images are Alternaria alternata, anthracnose, and bacterial blight disease affected and each type has 59 leaf images. Other, 60 images are leaf spot and canker affected where each disease type image numbers are 30.

  The proposed method can successfully detect and classify the examined disease with accuracy of 96.76 %.

- Vaishnnave, M. P., *et al* [19] proposed a method to aid in the detection and classification leaf diseases using KNN classification technique. The developmental method of the proposed scheme includes two components Disease Identification: Identification of disease affected. Disease Management: Remedial measure for disease.

  In the selected 250 images, 45 denoised images are trained with KNN classifier and their features are in use for pattern matching; remaining 105 images are used for testing. They categorized only 4 dissimilar disease with of efficiency.

2.1.2.5 Decision Tree

Is a supervised algorithm used for classifications, in some cases can be used for regression, it classifies the input data by sorting them based on attribute or feature values. The decision tree model is composed of nodes, each node represents a feature in an instance that we aim to classify, and each branch represent the value of the node. Instances are classified starting at the root node and sorted based on their feature values. [9]

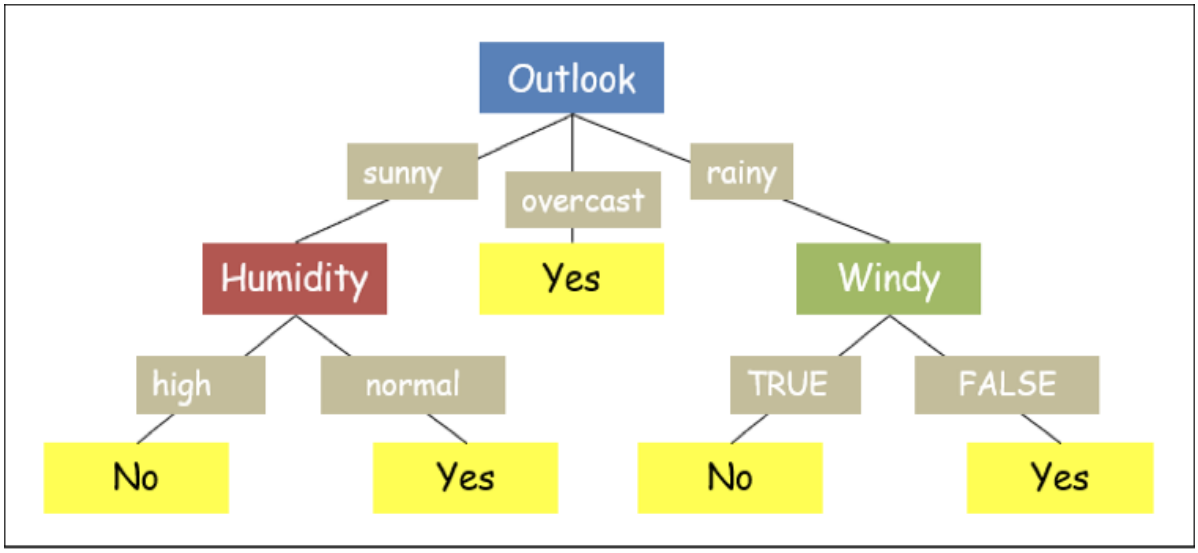

Figure 2.8: An example of a decision tree, this diagram is used by Tom Mitchel In his paper [**?**] he describes his ID3 algorithm, nodes represent attribute values Branches are tests, the last node explain if the person plays tinis or not

Related works using decision tree

- Rajesh, B., M. Vishnu Sai Vardhan, and L. Sujihelen. [20] proposed a system to aid in the detection and classification leaf diseases using Decision Tree classification technique. The main purpose of this proposed system is to detect the leaf disease and identify what type of disease. The proposed work uses a decision tree to identify the leaf disease. The proposed work is based on the morphological characteristics of plant leaves.

  The dataset contains more than 1000 images of the leaf are 145 Healthy images, 200 Late Blight images, 100 Bacterial Spot images, 355 Yellow Curl Virus images and 200 Anthracnose images.

  The proposed system can successfully detect and classify the examined disease with accuracy of 96 % for Tomato and 95 % for Lemon.

- Ahmed, Kawcher, *et al* [21] proposed a system to aid in the detection and classification leaf diseases using Decision Tree classification technique. The main idea of this work is to create a rice leaf disease detection model using machine learning algorithms that can be helpful for disease recognition.

  The dataset has 480 images for three types of rice leaf disease, was divided into two parts: training and test sets where training data contains 432 instances (90% of the 480) and test data contains 48 instances (10% of the 480).

The proposed system can successfully detect and classify the examined disease with accuracy of 94.9074 % for training set and 97.9167 % for testing set.

2.1.2.6 Naïve Bayes

Bayesian classifiers are statistical classifiers. They can predict class membership probabilities, such as the probability that a given sample belongs to a particular class. Bayesian classifier is based on Bayes' theorem. Naive Bayesian classifiers assume that the effect of an attribute value on a given class is independent of the values of the other attributes. This assumption is called class conditional independence. It is made to simplify the computation involved and, in this sense, is considered "naive". [22]

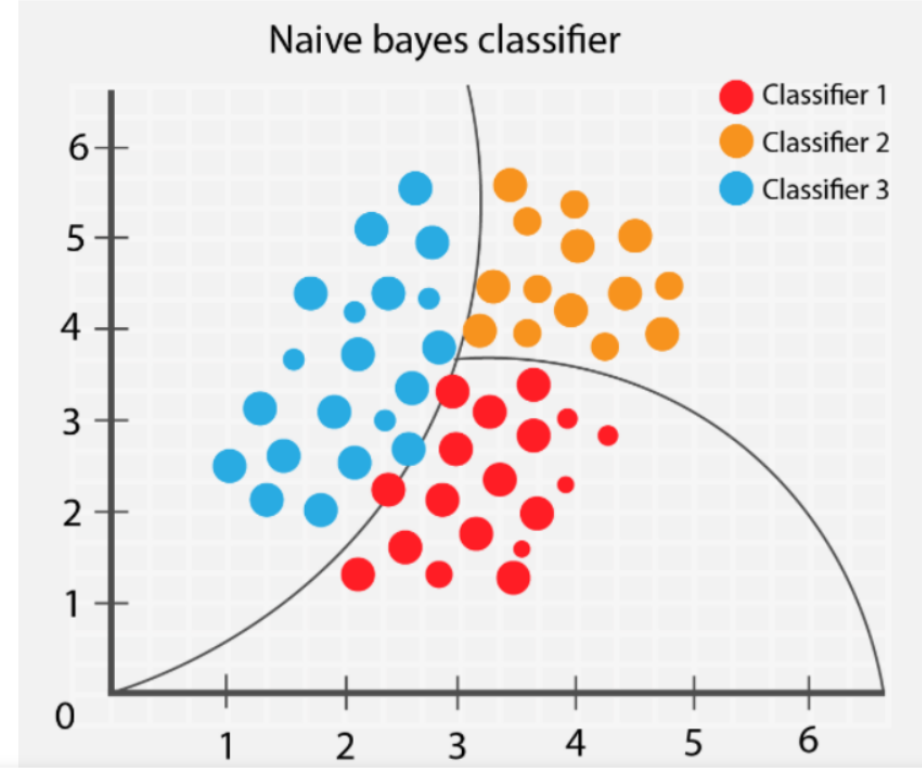

Figure 2.9: Naïve Bayes Classifier

Related Works using Naïve Bayes

- Panigrahi, Kshyanaprava Panda,*et al* [23] proposed a method to aid in the detection and classification leaf diseases using Naïve Bayes classification technique. The proposed method has several components such as image acquisition, image preprocessing, image segmentation, feature extraction, classification, and performance evaluation.

The proposed method can successfully detect and classify the examined disease with accuracy of 77.46 %.

2.1.3 Unsupervised learning

One important type of machine learning when the data is unlabeled and there is no priori identified classes, then it aims to find in the collection of data if there are clusters, groups to be discovered. usually members of same cluster or group are similar to each other and dissimilar to member of other groups. The process is simple , we try to maximize the distance between the clusters and minimize the distance between the members of each cluster. There are different objectives in unsupervised learning problems, such as clustering and visualization. example: Kmeans, Fuzzy c means , HCA. For visualization, the data is projected down, from a high-dimensional space, to two or three dimensions like PCA.[24]

2.2 Deep Learning

Deep learning, as a new area of machine learning research, is a process which allows the computer to learn to perform tasks which are natural for the brain like image recognition. Currently, deep learning (DL) methods have had a profound impact on computer vision and image analysis applications, such as image classification, segmentation, image completion and so on. Deep learning focuses on a specific category of machine learning called Artificial Neural Networks which is inspired by functionality of the human brain. Modern deep learning provides a very powerful framework for supervised learning. By adding more layers and more units within a layer, a deep network can represent functions of increasing complexity. Most tasks that consist of mapping an input vector to an output vector, and that are easy for a person to do rapidly, can be accomplished via deep learning, given sufficiently large models and sufficiently large datasets of labeled training examples. [25]

2.2.1 Feedforward Networks

Also named, Artificial Neural networks, have attracted considerable interest in recent years due to their ability to learn complicated maps from examples, an ability termed universal approximation. called networks because they are typically represented by composing together many different functions Called neural because they are loosely inspired by neuroscience. [25] The learning algorithm must decide how to use those layers to produce the desired output, but the training data does not say what each individual layer should do.

2.2.2 Layers

In this classic artificial neural networks there are many types of layers used in the network, each type of layer is responsible for some computations.

2.2.2.1 Input layer

Input layer is the first layer in the neural network, composed of input neurons and brings initial data to the hidden layers for further processing.

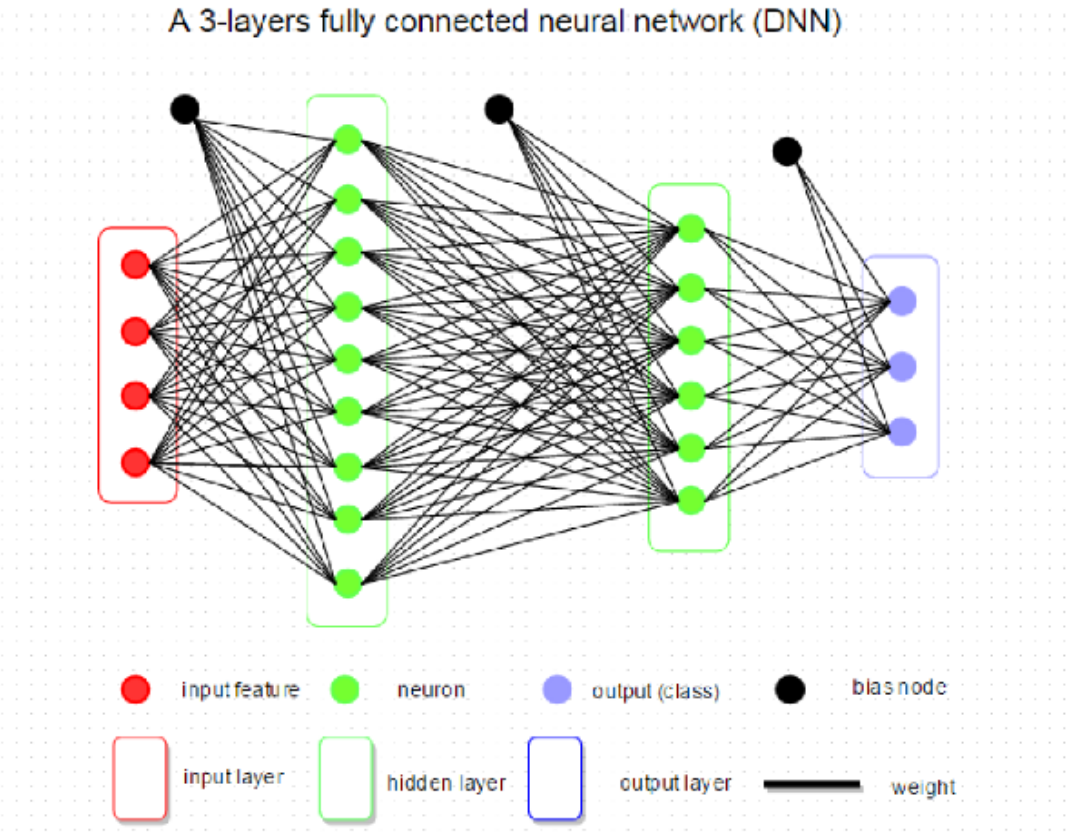

Figure 2.10: Artificial neural networks architecture

2.2.2.2 Hidden layer

The layer or group of layers between the input and output layer. Deep learning is an optimization problem looks for the optimal solution of a very complex problem, many of these computations are made in the hidden layer(s).The choice of hidden layers depends on the complexity of the data, when using a less complex data it's recommended to use few hidden layers, using many hidden layers in a simple problem can lead to overfitting, and using simple architecture in complex

problem leads to underfitting. Each hidden layer of the network is typically vector-valued. The dimensionality of these hidden layers determines the width of the model. [25]

2.2.2.3 Output layer

Output layer is the last layer in neural network which produces the outputs of the program, in classification tasks, the size of output layer is equal to number of classes.

2.2.2.4 Full connected layers (Dense layers)

Different layers in the neural network are connected with each other, in some cases all the neurons of a layer X are connected to neurons to next hidden layer X+1 this is called full connected layers,generally followed by a non-linear activation function. Other types of layers will be explained the next section

2.2.3 Backpropagation

The backpropagation algorithm looks for the minimum of the error function in weight space using the method of some optimization algorithm such as gradient descent. The combination of weights which minimizes the error function is considered to be a solution of the learning problem [26] Backpropagation is considered as the learning algorithm in neural networks, it works as follows, Each node in a chosen layer X have an input value that's equal to the weighted sum of each connection (previous layer) multiplied the previous layers output then pass this result to an activation function, the result obtained from the activation function is the value of the node, then this value is passed as an input to the node of the next layer. This process happens to all the nodes of the layers until reaching the output layer. in classification tasks , each value would correspond to a value that refers a specific class. This process is called Forward propagation. Given the results of the output layers, the algorithm calculates the loss.(explained later), the goal here is to minimize this value so the modal would fit the data properly, gradient descent is used here to minimize the loss, Back propagation is the tool that gradient descent uses to calculate the gradient of the loss function, the process of backpropagation is simple, after calculating the results in the first phase , the gradient descent uses backpropagation to update the weights of the node(neurons) in order to minimize the loss value. [25]

2.2.3.1 Bias and weights

Weight is a number that refers to the strength of a connection between two nodes, Weights are what connect the nodes between layers , the weights are initialized randomly initialized (numbers) mean of 0 and std of 1 each neurons has it own bias , and these biases are learnable meaning that SGD update weights it also updates the bias as well , the bias will determine by

whether or not or how much the neuron will fire throw the network (forward pass), the bias is passed with the SUM of the weights to the activation function

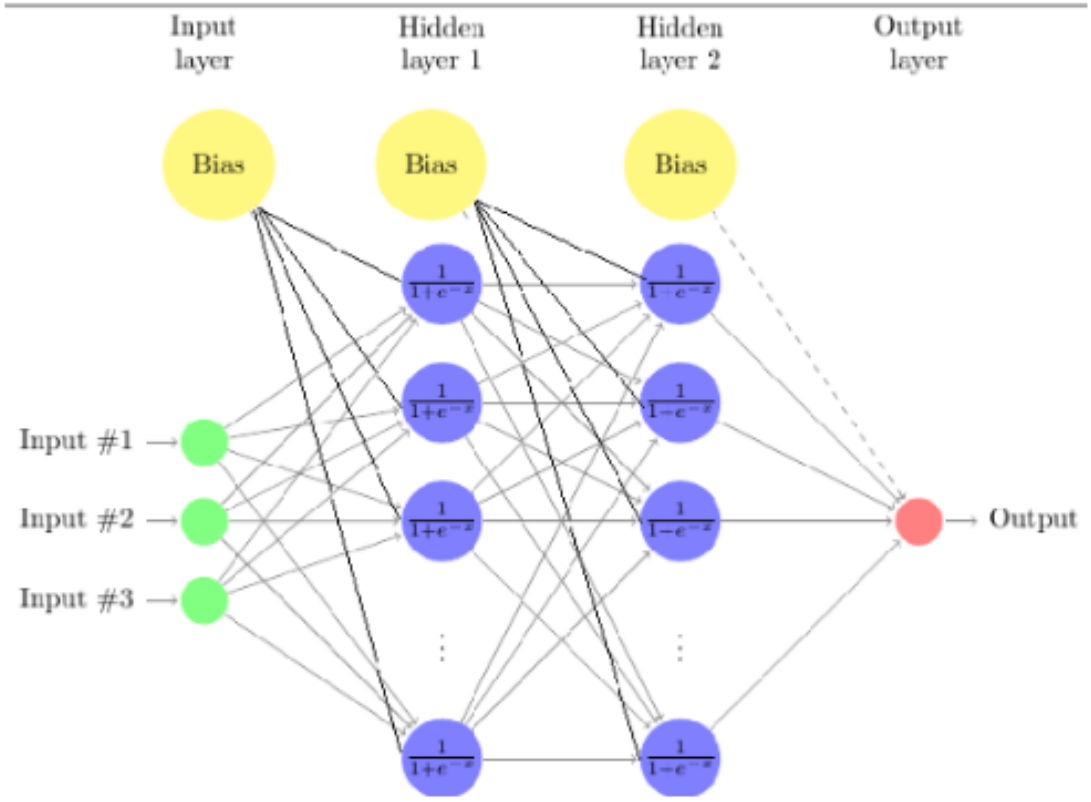

Figure 2.11: Fully connected layers with bias term and Sigmoid activation functions

2.2.4 Optimization in deep learning

The choice of the optimizer in deep learning model can improve the results of the models and reduces the timing from days to hours or minutes. The objective of all optimizers is to reach the global minima where the cost function attains the least possible value. The optimizer in the neural network is a parameter that could make the difference between algorithm converging or exploding. There are a considerable number of optimizers, here's the most common used.

2.2.4.1 Gradient Descent Optimizer

he most popular and classic optimizer used in many optimization tasks , its role in neural networks is to find the optimum values of a cost function. it is an optimization algorithm, based on a convex function, that tweaks it's parameters iteratively to minimize a given function to its local minimum." A gradient measures how much the output of a function changes if you change the inputs a little bit."[25]

2.2.4.2 Stochastic gradient descent , Mini batch gradient descent

Extension of the first described optimizer, is a popular algorithm for training a wide range of different machine learning models such as SVM , ANN and logistic regression. Stochastic gradient descent maintains a single learning rate (termed alpha) for all weight updates and the learning rate does not change during training, and it updates the weights on the connections in each epoch. The motivation behind SGD is that in very large redundant datasets , the gradient on the first half is almost identical to the gradient on the second half. So instead of computing the full gradient, update the weights using the gradient on the first half and then get a gradient for the new weights on the second half. The extreme version of this approach updates weights after each case. It is called Online[27]

Mini batches: according to Goeffery hinton [27] the use of mini batches have proven that it efficielntly helps in optimizing neural networks work better than the online version in SGD, they offer many advantages :

- Less computaion is used updating the weights.
- Computng the gradient for many cases simultaneously uses matrix-matrix muliplies which are very efficient, especially on GPUs

Balanced mini batches: while training the deep model, the weights of each class should be the same, meaning that the number of samples (training examples) of each class in the mini-batch should be the same.

2.2.4.3 Adam

Short to : adaptive moment estimation, Adam is an optimization algorithm that can used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data [28] ,Computationally efficient. Little memory requirements and Well suited for problems that are large in terms of data and/or parameters. Empirical results demonstrate that Adam works well in practice and compares favorably to other stochastic optimization methods [28]

2.2.4.4 RMSprop optimizer

short for RootMean Squar prop. used for speeding the gradient descent optimization process, is an unpublished optimizer proposed by Goeffrey hinton8 in [27] and well recommended by him, appeared after rporp which has proven that it doesn't work well with very large datasets [27] RMSprop gave the ability to deal with larg datasets and works well with mini batches.

2.2.5 Regularization for Deep Learning

Deep neural networks with a large number of parameters and layers are very powerful machine learning systems. However, overfitting is a serious problem in such networks. Large networks are also slow to use, making it difficult to deal with overfitting by combining the predictions of many different large neural nets at test time.

2.2.5.1 Dropout

The key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. During training, dropout samples from an exponential number of different thinned networks. At test time, it is easy to approximate the effect of averaging the predictions of all these thinned networks by simply using a single unthinned network that has smaller weights.[25] This significantly reduces overfitting and gives major improvements over other regularization methods. Dropout is usually only applied after fully connected layers, but not after convolutional layers as it usually increases the test error as pointed out in [29].

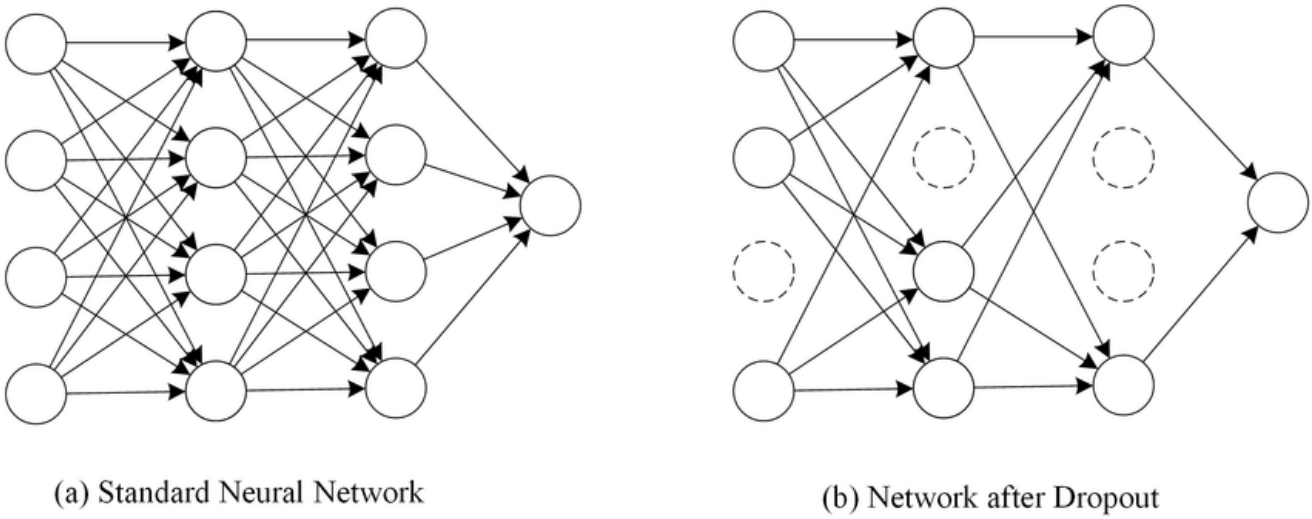

Figure 2.12: Dropout a neural network

2.2.5.2 Batch Normalization

Batch normalization is one of the most exciting recent innovations in optimizing deep neural networks [25], if the batch norm is applied to a layer, it aims to normalize or standarise the ouptut value of the activation function by subtracting the batch mean and dividing by the batch standard deviation, yet Andrew NG [44] recommends to apply batch normalization before the activation function. Batch normalization also allows each layer of a network to learn by itself a

little bit more independently of other layers.[30] [31]

Batch Normalization reduces overfitting because it has a slight regularization effects. Similar to dropout, it adds some noise to each hidden layer's activations. Therefore, while using batch normalization, this means we should use less dropout, which is a good thing because we are not going to lose a lot of information. [31]

2.2.5.3 Early stopping

When training large models with sufficient representational capacity to overfit the task, we often observe that training error decreases steadily over time, but validation set error begins to rise again. Instead of running our optimization algorithm until we reach a (local) minimum of validation error, we run it until the error on the validation set has not improved for some amount of time.[25] When building a deep model, some features are used to perfectly fit the model to the complex data. Generally there is no rule to apply directly these parameteres, the developer, data scientist needs some times to keep changing until the problem is converged. In the case of early stopping, we are controlling the effective capacity of the model by determining how many steps it can take to fit the training set.[32]

2.2.5.4 Data augmentation

Is a method of boosting the size of the training set, usually it works well when the dataset isn't efficiently large to be trained by a deep neural nets, Training deep learning neural network models on more data can result in more skillful models, and the augmentation techniques can create variations of the images that can improve the ability of the fit models to generalize what they have learned to new images. This can take several forms depending of the dataset. Usually done by taking the images and make copies of them by scaling , cropping, rotating these images.[25]

2.2.6 Model parameters

When compiling and fitting the model to a given dataset , some parameters needs to be specified , here are some of them mentioned briefly.

2.2.6.1 learning rate

The learning rate is introduced as a constant (usually very small), in order to force the weight to get updated very smoothly and slowly (to avoid big steps and chaotic behaviour). Geoffery hinton explain how the learning rate parameter is ajusted.[27] If the error keeps going worse or oscillates wildly, reduce the learning rate. If the error is falling fairly consistently but slowly, increase the learning rate. Turn down the learning rate when the error stops decreasing.

2.2.6.2 epoch

One Epoch is when the entire dataset is passed forward and backward through the neural network only once. Epoch parameters refers to how many times the models aims to use the complete dataset for training.

2.2.6.3 Batch

While training, the dataset is divided to a number of batches/parts, big mini-batches are more computationally efficient.

2.2.6.4 Batch size

Total number of training examples present in a single batch.

2.2.6.5 loss functions

Is a performance metric on how well the neural net manages to reach its goal of generating outputs as close as possible to the desired values loss = sum of squares [Desired output-actual output].where the robustness of model increases along with the decrease of the value of loss function. The goal of the model is to minimize this loss

Categorical crossentropy It is a Softmax activation plus a Cross-Entropy loss. If we use this loss, we will train a CNN to output a probability over the C classes for each image. It is used for multi-class classification. Compares the predicted label and true label and calculates the loss.

2.2.6.6 Activation functions

Activation functions are an extremely important feature of the artificial neural networks. They basically decide whether the information(Weight Å bias) received by the neuron(from the previous layer) is enough for the neuron to be activated or not. This transformed output is then sent to the next layer of neurons as input.

Why activation function ? If the model is trained without an activation function, the information shared between neurons in different layers are following a simple linear transformation, while problem solved by the deep neural nets are generally very complicated, this where the activation functions works, they apply a non linear transformation of the input signal. A neural network without an activation function is just a linear regression model. and since the activation function applies a non linear transformations, they help the backpropagation task.

Sigmoid A very popular activation function that has proven solving the two-classes classification task since it produces values between 0 and 1. if the instance value's weight is minthreshold, the class is negative and so on.

ReLU Stands for Rectified Linear Unit, most widely used activation function in the field of neural networks, is a non linear function. The major advantage of ReLU is that it does not activate all the neurons at the same time, meaning that, if the input is negative it will convert it to zero(in other words it returns the highest value between 0 and the weight+bias) and the neuron does not get activated, producing few neurons to be activated at a time.

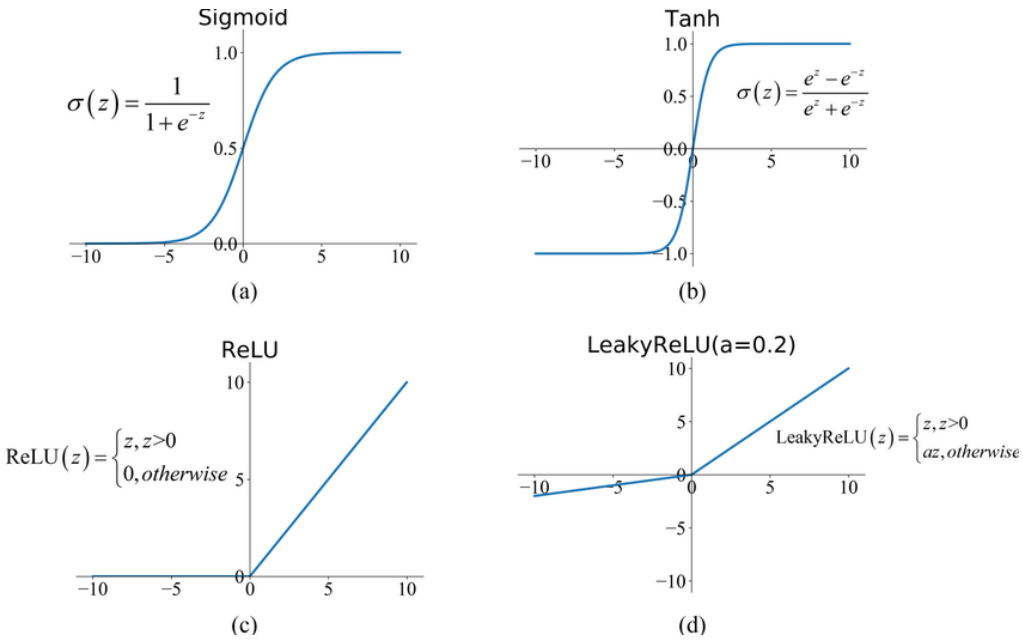

Figure 2.13: Caption

Softmax An enhancement of sigmoid function, it enables the multi classification tasks,the softmax function would squeeze the outputs for each class between 0 and 1 and would also divide by the sum of the outputs. This essentially gives the probability of the input being in a particular class.

TanH Hyperbolic tangent is a non linear transformation function, works similar to the sigmoid function but is symmetric over the origin. it ranges from -1 to 1.

2.3 Convolutional neural network

Proposed by Yann Lecun [33] is considered as the most used type of Artificial Neural Nets in image and video recognition , image classification, computer vision and natural language

processing. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers. Convolutional networks have played an important role in the history of deep learning. They are a key example of a successful application of insights obtained by studying the brain to machine learning applications. They were also some of the first deep models to perform well, long before arbitrary deep models were considered viable.[25]

A typical layer of a convolutional network consists of three stages. In the first stage, the layer performs several convolutions in parallel to produce a set of linear activations. In the second stage, each linear activation is run through a nonlinear activation function, such as ReLU. This stage is sometimes called the detector stage. In the third stage, we use a pooling function to modify the output of the layer further.[25] 3D volumes of neurons. Convolutional Neural Networks take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. In particular, unlike a regular Neural Network, the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth.

Simple ConvNet is a sequence of layers, and every layer of a ConvNet transforms one volume of activations to another through a differentiable function. We use three main types of layers to build ConvNet architectures: Convolutional Layer, Pooling Layer, and Fully-Connected Layer.

2.3.1 Convolution layer

The Convolution layer(or Conv layer) is the core building block of a Convolutional Network that does most of the computational heavy lifting. Most generally, we can think of a CNN as an artificial neural network that has some type of specialization for being able to pick out or detect patterns. This pattern detection is what makes CNNs so useful for image analysis.

Convolutional layer receives input, transforms the input by doing convolution operation, and then outputs the transformed input to the next layer. The inputs to convolutional layers are called input channels, that have three channels (Width , Hight , depth).

With a convolutional layer, the transformation that occurs is called a convolution operation. This is the term that's used by the deep learning community anyway. Mathematically, the convolution operations performed by convolutional layers are actually called cross-correlations. [25]

2.3.1.1 Convolution operation

Convolution is a mathematical operation to merge two sets of information. In our case the convolution is applied on the input data using a convolution filter to produce a feature map. The convolution operation is performed by sliding (i.e convolving) this filter over the image. At every

location, we do element-wise matrix multiplication and sum the result. This sum goes into the feature map. the layers are organized in 3 dimensions: width, height and depth

2.3.1.2 Filters

he Conv layer's parameters consist of a set of filters also known as kernels. the filters are especially designed to detect whether or not the image does contain any such characteristics. Every filter is small spatially (along width and height), but extends through the full depth of the input volume , meaning that it should have the same depth of the input.

2.3.2 Pooling layer

Also called features detector, Its function is to progressively reduce the spatial size of the representation in order to reduce the amount of parameters and computation in the network, hence to control overfitting. In image classifications task [25] , pooling reduces the dimonsiality of the image by the reducing the number of pixels in the output from the previous convolutinal layer.

Pooling helps to make the representation become approximately invariant to small translations of the input. Invariance to translation means that if we translate the input by a small amount, the values of most of the pooled outputs does not change. [25],

Many types of Pooling layers were proposed [34], these are the most common used :

2.3.2.1 Max pooling

uses the maximum value from each cluster of neurons at the prior layer. main goal of max poooling is to capture the main feauters and give them high weights , when these features are found while filtering the image (in the convoloution phase) these feautres are returened (because they have the maximum value) while the no-important features(those who have less value are neglected) [30]

2.3.2.2 Average pooling

uses the average value from clusters of neurons at the prior layer

2.3.3 Padding

As already mentioned in the previous subsection, each convolution layer is applied with a filter that has a dimension, When a filter convolves a given input channel, it gives us an output channel. This output channel is a matrix of pixels with the values that were computed during the convolutions that occurred on the input channel. When this happens, the dimensions of our

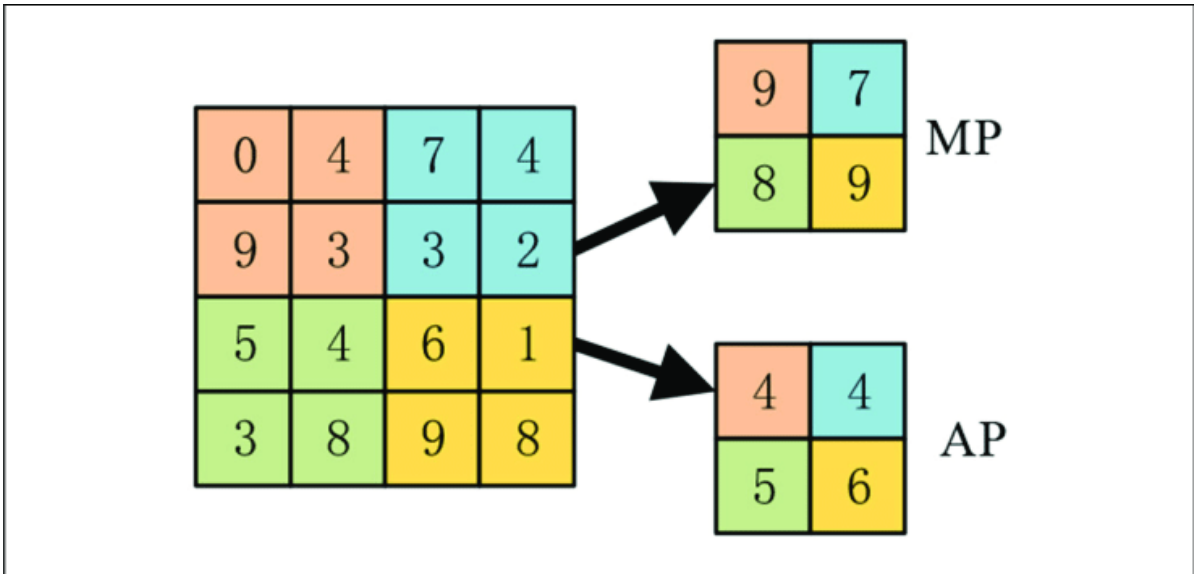

Figure 2.14: Max Pooling Average Pooling

image are reduced.

This means that when this 3 x 3 filter finishes convolving this 4 x 4 input, it will give us an output of size 2 x 2 This issue is resolved by the Zero padding: Zero padding is a technique that allows us to preserve the original input size. With each Conv layer added to the strucutre it is possible to specify whether or not to use padding. Zero padding refers to the addition of border of pixels with zreo values arround the edge of the image.

2.3.3.1 Same Padding

This means that we want to pad the original input before we convolve it so that the output size is the same size as the input size.

2.3.3.2 Valid Padding

This just means no padding. If we specify valid padding, that means our convolutional layer is not going to pad at all, and our input size won't be maintained.

2.4 Attention Mechanism

The idea behind the attention mechanism was to permit the model to utilize the most relevant parts of the input sequence in a flexible manner,by providing an additional focus on a specific component is able to learn the association between them, Attention Mechanism is also an attempt

to implement the same action of selectively concentrating on a few relevant things, while ignoring others in deep neural networks.[35]

2.4.1 Convolutional Block Attention Module (CBAM)

Convolutional Block Attention Module (CBAM) a simple yet effective attention module for feed-forward convolutional neural networks. Given an intermediate feature map, our module sequentially infers attention maps along two separate dimensions, channel and spatial, then the attention maps are multiplied to the input feature map for adaptive feature refinement. Because CBAM is a lightweight and general module, it can be integrated into any CNN architectures[35]

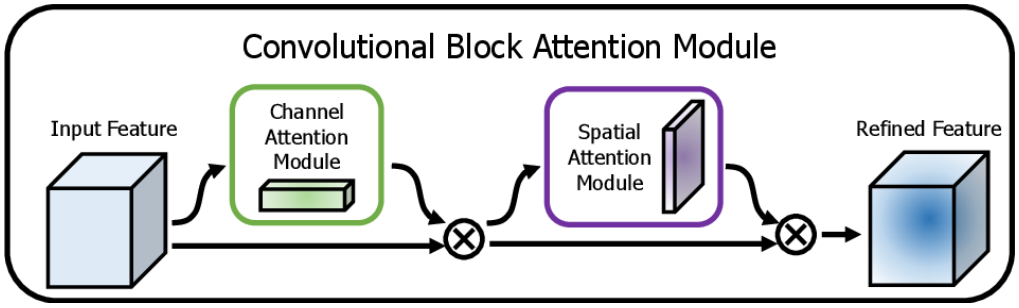

Figure 2.15: Convolutional Block Attention Module (CBAM)

CBAM contains two sequential sub-modules called the Channel Attention Module (CAM) and the Spatial Attention Module (SAM)

2.4.1.1 Channel Attention Module (CAM)

Channel Attention Module (CAM) focuses on 'what' is meaningful given an input image,it decomposes the input tensor into 2 subsequent vectors of dimensionality (c × 1 × 1). One of these vectors is generated by GAP while the other vector is generated by Global Max Pooling (GMP). Average pooling is mainly used for aggregating spatial information, whereas max pooling preserves much richer contextual information in the form of edges of the object within the image which thus leads to finer channel attention. Simply put, average pooling has a smoothing effect while max pooling has a much sharper effect.[35]

2.4.1.2 Spatial Attention Module (SAM)

Spatial Attention Module (SAM) focuses on 'where' is an informative part. is comprised of a three-fold sequential operation. The first part of it is called the Channel Pool, where the Input Tensor of dimensions (c × h × w) is decomposed to 2 channels, i.e. (2 × h × w), where each of the 2 channels represent Max Pooling and Average Pooling across the channels. This serves as the

input to the convolution layer which output a 1-channel feature map, i.e., the dimension of the output is (1 × h × w).[35]

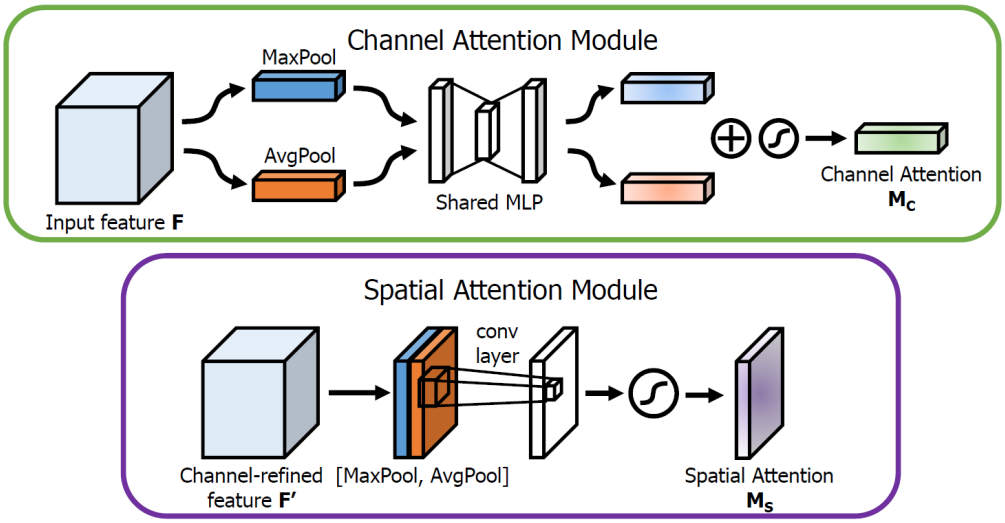

Figure 2.16: Channel Attention Module (CAM) Spatial Attention Module (SAM)

2.4.2 Gradient-weighted Class Activation Mapping (Grad-CAM)

For the qualitative analysis, we apply the Grad-CAM [36] is a technique for producing visual explanations for decisions from a large class of CNN-based models, making them more transparent. Grad-CAM is a recently proposed visualization method which uses gradients in order to calculate the importance of the spatial locations in convolutional layers. As the gradients are calculated with respect to a unique class, Grad-CAM result shows attended regions clearly. By observing the regions that network has considered as important for predicting a class, we attempt to look at how this network is making good use of features.

2.5 Transfer learning

The term of transfer learning is related to use a pretrained model, regarding the weights and the parameters, and fine tune it to solve a specific image classification task.

2.5.1 Transfer learning VS traditional learning

Traditional learning methods build a new classifier from scratch for each classification task.Transfer learning applies knowledge from a source classifier to simplify the construction of a classifier for a new, target tasks. [11]

2.5.2 ImageNet

ImageNet is an image database that contains 14,197,122 images and more than 1,000 classes, and can be freely accessed in ImageNet website [37] Since 2010, the ImageNet project runs an annual software contest, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), where software programs compete to Correctly classify and detect objects and scenes.

2.5.3 Pretrained Model

here we presents some famous pretrained model on ImageNet dataset and their architecture. In the pretrained model the weights and parameters of a network that has been trained on a imageNet dataset and saved for further use of image classification tasks.

2.5.3.1 VGG-16

Stands for Visual Geometry Group with pushing the depth to 16 weight layers ,is a pretrained VERY DEEP Convolutional networks for large-scale image recognition thatwas proposed by the VGG group [38] ,University of Oxford, while their participation in ImageNet LSVRC-2014 where they secured the first and second place for image classification and localization. The model contains 138M parameters and 294,9112 weights.

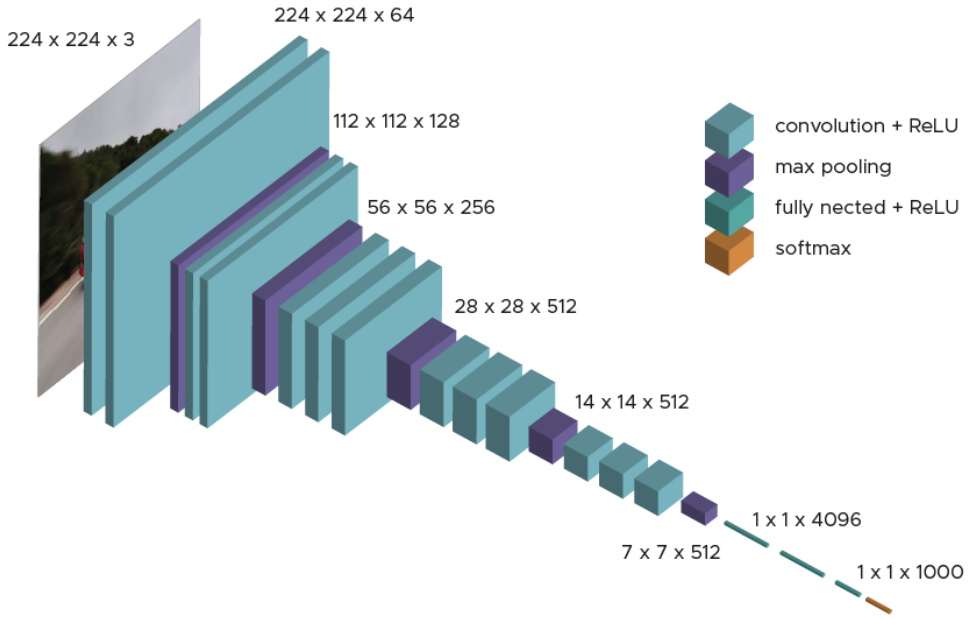

Figure 2.17: VGG16 architecture

2.5.3.2 VGG-19

The visual geometry group network (VGGNet) is a deep neural network with a multilayered operation. The VGGNet is based on the CNN model and is applied on the ImageNet dataset. VGG-19 is useful due to its simplicity as 3×3 convolutional layers are mounted on the top to increase with depth level [39] The number 19 stands for the number of layers with trainable weights, 16 Convolutional layers and 3 Fully Connected layers.

2.5.3.3 Inception

The Inception deep convolutional architecture was introduced as GoogLeNet in (Szegedyet al.)[40], here named Inception-v1. Later the Inception architecture was refined in various ways, first by the introduction of batch normalization (Ioffe and Szegedy 2015) (Inception-v2). Later by additional factorization ideas in the third iteration (Szegedyet al. 2015b) which will be referred to as Inception-v3 in this report.is a pretrained model that contains 7 million parameters and 42-layer deep learning network.

2.5.3.4 Xception

The Xception architecture [41], introduced by Francois Chollet, is an extension of the Inception architecture. This architecture is a linear stack of depthwise separable convolution layers with residual connections.

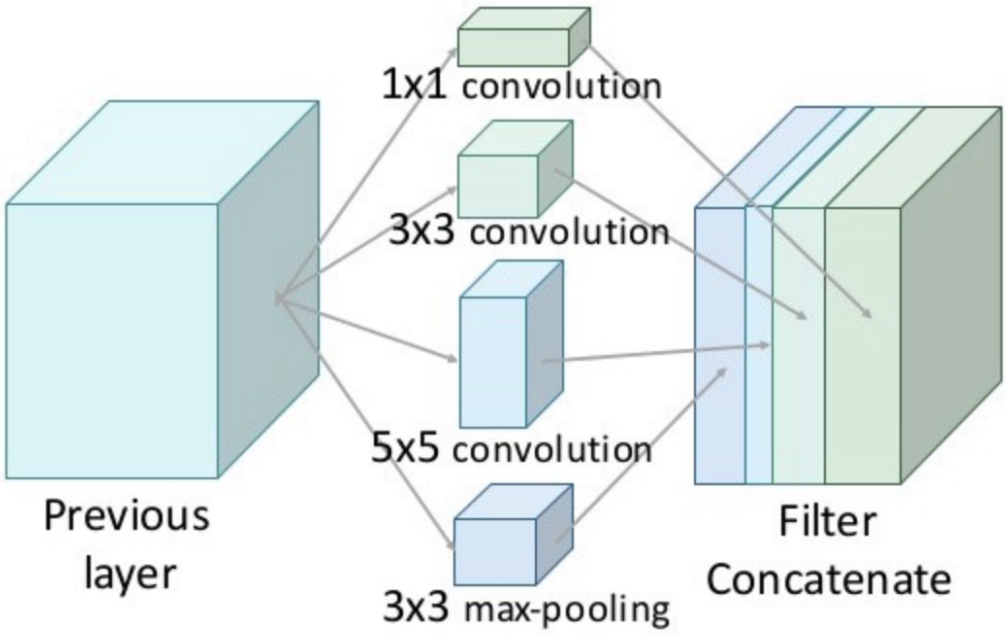

Figure 2.18: Xception architecture

2.5.3.5 ResNet

ResNet is a short name for Residual Network proposed in [42]. As the name of the network indicates, the new terminology that this network introduces is residual learning. Residual nets with a depth of up to 152 layers—8x deeper than VGG nets but still having lower complexity. An ensemble of these residual nets achieves 3.57 error on the ImageNet test set. This result won the 1st place on the ILSVRC 2015 classification task.

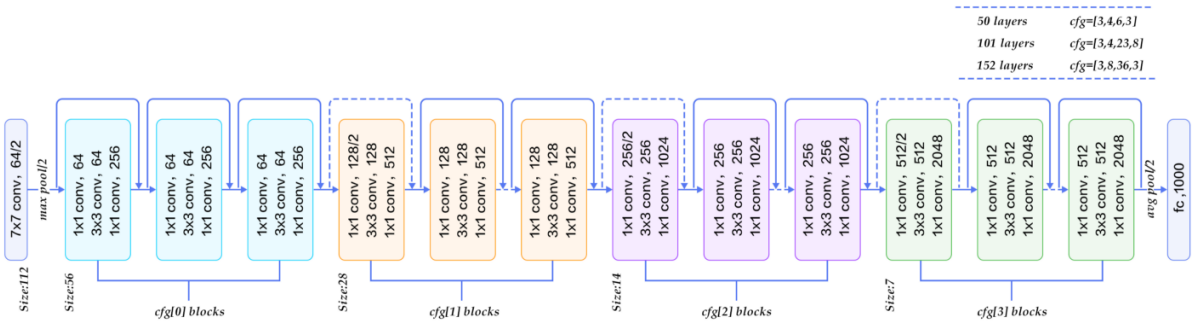

Figure 2.19: ResNet architecture

2.5.3.6 MobileNet V2

MobileNetV2 is a convolutional neural network architecture that seeks to perform well on mobile devices. It is based on an inverted residual structure where the residual connections are between the bottleneck layers. The intermediate expansion layer uses lightweight depthwise convolutions to filter features as a source of non-linearity. As a whole, the architecture of MobileNetV2 contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers. [43]

2.5.3.7 EfficientNet

EfficientNet is a convolutional neural network architecture and scaling method that uniformly scales all dimensions of depth/width/resolution using a compound coefficient to scale up models in a simple but effective manner. Instead of randomly scaling up width, depth or resolution, compound scaling uniformly scales each dimension with a certain fixed set of scaling coefficients. Using the scaling method and AutoML, the authors of efficient developed seven models of various dimensions, which surpassed the state-of-the-art accuracy of most convolutional neural networks, and with much better efficiency. EfficientNet is based on the baseline network developed by the neural architecture search using the AutoML MNAS framework The architecture uses a mobile inverted bottleneck convolution similar to MobileNet V2. EfficientNets have been the State-of-the-art for high quality and quick image classification. [44] The number of layers in EfficientNet-B0 is 237 and in EfficientNet-B7 the total comes out to 813!

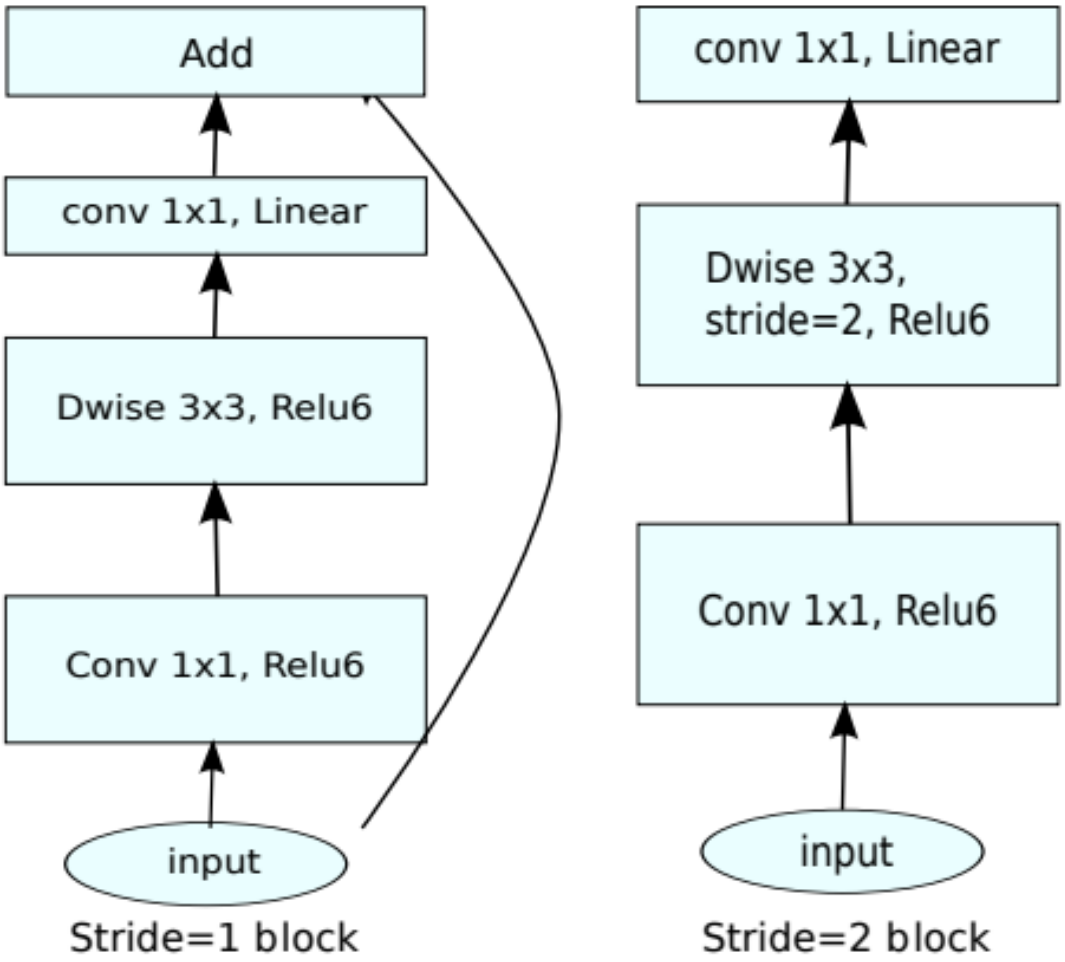

Figure 2.20: MobileNetV2 architecture

2.5.3.8 DenseNet

DenseNet (Dense Convolutional Network) is an architecture that focuses on making the deep learning networks go even deeper, but at the same time making them more efficient to train,that utilises dense connections between layers, through Dense Blocks, where we connect all layers (with matching feature-map sizes) directly with each other,the first layer is connected to the 2nd, 3rd, 4th for example,DenseNets have several compelling advantages: they alleviate the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters.[45]

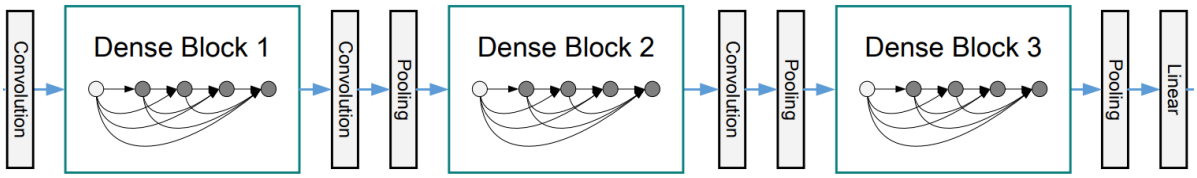

Figure 2.21: DenseNet architecture

2.5.3.9 Fine-tuning

Fine-tuning, in general, means making small adjustments to a process to achieve the desired output or performance. Fine-tuning deep learning involves using weights of a previous deep learning algorithm for programming another similar deep learning process. Weights are used to connect each neuron in one layer to every neuron in the next layer in the neural network. The fine-tuning process significantly decreases the time required for programming and processing a new deep learning algorithm.

2.6 Vision Transformer

The Vision Transformer, or ViT, is a model for image classification that employs a Transformer-like architecture over patches of the image. An image is split into fixed-size patches, each of them are then linearly embedded, position embeddings are added to retain positional information, and the resulting sequence of vectors is fed to a standard Transformer encoder. In order to perform classification, the standard approach of adding an extra learnable “classification token” to the sequence is used, Internally, the transformer learns by measuring the relationship between input token pairs,Recently, Vision Transformers (ViT) have achieved highly competitive performance in benchmarks for several computer vision applications, such as image classification, object detection, and semantic image segmentation.[46]

The overall architecture of the vision transformer model is given as follows in a step-by-step manner [47]:

1. Split an image into patches (fixed sizes)
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 model with image labels, which is then fully supervised on a big dataset

7. Fine-tune on the downstream dataset for image classification

The transformer encoder includes [47]:

- **Multi-Head Self Attention Layer (MSP):** This layer concatenates all the attention outputs linearly to the right dimensions. The many attention heads help train local and global dependencies in an image.

- **Multi-Layer Perceptrons (MLP) Layer:** This layer contains a two-layer with Gaussian Error Linear Unit (GELU).

- **Layer Norm (LN):** This is added prior to each block as it does not include any new dependencies between the training images. This thereby helps improve the training time and overall performance.

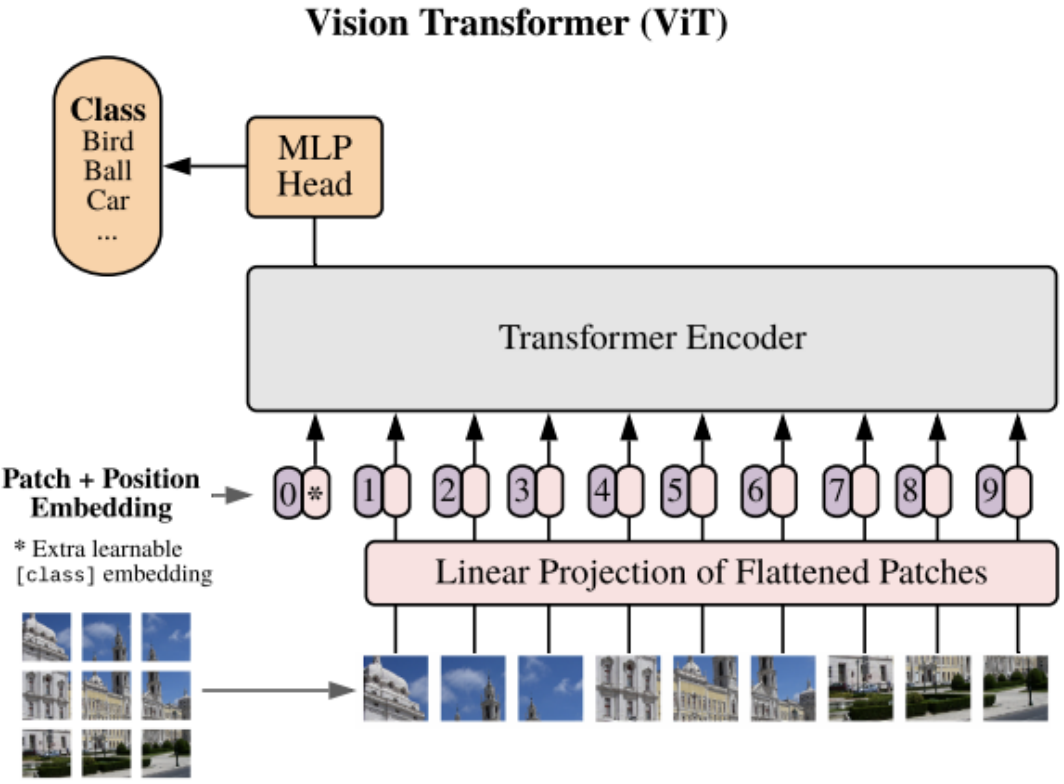

Figure 2.22: Vision Transformer architecture

2.7 Related work using deep learning

This section mentions the previous works dealing with classifying Plant leaf diseases images using deep convolutional neural nets and pretrained models.

- Sardogan *et al* [48] have proposed a very deep CNN network architecture contained 4 CONV blocks and 1 FC layers. Max pooling was placed after the 2nd, 3rd and the 4th CONV blocks.A 9x9 filter has been used in the first and second convolutions, and a 5x5 filter has been used in the third and fourth convolutions. After these implications, three different 3x3 matrices have been obtained for R, G and B channels separately. The FC layer included 27 neurons. A Relu layer was placed at the end of each CONV.
  The experiments showed that the average prediction accuracy of CNN-based method was 86%.

- Mohanty *et al* [49] evaluate the applicability of deep convolutional neural networks for the classification problem. Focusing on two popular architectures, namely AlexNet [50], and GoogLeNet (Szegedy et al., 2015). In case of transfer learning, they re-initialize the weights of layer fc8 in case of AlexNet, and of the loss 1,2,3/classifier layers in case of GoogLeNet. Then, when training the model, they do not limit the learning of any of the layers, as is sometimes done for transfer learning. In other words, the key difference between these two learning approaches (transfer vs. training from scratch) is in the initial state of weights of a few layers, which lets the transfer learning approach exploit the large amount of visual knowledge already learned by the pre-trained AlexNet and GoogleNet models extracted from ImageNet [40]. To summarize, they have a total of 60 experimental configurations.
  The trained model achieves an accuracy of 99.35% on a held-out test set.

- Chen, J *et al* [51] inspired by the performances, the VGGNet with the Inception module is selected in our approach. The conventional VGGNet was enhanced by replacing its last convolutional layers with an additional convolutional layer of $3 \times 3 \times 512$, then the batch normalization (BN) was added on the convolutional layer and the Swish activation function is used to directly replace ReLU. Furthermore, they modified the conventional VGGNet by replacing its full connection layers with a global pooling layer, and two Inception modules, as stated before, are introduced in the VGGNet to improve the feature extraction ability of the new network, namely, INC-VGGN. and it is of little value to utilize Inception to extract these features. Therefore, the Conv1-1 to Pool3 layers of the VGGNet are preserved and the subsequent layers of VGGNet, Conv5-1 to Conv5-3, are replaced with two Inception modules and one added BN convolutional layer, in which the Swish is used as the activation function instead of the ReLU. a convolutional layer with batch normalization and Swish is added behind Pooling3, and then the two Inception modules composed of Inception and Concat are conducted to enhance the multi-scale feature extraction ability of the network.Finally,

a global pooling layer instead of fully-connected layers are added after the second Inception module and followed by a Softmax classifier.
The experiments showed that the average prediction accuracy was no less than 91.83% on the public dataset.

- Another paper on classifying Plant-Leaf Diseases using deep learning methods by Liu, B *et al* [52] deep convolutional neural network model is proposed to identify apple leaf diseases. they proposed an architecture where the first convolutional layer is designed to be 96 kernels of size 9 × 9 × 3, which is different from the Symmetry 2018, 10, 11 7 of 16 first convolutional layer's kernel size of 11 × 11 × 3 in the standard AlexNet. The second convolutional layer filters the noise with 256 kernels of size 5 × 5 × 48, response-normalization layers follow the first two convolutional layers, which are themselves followed by max-pooling layers. The third convolutional layer has 384 kernels with a size of 3 × 3 × 256 connected to the (normalized, pooled) outputs of the second convolutional layer. The fourth layer is filtered with 384 kernels of size 3 × 3 × 192, and the fifth layer has 256 kernels with a size of 2 × 2 × 192 to improve the ability to extract small features, which is also different from the standard AlexNet, and is then followed by a max-pooling layer. After AlexNet Precursor, an architecture named Cascade Inception is designed including two max-pooling layers and two Inception structures. The first max-pooling layer is applied to filter the noise of feature maps generated by AlexNet Precursor, and the two Inceptions then extract the optimal discrimination features from multidimension analysis. Feature maps before the first Inception are input into the second Inception's concatenation layer, which prevents some of the features being filtered by these two Inceptions. Meanwhile, the sixth convolutional layer followed by the Cascade Inception has 4096 kernels with a size of 1 × 1 × 736, which replaces the first two fully connected layers of the standard AlexNet. The fully connected layer is adjusted to predict four classes of apple leaf diseases, and the final layer is a four-way Softmax layer.
  the experimental results show that the proposed disease identification approach based on the convolutional neural network achieves an overall accuracy of 97.62%.

- Published By Brahimi M, Boukhalfa K and Moussaoui A [53] Pre-taining phase: in this phase they train deep architectures on a large dataset like ImageNet using powerful machines. The objective of this phase is the initialization of network weights for the next phase. Training (fine-tuning): they fine-tune the resulted network from the first phase. Also, they replace the output layer of the pre-tained networks (ImageNet contains 1000 classes) by a new output layer having nine classes (nine diseases of tomato). They have used two CNN models (AlexNet (Alex, Sutskever, and Hinton 2012) and GoogleNet (Szegedy et al. 2015)).
  The obtained results are encouraging, reaching 99.18% of accuracy

2.7.1 Related works summarized

Paper	Year	Data	Classification Methods	Accuracy
[12]	2016	Grape leaves images	SVM	88.89 %
[13]	2013	9 Types of plants(banana, beans, jackfruit, lemon, mango, potato, tomato, guava and sapota)	SVM	94.74 %
[14]	2018	Papaya leaves images	Random Forest	70.14 %
[15]	2021	Maize Leaf Disease Dataset	Random Forest	80.68 %
[16]	2004	Soil texture, weather variables and tillage practices	Logistic Regression	78.5 and 77.8 %
[17]	2020	PlantVillage dataset	Logistic Regression	97.7 %
[18]	2017	Original cassava dataset	KNN	96.76 %
[19]	2019	Groundnut leaves images	KNN	/

[20]	**2020**	**Plant leaf images (1000 images)**	**Decision Tree**	**96 %**
[21]	**2019**	**Rice leaf images(480 images)**	**Decision Tree**	**97.9167 %**
[23]	**2020**	**Original cassava dataset**	**Naïve Bayes**	**77.46 %**
[48]	**2018**	**Tomato leaves images 500**	**CNN**	**86 %**
[49]	**2016**	**PlantVillage dataset**	**CNN, AlexNet and GoogleNet**	**99.35 %**
[51]	**2020**	• **Public dataset: plantVillage dataset** • **collected dataset: rice and maize image datasets**	**Cnn(VGGNet)**	**91.83 %**
[52]	**2017**	**13,689 diseased apple leaves images**	**CNN(AlexNet)**	**97.62 %**
[53]	**2017**	**Images of tomato leaves from plantVillage**	**CNN,AlexNet and GoogleNet**	**99.18 %**

Table 2.1: List of various Related Works and its Results

2.8 Object Detection

Object detection involves detecting instances of objects from one or several classes in an image. [54]

2.8.1 YOLO

YOLO an acronym for 'You only look once', is an object detection algorithm that divides images into a grid system. Each cell in the grid is responsible for detecting objects within itself.

YOLO is one of the most famous object detection algorithms due to its speed and accuracy. [55]

YOLOv5 is a family of object detection architectures and models pretrained on the COCO dataset, and represents Ultralytics open-source research into future vision AI methods, incorporating lessons learned and best practices evolved over thousands of hours of research and development. [56]

2.9 Summary

In this chapter we talked about machine learning concepts and algorithms followed by a related works corresponding to our thesis subject, then we mentioned deep learning by explaining some concepts and techniques like convolution neural network, transfer learning and Vision Transformer also followed by a related works on plant leaf diseases.

CHAPTER 3

DATASET AND IMPLEMENTATION TOOLS

This chapter will describe our dataset, followed by the pre-processing phase, as well as the implementation tools and frameworks.

3.1 Dataset description

In this project we have used two datasets:

3.1.1 plantvillage

is a general dataset and contains several types of plant leaves,in this data-set, 39 different classes of plant leaf and background images are available. The data-set containing 61,486 images of 14 different plants (infected by different diseases). We split the dataset into training set with 43,444 images (80%) and validation set with 10,861 images (20%)

3.1.2 Tomato

is a specific dataset about tomato found it in the famous website kaggle [57] The dataset consists of over 10,000 tomato leaves images, is organized into 2 folders (train with 10000 images and 1000 images for validation) and contains subfolders for each diseases category (9 types of diseases and one healthy) there are : **Bacterial spot** ,**Early blight**, **Late blight**,**Leaf Mold**, **Septoria leaf spot**,**Spider mites**, **Target Spot**,**mosaic virus**,**Yellow Leaf Curl Virus**,**healthy** , The original size of the images is 256x256, our dataset is already augmented using techniques like rotation, flip and Brightness that's helped to get more number of parameters to learn almost all the features from the data.

Class name	Images
1. Apple Scab, Venturia inaequalis	630
2. Apple Black Rot, Botryosphaeria obtusa	621
3. Apple Cedar Rust, Gymnosporangium juniperivirginianae	275
4. Apple healthy	1645
5. Blueberry healthy	1502
6. Cherry healthy	854
7. Cherry Powdery Mildew, Podoshaera clandestine	1052
8. Corn Grey Leaf Spot, Cercospora zeae-maydis	513
9. Corn Common Rust, Puccinia sorghi	1192
10. Corn healthy	1162
11. Corn Northern Leaf Blight, Exserohilum turcicum	985
12. Grape Black Rot, Guignardia bidwellii	1180
13. Grape Black Measles (Esca), Phaeomoniella aleophilum, Phaeomoniella chlamydospora	1383
14. Grape Healthy 423 15. Grape Leaf Blight, Pseudocercospora vitis	1076
16. Orange Huanglongbing (Citrus Greening), Candidatus Liberibacter spp.	5507
17. Peach Bacterial Spot, Xanthomonas campestris	2297
18. Peach healthy	360
19. Bell Pepper Bacterial Spot, Xanthomonas campestris	997
20. Bell Pepper healthy	1478
21. Potato Early Blight, Alternaria solani	1000
22. Potato healthy	152
23. Potato Late Blight, Phytophthora infestans	1000
24. Raspberry healthy	371
25. Soybean healthy	5090
26. Squash Powdery Mildew, Erysiphe cichoracearum	1835
27. Strawberry Healthy	456
28. Strawberry Leaf Scorch, Diplocarpon earlianum	1109
29. Tomato Bacterial Spot, Xanthomonas campestris pv. vesicatoria	2127
30. Tomato Early Blight, Alternaria solani	1000
31. Tomato Late Blight, Phytophthora infestans	1591
32. Tomato Leaf Mold , Passalora fulva	1909
33. Tomato Septoria Leaf Spot, Septoria lycopersici	952
34. Tomato Two Spotted Spider Mite, Tetranychus urticae	1771
35. Tomato Target Spot, Corynespora cassiicola	1676
36. Tomato Mosaic Virus	1404
37. Tomato Yellow Leaf Curl Virus	373
38. Tomato healthy	5375
Total	54323

Table 3.1: PlantVillage dataset details

Class name	Train Images	valid Images	Test Images
1. Bacterial spot	900	100	100
2. Early blight	900	100	100
3. Late blight	900	100	100
4. Leaf Mold	900	100	100
5. Septoria leaf spot	900	100	100
6. Spider mites	900	100	100
7. Target Spot	900	100	100
8. mosaic virus	900	100	100
9. Yellow Leaf Curl Virus	900	100	100
10. Healthy	900	100	100
Total	9000	1000	1000

Table 3.2: Tomato dataset details

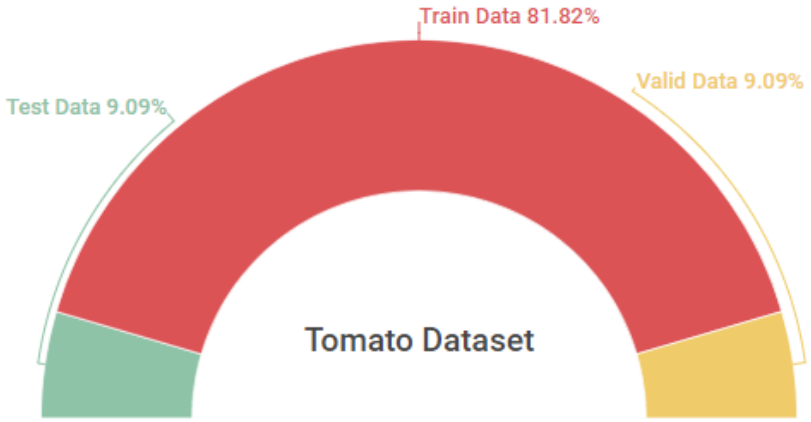

Figure 3.1: Tomato Dataset splitting

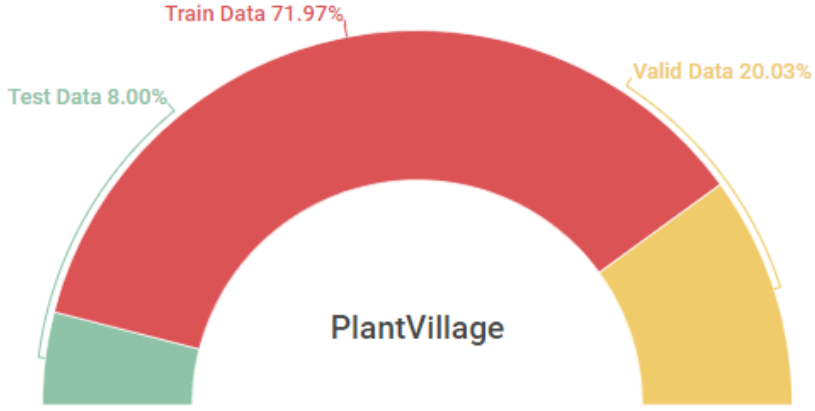

Figure 3.2: PlantVillage Dataset splitting

3.2 Preprocessing

The resulting dataset contains plant leaf images with dif- ferent shapes and resolutions. In order to normalize the input dataset and ensure consistency across all dataset, we reshaped all images with a fixed size of 256×256 pixels.

3.3 Data Augmentation

After collecting the dataset, we used augmentation tech- niques to increase the size of the dataset and include more variations. For this purpose, the images on our datasets were rotated, zoomed, and horizontally flipped, making the dataset larger and diverse.

3.4 Implementation frameworks and tools

3.4.1 Python

Python is a programming language used in machine learning and data science and other sectors of activity created by Guido van Rossum, is an open source, cross-platform, object-oriented programming language. Thanks to specialized libraries, Python is used for many situations such as software development, data analysis, or infrastructure management. It is an interpreted programming language, Python allows the execution of code on any computer. Usable by both beginner and expert programmers, Python allows you to create programs in a quick and easy way.[58]

3.4.2 Tensorflow

An open source machine learning library for research and production developed by Google brain team , with Python and C++ programming languages as backend,Tensorflow use various optimization techniques to make the calculation of mathematical easier and faster and more efficient and automatically computing their derivation,Its flexible architecture allows development on several varieties of platforms (CPU, GPU, TPU). [59]

3.4.3 Keras

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation.[60]
keras is :

- **Simple** : Keras reduces developer cognitive load to free you to focus on the parts of the problem that really matter.

- **Flexible** : Keras adopts the principle of progressive disclosure of complexity: simple workflows should be quick and easy, while arbitrarily advanced workflows should be possible via a clear path that builds upon what you've already learned.
- **Powerful** :Keras provides industry-strength performance and scalability: it is used by organizations and companies including NASA, YouTube, or Waymo. [60]

3.4.4 PyTorch

PyTorch is an open source machine learning framework based on the Torch library, used for applications such as computer vision and natural language. The open-source software was developed by the artificial intelligence teams at Facebook Inc. in 2016. PyTorch offers two significant features including tensor computation, as well as functional deep neural networks.[61]

3.4.5 Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It can be combined with scientific python libraries like numpy also It can export matrix formats (PNG, JPEG) and vector formats (PDF, SVG). [62] this library can:

- Create publication quality plots.
- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout. [62]

3.4.6 Google Colab

Colab is a free Jupyter notebook environment that runs entirely in the cloud with free GPU and TPU with a limit of 12 hours per session and 12 GB of RAM limits. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members.[63]
Google Collab allows us to:

1. Write and execute code in Python
2. Create/Upload/Share notebooks
3. Import/Save notebooks from/to Google Drive
4. Import/Publish notebooks from GitHub
5. Import external datasets e.g. from Kaggle
6. Integrate PyTorch, TensorFlow, Keras, OpenCV

3.4.6.1 Google Colab Pro

Google Colab Pro offers a multitude of improvements, such as enabling a faster online GPU, a P100 or T4 GPU. The standard Colab comes with a slower, K80 GPU. The Pro subscription also allows users to work with more memory provided by a larger RAM, and train their models for a longer period of time,is approximately 40% faster than the standard Google Colab environment.[64]

3.4.7 kaggle

Kaggle is an online community platform for data scientists and machine learning enthusiasts. Kaggle allows users to collaborate with other users, find and publish datasets, use GPU integrated notebooks, and compete with other data scientists to solve data science challenges. Kaggle provides powerful resources on cloud and allows you to use a maximum of 30 hours of GPU and 20 hours of TPU per week.

3.4.8 Hugging Face

Hugging Face is a community and data science platform that provides:

- Tools that enable users to build, train and deploy ML models based on open source (OS) code and technologies.
- A place where a broad community of data scientists, researchers, and ML engineers can come together and share ideas, get support and contribute to open source projects.

Transformers of huggingFace provides APIs to easily download and train state-of-the-art pretrained models. Using pretrained models can reduce your compute costs and save you time from training a model from scratch. The models can be used across different modalities such as: Text,Audio and our case Images: image classification, object detection, and segmentation.

This library supports seamless integration between three of the most popular deep learning libraries: PyTorch, TensorFlow and Train your model in three lines of code in one framework, and load it for inference with another. [65]

3.4.9 Roboflow

Roboflow is a Computer Vision developer framework for better data collection to preprocessing, and model training techniques. Roboflow has public datasets readily available to users and has access for users to upload their own custom data also. Roboflow accepts various annotation formats. Roboflow makes managing, preprocessing, augmenting, and versioning datasets for computer vision seamless. Developers reduce 50% of their code when using Roboflow's workflow, automate annotation quality assurance, save training time, and increase model reproducibility.[66]

3.4.9.1 PlantDoc

PlantDoc is a dataset of 2,569 images with size of 416 X 416 across 13 plant species and 30 classes (diseased and healthy) for image classification and object detection. There are 8,851 labels. we use it in detection while we work with algorithm YOLOv5 in our project. [66]

3.5 Model Evaluation Metrics

This section discusses how we can evaluate our model results, what makes a model better than another, and how we can assure that our model will give good results in the testing phase.

3.5.1 Accuracy

In order to know how the model is accurate, we need to mention accuracy which is an important evaluation metric and the most intuitive performance measure, it is simply a ratio of correctly predicted observations to the total observations.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

3.5.2 Precision

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

$$Precision = \frac{TP}{TP + FP}$$

3.5.3 Recall

Named also Sensitivity, Recall is the ratio of correctly predicted positive observations to the all observations in actual class, in other way, recall measures the proportion of actual positives that are correctly identified.

$$Recall = \frac{TP}{TP + FN}$$

3.5.4 F1-score

F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0.

Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if we have an uneven class distribution. Accuracy works best if false positives and false negatives

have similar costs.

If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.F1 Score single metric that combines recall and precision using the harmonic mean [67].

$$F1-score = 2X\frac{RecallXPrecision}{Recall+Precision}$$

- **True Positive (TP) :** Number of normal data classified correctly as normal.
- **False Positive (FP) :** Number of abnormal data incorrectly classified as normal.
- **True Negative (TN) :** Number of abnormal data classified correctly as abnormal .
- **False Negative (FN):** Number of normal data incorrectly classified as abnormal.

3.5.5 Confusion matrix

A confusion matrix is a technique for summarizing the performance of a classification algorithm. Classification accuracy alone can be misleading if you have an unequal number of observations in each class or if you have more than two classes in your dataset. Calculating a confusion matrix can give you a better idea of what your classification model is getting right and what types of errors it is making. The confusion matrix shows the ways in which your classification model is confused when it makes predictions.[68]

3.6 Summary

This chapter provided an overview of the dataset and how we processed it to make it ready to fit the algorithms, as well as the tools that will be used to provide the results and the evaluation metrics for further validating and comparing our models.

CHAPTER 4

EXPIREMENTS AND RESULTS

This chapter summarizes our work by detailing and explaining various experiments on various proposed architectures and approaches. Tables and graphs show comparisons between the experiments. We ran twelve 15 experiments on two datasets, and we analyzed each result by changing the deep architecture, hyperparameters, tuned pretrained models, modified versions of these models and based on vision transformers models, We compare them and obtained different results.

4.1 Deep learning approaches

In this section, we will show how we used deep learning to classify and diagnose plant leaf diseases. We focused on the use of convolution neural networks because they perform well when dealing with images.

During the experiments, we changed the hyperparameters and evaluated the results,We also demonstrate the utility of some pretrained models by fine-tuning them to fit our dataset.

Finally, comparisons for the various experiments are provided.

4.2 Convolution neural networks

4.2.1 Our Proposed CNN From scratch

The first CNN experiment architecture has 256 X 256 X 3 as input shape with both tomato and PlantVillage dataset , is composed of two Convolution layers with 32 filters of size 3x3 followed by a max pooling layer with a window of size 3x3 , a second block contain two convolutional layers and max pooling layer with the same configuration of block 1 but with 64 filters in convolution, a

Third block contain two convolutional layers and max pooling layer with the same configuration of block 1 but with 128 filters in convolution, the fourth block has two Convolution layers with 256 filters of size 3x3 without max Pooling, then the last block contain two Convolution layers with 512 filters of size 5X5 without max pooling
A fully connected layer of 1024 units followed by a dropout layer with 50% and an output layer with 10 units (in case of tomato Dataset) and 38 units (in case of plantVillage dataset) according to the number of classes . We will be using the Rectified linear unit (ReLU) activation function for all the layers except the final output layer with a softmax function,

using Early Stopping 2.2.5.3 The training was done during 14 epochs in **28 minutes** with tomato Dataset, and for PlantVillage Dataset it took 17 epochs in **3 hours and 21 minutes** using google colab pro 3.4.6.1, the models are optimized by Adam Optimizer with learning rate equal to 0.0001.

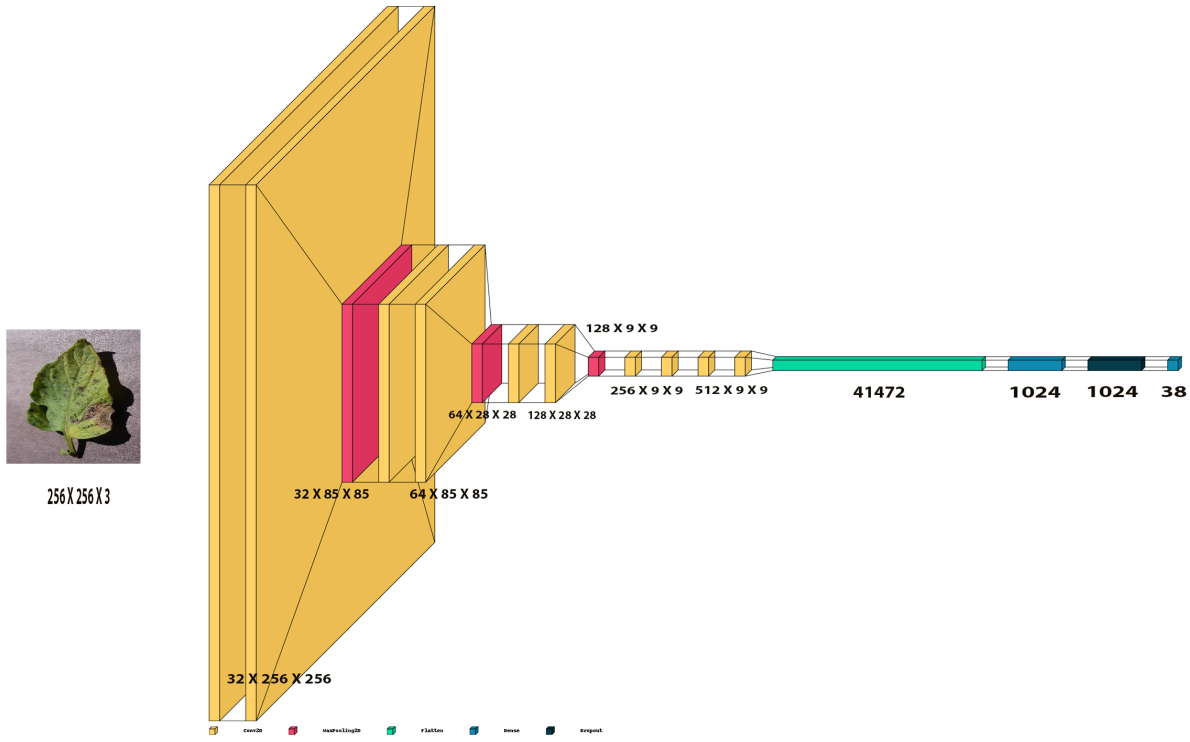

Figure 4.1: Cnn Architucture

The hyperparameters of the architecture are described below.

Parameter	**Tomato Dataset**	**PlantVillage Dataset**
Number of epochs	14	17
batch size	30	30
Optimizer	Adam	Adam
Dropout rate	0.5	0.5
Learning rate	0.0001	0.0001

Table 4.1: hyperparameters of Our Proposed CNN From scratch

- We obtained from this experiment an accuracy of 87.9% on validation set and 90.4% on test set with tomato dataset.

- We obtained from this experiment an accuracy of 96.57% on validation set and 96.5% on test set with PlantVillage dataset.

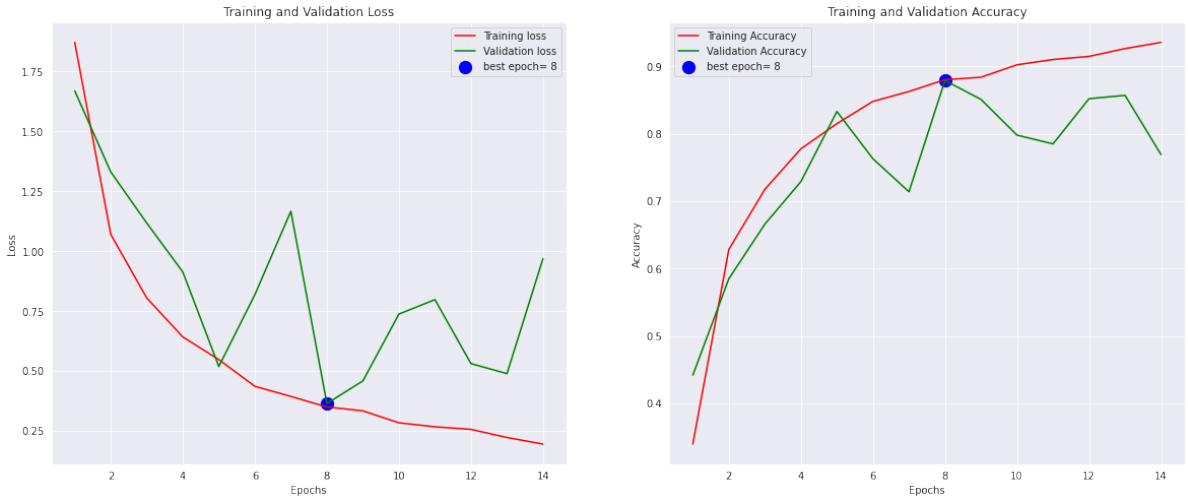

Figure 4.2: Training and validation loss and accuracy Cnn Tomato Dataset

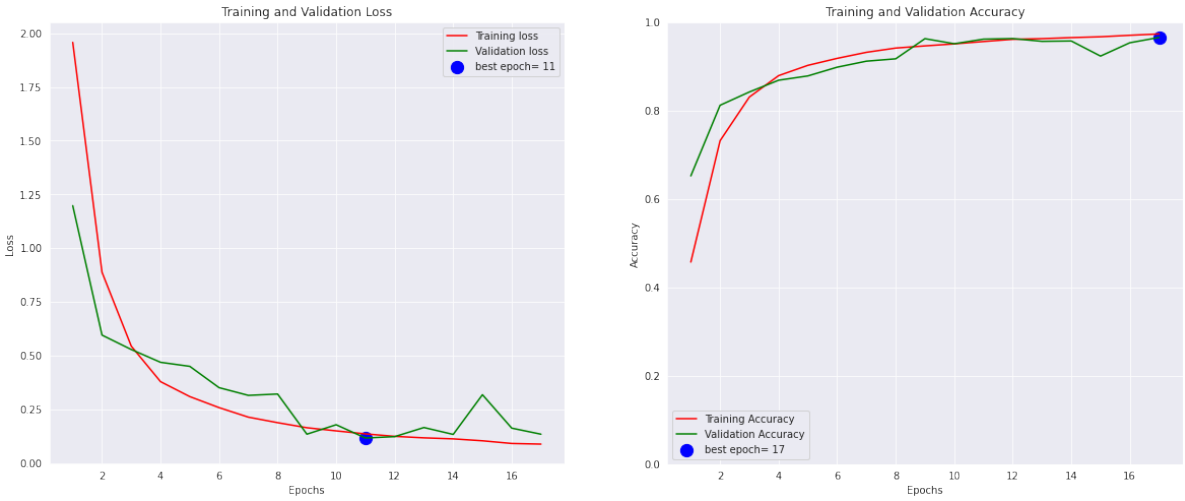

Figure 4.3: Training and validation loss and accuracy Cnn PlantVillage Dataset

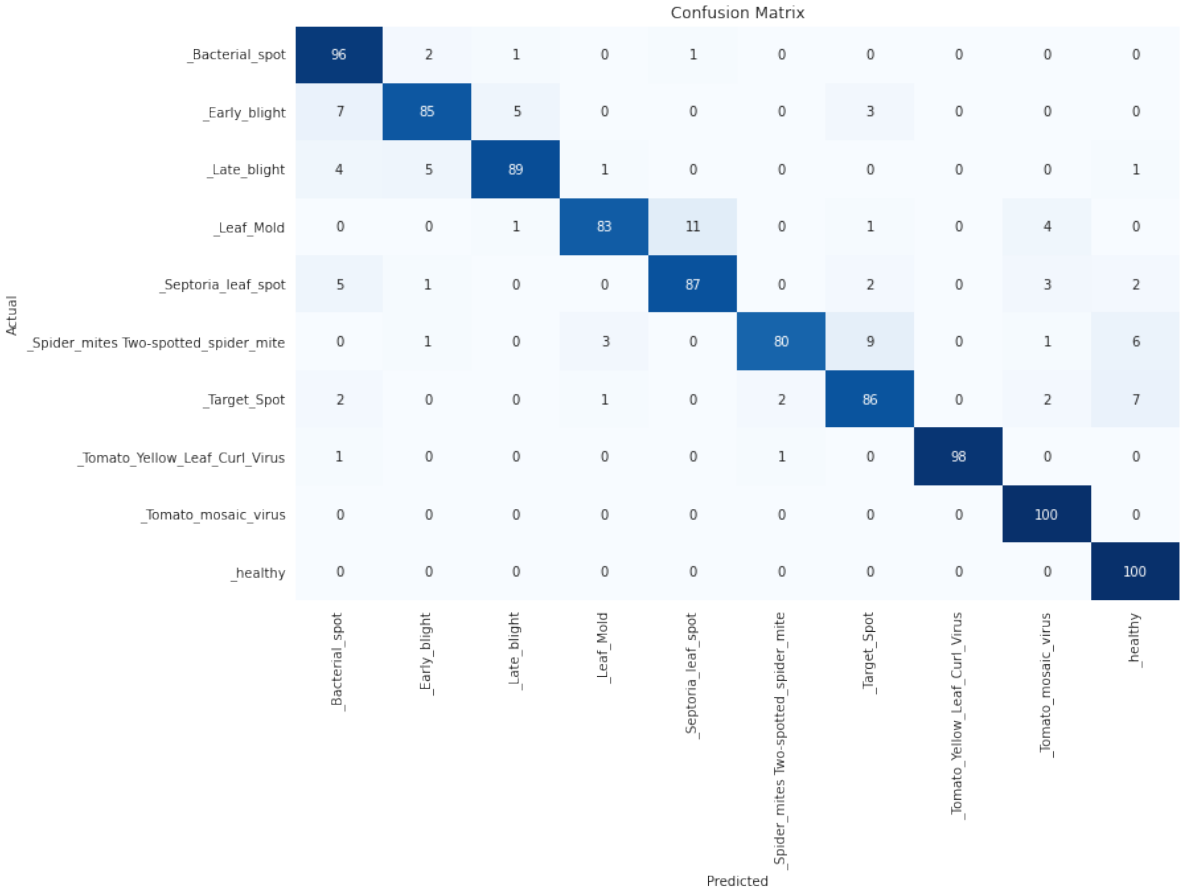

Figure 4.4: Confusion Matrix Cnn Tomato Dataset

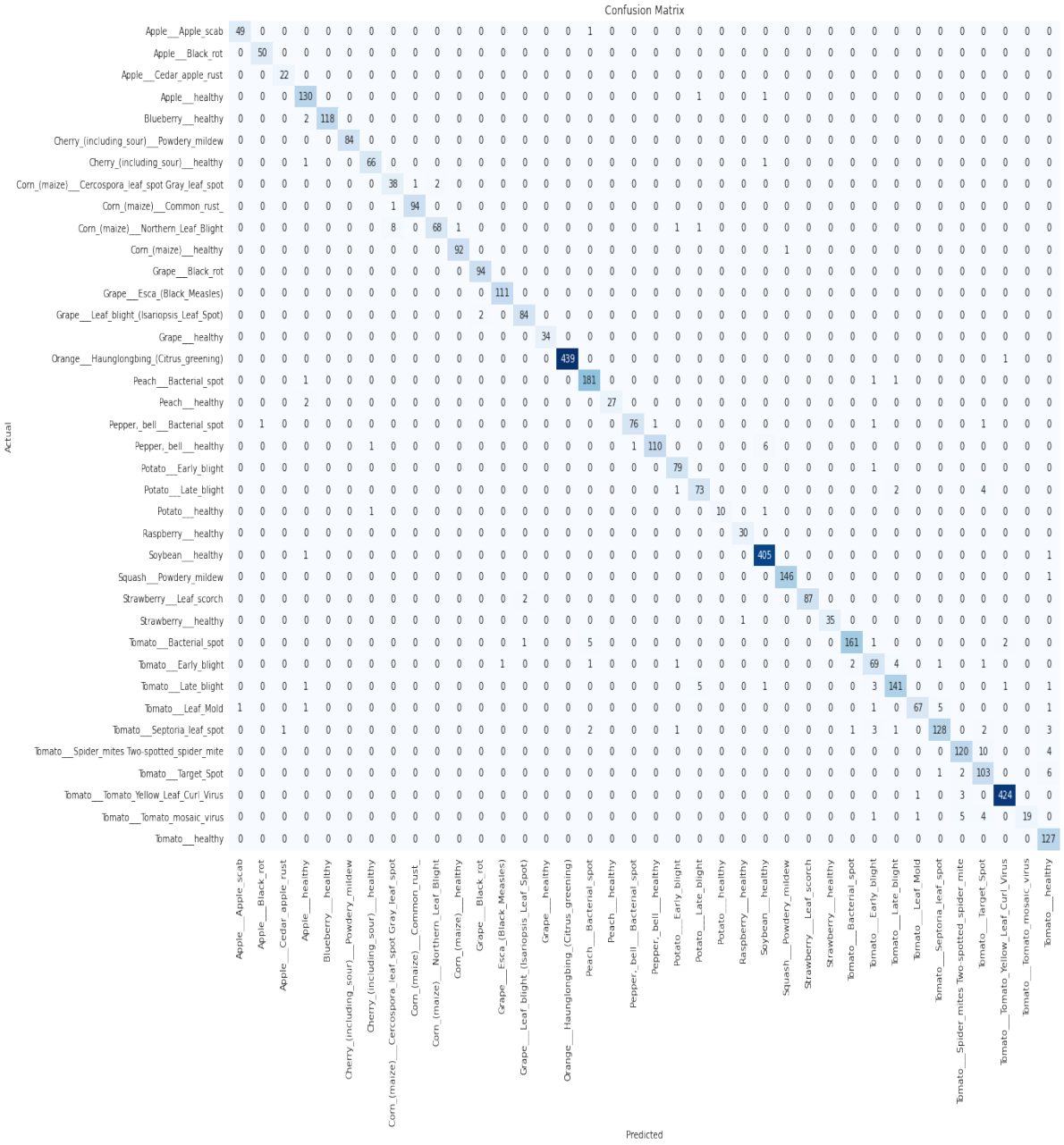

Figure 4.5: Confusion Matrix Cnn PlantVillage Dataset

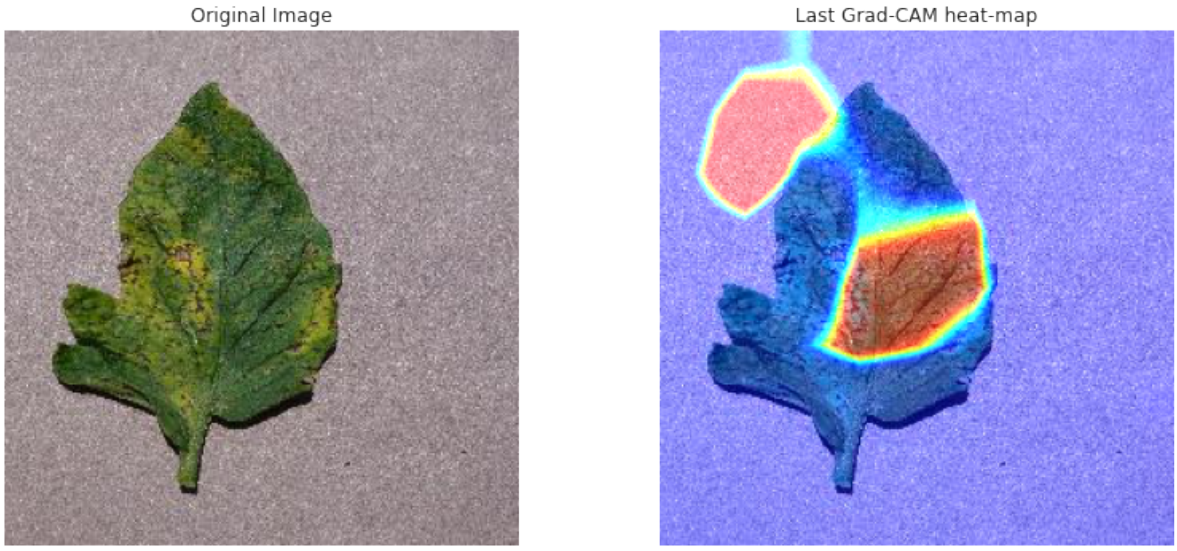

Figure 4.6: Grad Cam visualisation of cnn

4.2.2 Our Proposed CNN with Attention

this CNN architecture has 256 X 256 X 3 as input shape with both tomato and PlantVillage dataset , is composed of two Convolution layers with 32 filters of size 3x3 followed by a max pooling layer with a window of size 3x3 then we added a convolution block attention module: channel attention with 32 filters and spatial attention with 7 as kernel size, a second block contain two convolutional layers and max pooling layer with the same configuration of block 1 but with 64 filters in convolution then we added a convolution block attention module: channel attention with 64 filters and spatial attention with 7 as kernel size, a Third block contain two convolutional layers and max pooling layer with the same configuration of block 1 but with 128 filters in convolution then we added a convolution block attention module: channel attention with 128 filters and spatial attention with 7 as kernel size the fourth block has two Convolution layers with 256 filters of size 3x3 without max Pooling then we added a convolution block attention module: channel attention with 256 filters and spatial attention with 7 as kernel size then the last block contain two Convolution layers with 512 filters of size 5X5 without max pooling then we added a convolution block attention module: channel attention with 512 filters and spatial attention with 7 as kernel size

A fully connected layer of 1024 units followed by a dropout layer with 50% and an output layer with 10 units (in case of tomato Dataset) and 38 units (in case of plantVillage dataset) according to the number of classes . We will be using the Rectified linear unit (ReLU) activation function for all the layers except the final output layer with a softmax function,

using Early Stopping 2.2.5.3 The training was done during 14 epochs in **4 hours and 17 minutes** with tomato Dataset, and for PlantVillage Dataset it took 25 epochs in **8 hours and**

20 minutes using google colab pro 3.4.6.1, the models are optimized by Adam Optimizer with learning rate equal to 0.0001. The hyperparameters of the architecture are described below.

Parameter	**Tomato Dataset**	**PlantVillage Dataset**
Number of epochs	14	25
batch size	30	30
Optimizer	Adam	Adam
Dropout rate	0.5	0.5
Learning rate	0.0001	0.0001

Table 4.2: hyperparameters of Our Proposed CNN From scratch

- We obtained from this experiment an accuracy of 90.7% on validation set and 92.7% on test set with tomato dataset.

- We obtained from this experiment an accuracy of 97.74% on validation set and 97.49% on test set with PlantVillage dataset.

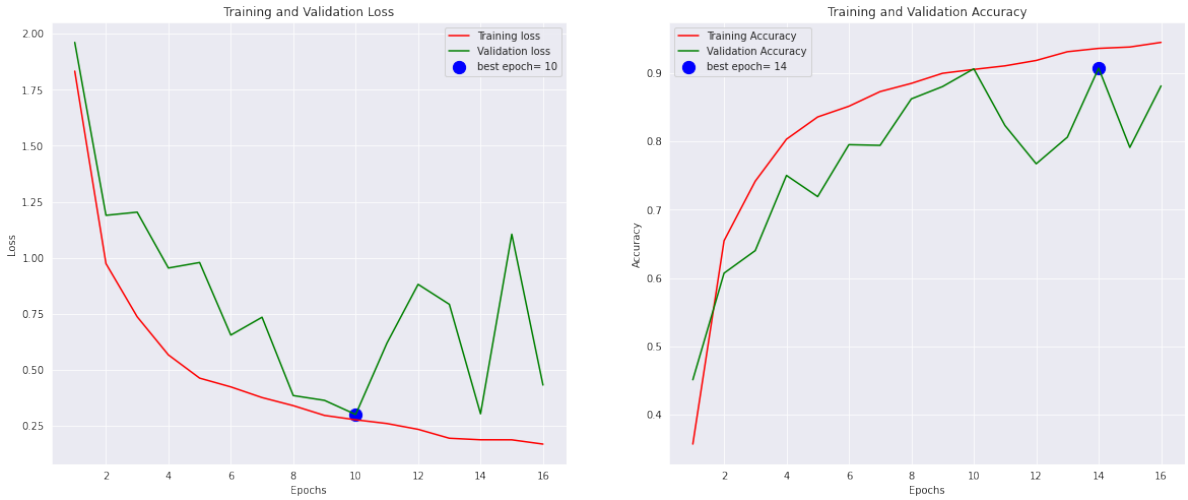

Figure 4.7: Training and validation loss and accuracy Cnn with Attention Tomato Dataset

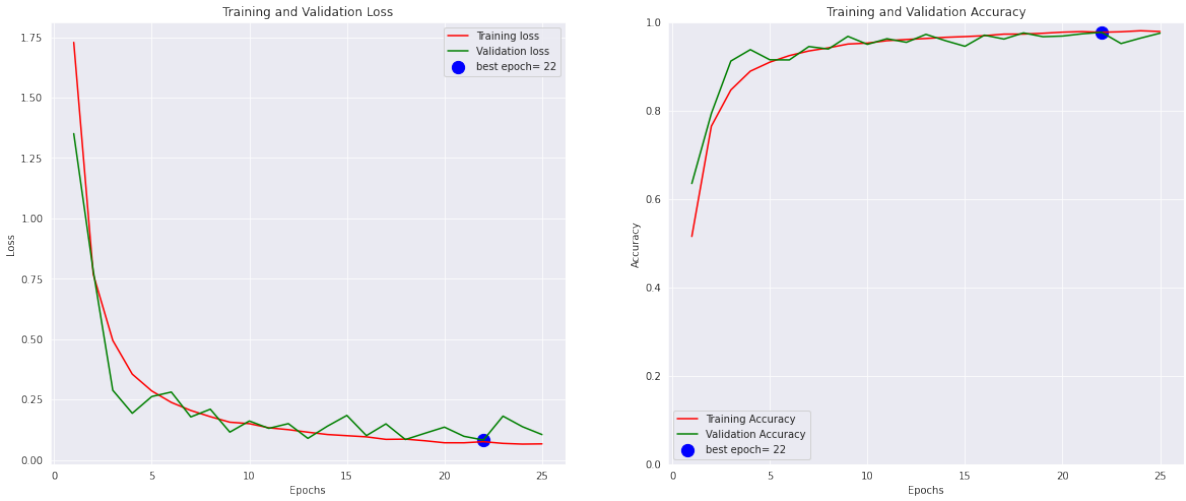

Figure 4.8: Training and validation loss and accuracy Cnn with Attention PlantVillage Dataset

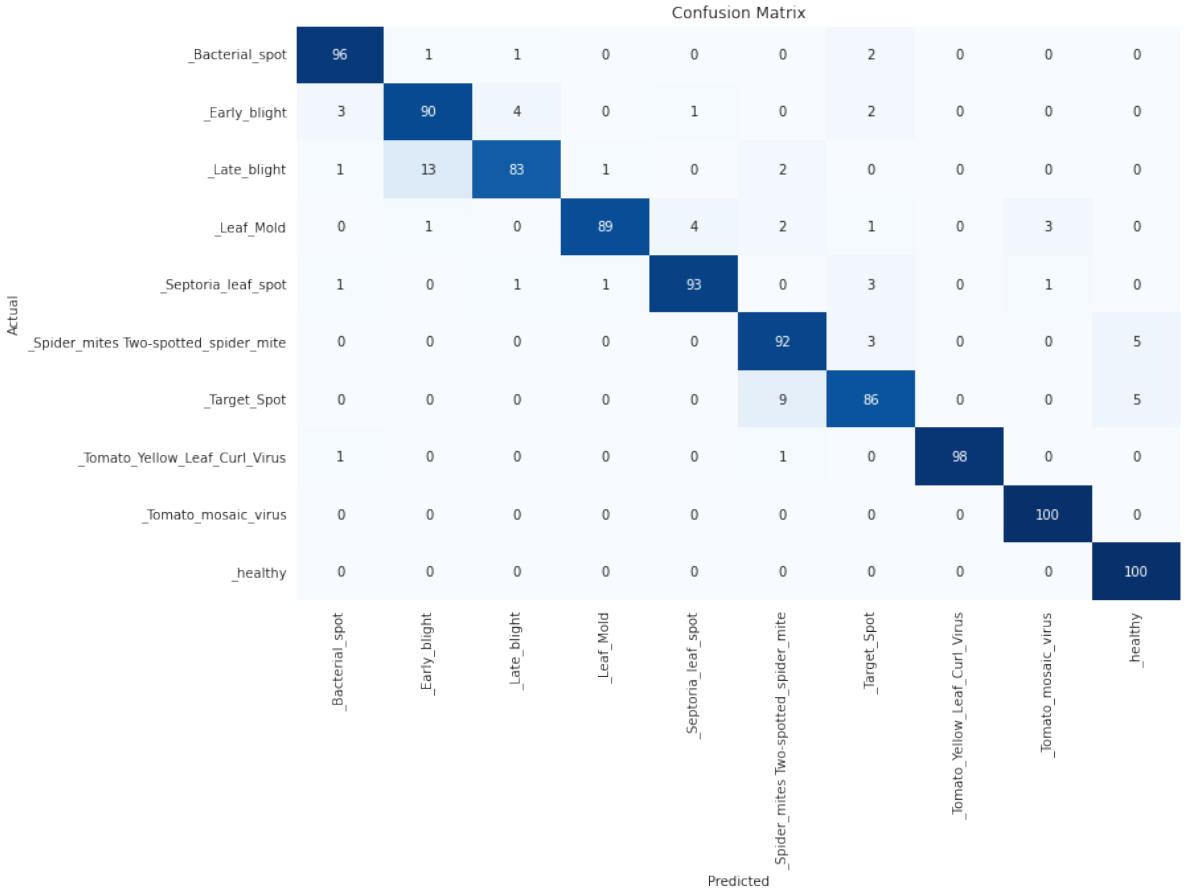

Figure 4.9: Confusion Matrix Cnn with Attention Tomato Dataset

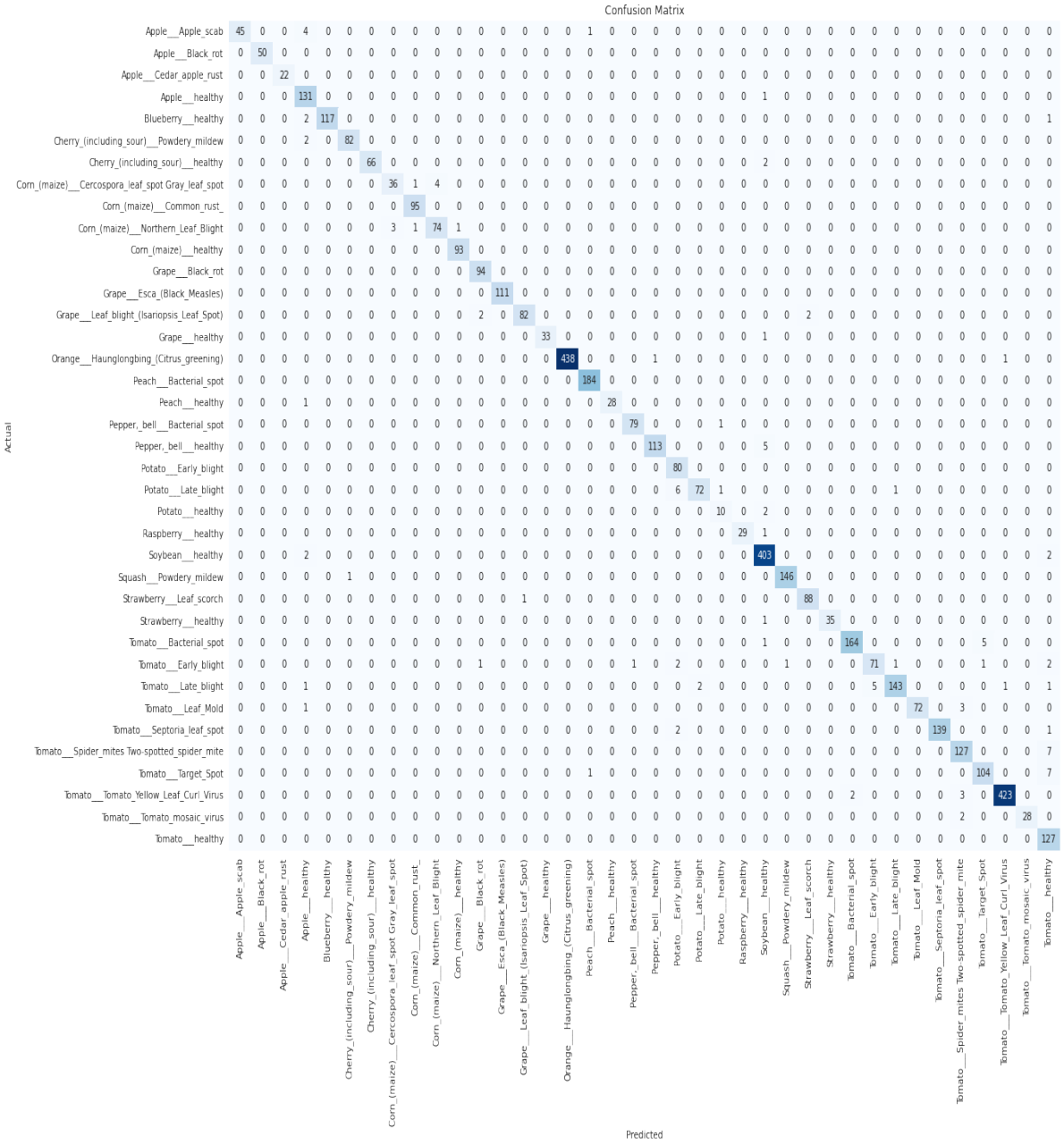

Figure 4.10: Confusion Matrix Cnn with Attention PlantVillage Dataset

Figure 4.11: Grad Cam visualisation of cnn with Attention

4.3 Pretrained models (Transfer learning)

Transfer learning is a popular method in computer vision because it allows us to build accurate models in a time saving way [69]. With transfer learning, instead of starting the learning process from scratch, you start from patterns that have been learned when solving a different problem. This way you leverage previous learnings and avoid starting from scratch 2.5
in this section we will try some pretrained models to classify our images and see if these models will give better results compared to our proposed architectures.we will divide this phase into x main approaches (VGG 16, Resnet , Mobile NET)

4.3.1 Modified VGG-16 pretrained model

We tuned VGG16 model by removing the top layer (instruction in keras : include-top=False) and unfrozen the model(instruction in keras trainable = true), the input data of this model (tomato dataset) has a shape (128,128,3) and (256,256,3) with plantVillage dataset it was normalized and augmented by a horizontal flip, rotation, zoom and rescaling values.
we added a batch normalization layer followed by two fully connected layers the first with 256 units and the second with 512 units then we added a dropout with 40% , and changed the output layer to have ten units (in case of tomato Dataset) and 38 units (in case of plantVillage dataset) according to the number of classes . We will be using the Rectified linear unit (ReLU) activation function for all the layers except the final output layer with a softmax function, using Early Stopping 2.2.5.3 The training was done during 29 epochs in **26 minutes** with tomato Dataset, and for PlantVillage Dataset it took 20 epochs in **4 hours 20 minutes** using google colab pro

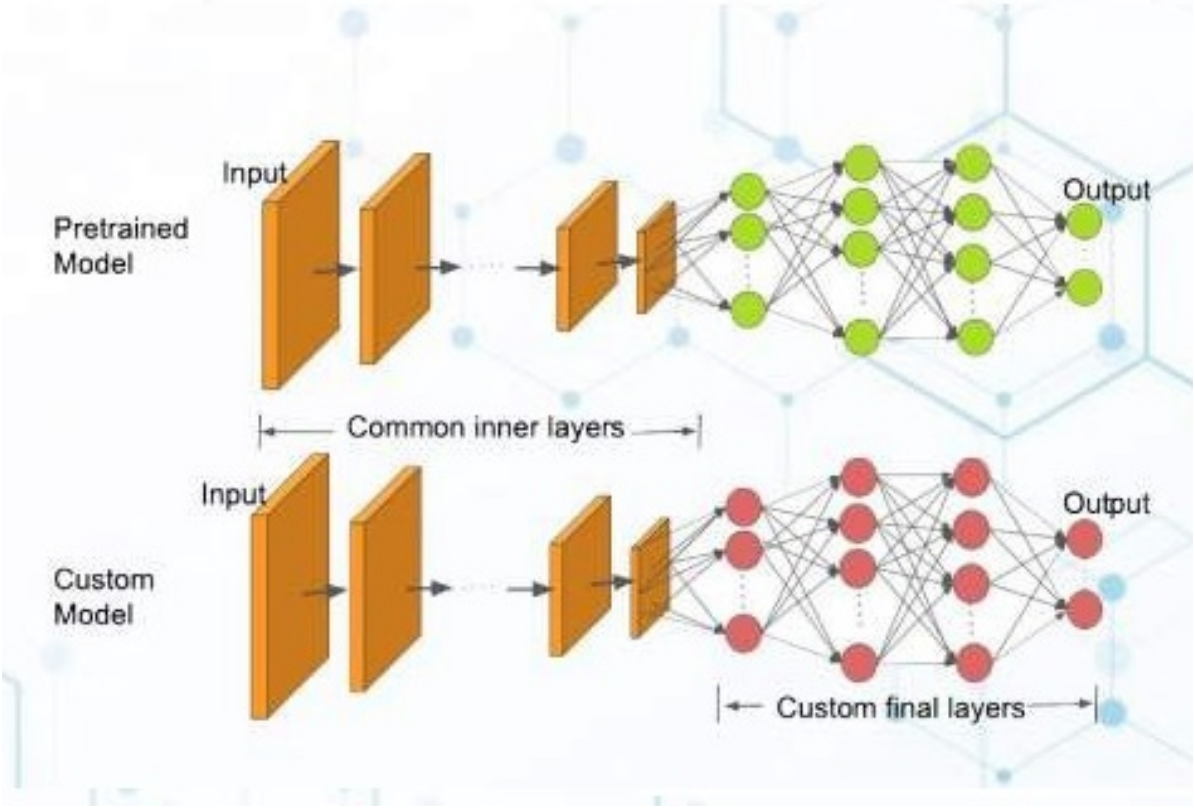

Figure 4.12: Transfer learning process

3.4.6.1, the models are optimized by Adam Optimizer with learning rate equal to 0.001, The hyperparameters of the architecture are described below.

Parameter	**Tomato Dataset**	**PlantVillage Dataset**
Number of epochs	29	20
batch size	30	30
Optimizer	Adam	Adam
Dropout rate	0.4	0.4
Learning rate	0.001	0.001

Table 4.3: hyperparameters of Vgg16

- We obtained from this experiment an accuracy of 97.5% on validation set and 97.8% on test set with tomato dataset.

- We obtained from this experiment an accuracy of 99.13% on validation set and 98.37% on test set with PlantVillage dataset.

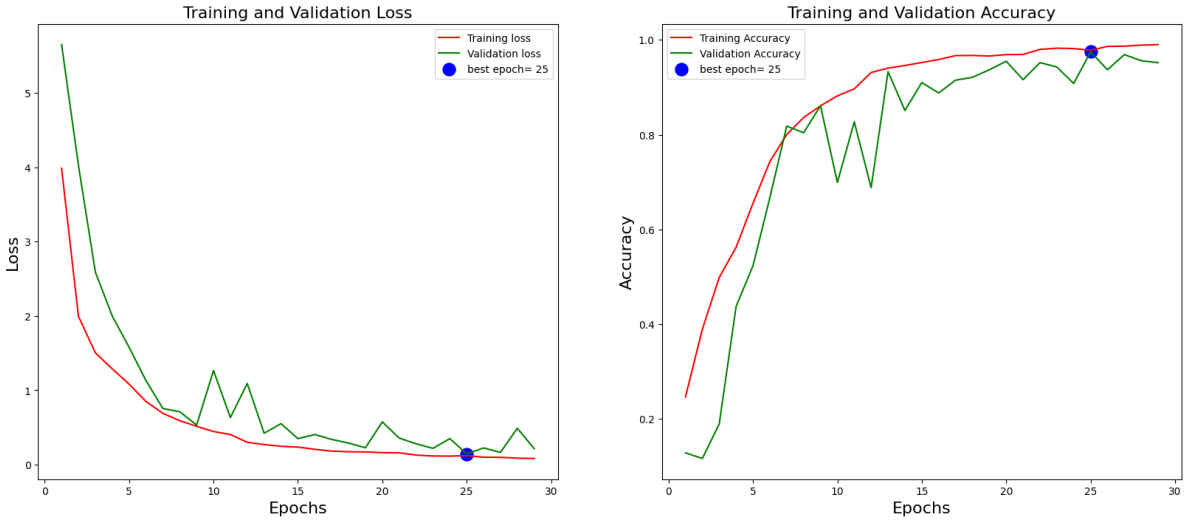

Figure 4.13: Training and validation loss and accuracy Vgg16 Tomato Dataset

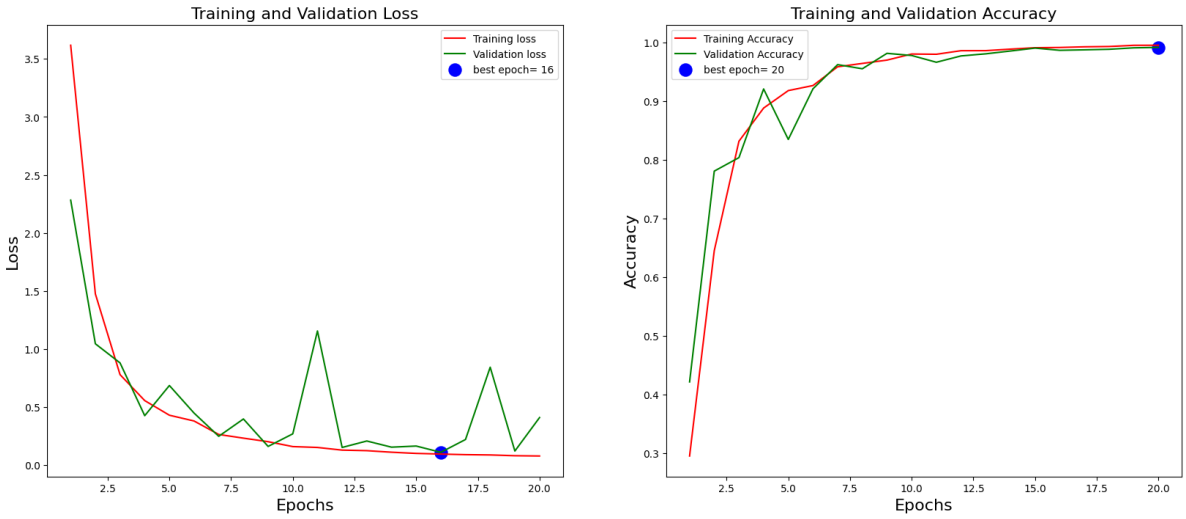

Figure 4.14: Training and validation loss and accuracy Vgg16 PlantVillage Dataset

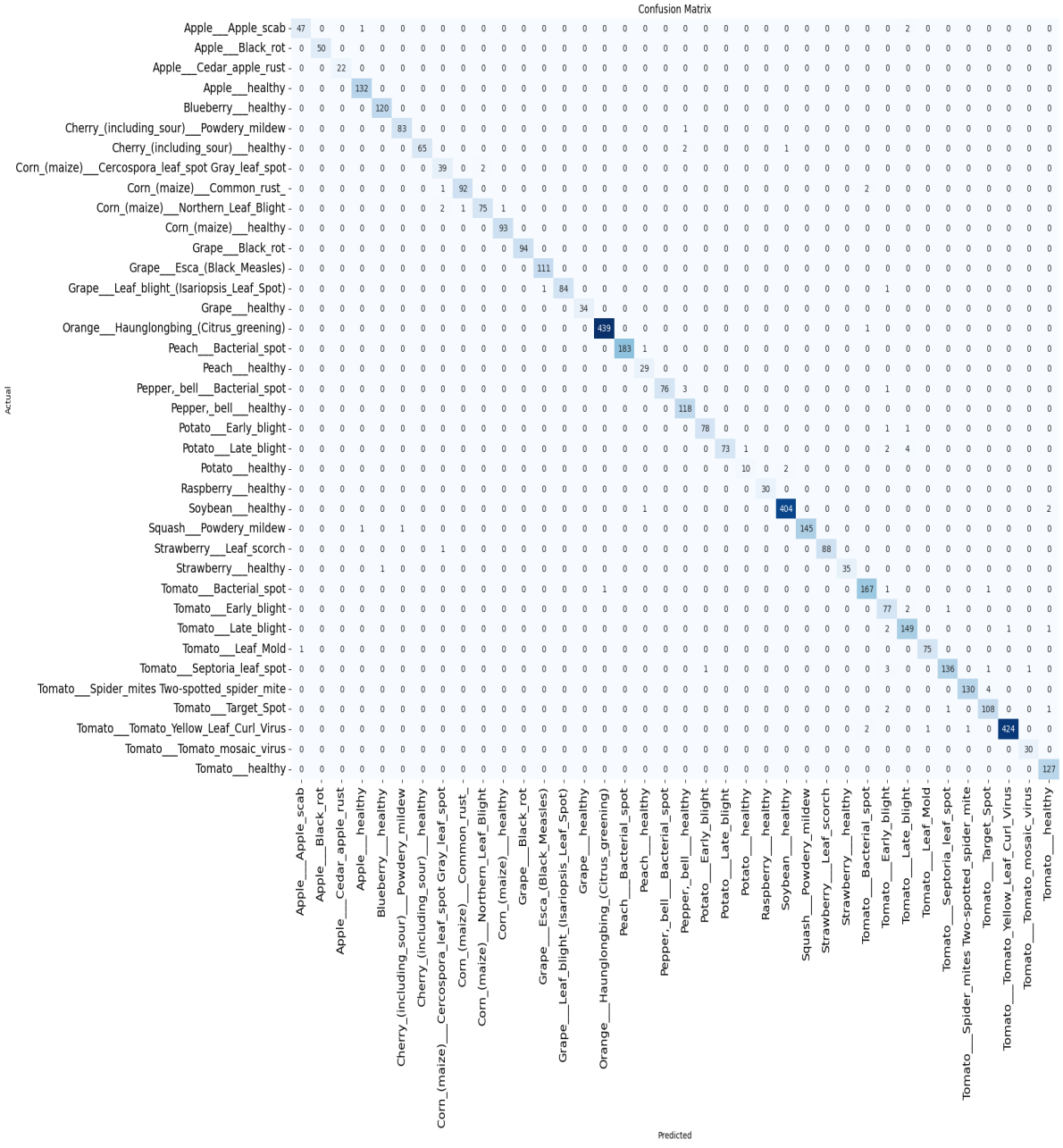

Figure 4.15: Confusion Matrix Vgg16 PlantVillage Dataset

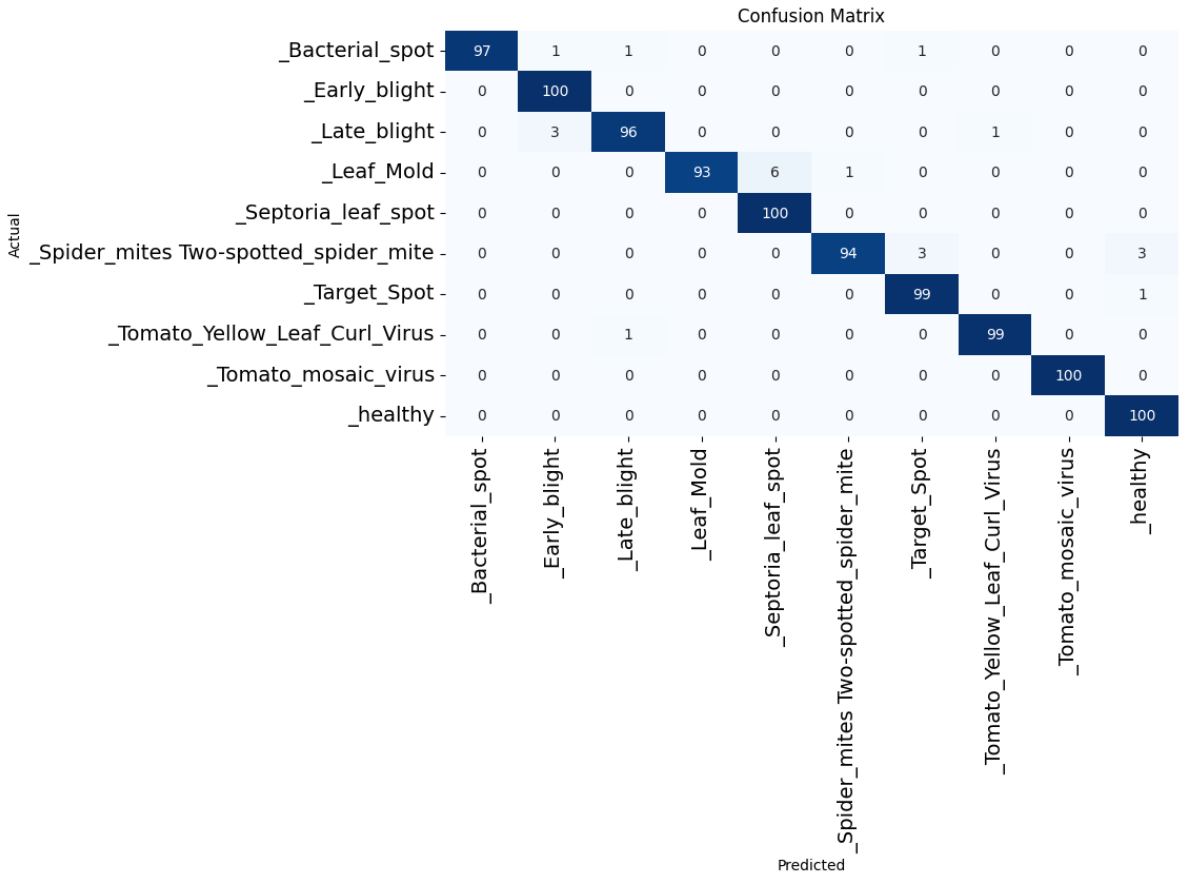

Figure 4.16: Confusion Matrix Vgg16 Tomato Dataset

4.3.2 Modified VGG-19 pretrained model

We tuned VGG19 model by removing the top layer (instruction in keras : include-top=False) and unfrozen the model(instruction in keras trainable = true), the input data of this model (tomato dataset) has a shape (128,128,3) and (256,256,3) with plantVillage dataset it was normalized and augmented by a horizontal flip, rotation, zoom and rescaling values.
we added a batch normalization layer followed by two fully connected layers the first with 256 units and the second with 512 units then we added a dropout with 40% , and changed the output layer to have ten units (in case of tomato Dataset) and 38 units (in case of plantVillage dataset) according to the number of classes . We will be using the Rectified linear unit (ReLU) activation function for all the layers except the final output layer with a softmax function, using Early Stopping 2.2.5.3 The training was done during 35 epochs in **34 minutes** with tomato Dataset, and for PlantVillage Dataset it took 14 epochs in **2 hours 48 minutes** using google colab pro 3.4.6.1, the models are optimized by Adam Optimizer with learning rate equal to 0.001, The hyperparameters of the architecture are described below.

Parameter	**Tomato Dataset**	**PlantVillage Dataset**
Number of epochs	35	14
batch size	30	30
Optimizer	Adam	Adam
Dropout rate	0.4	0.4
Learning rate	0.001	0.001

Table 4.4: hyperparameters of Vgg19

- We obtained from this experiment an accuracy of 94.7% on validation set and 97.2% on test set with tomato dataset.

- We obtained from this experiment an accuracy of 97.64% on validation set and 95.9% on test set with PlantVillage dataset.

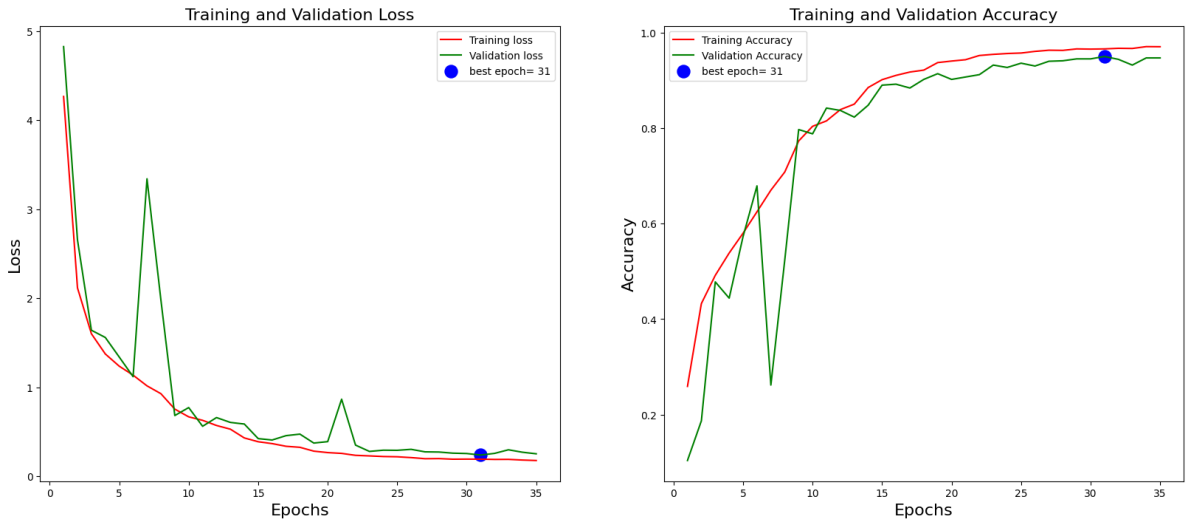

Figure 4.17: Training and validation loss and accuracy Vgg19 Tomato Dataset

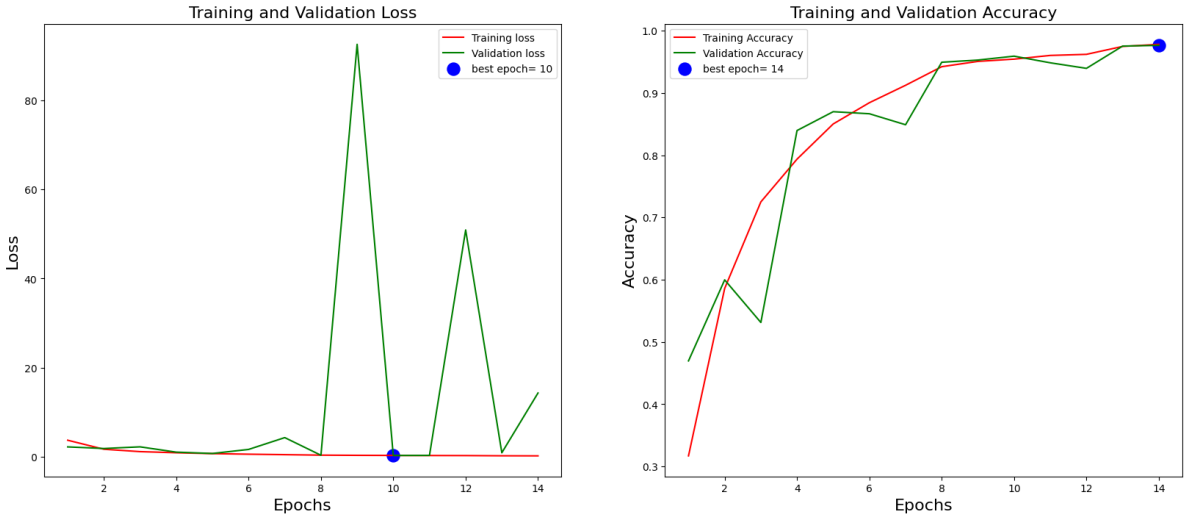

Figure 4.18: Training and validation loss and accuracy Vgg19 PlantVillage Dataset

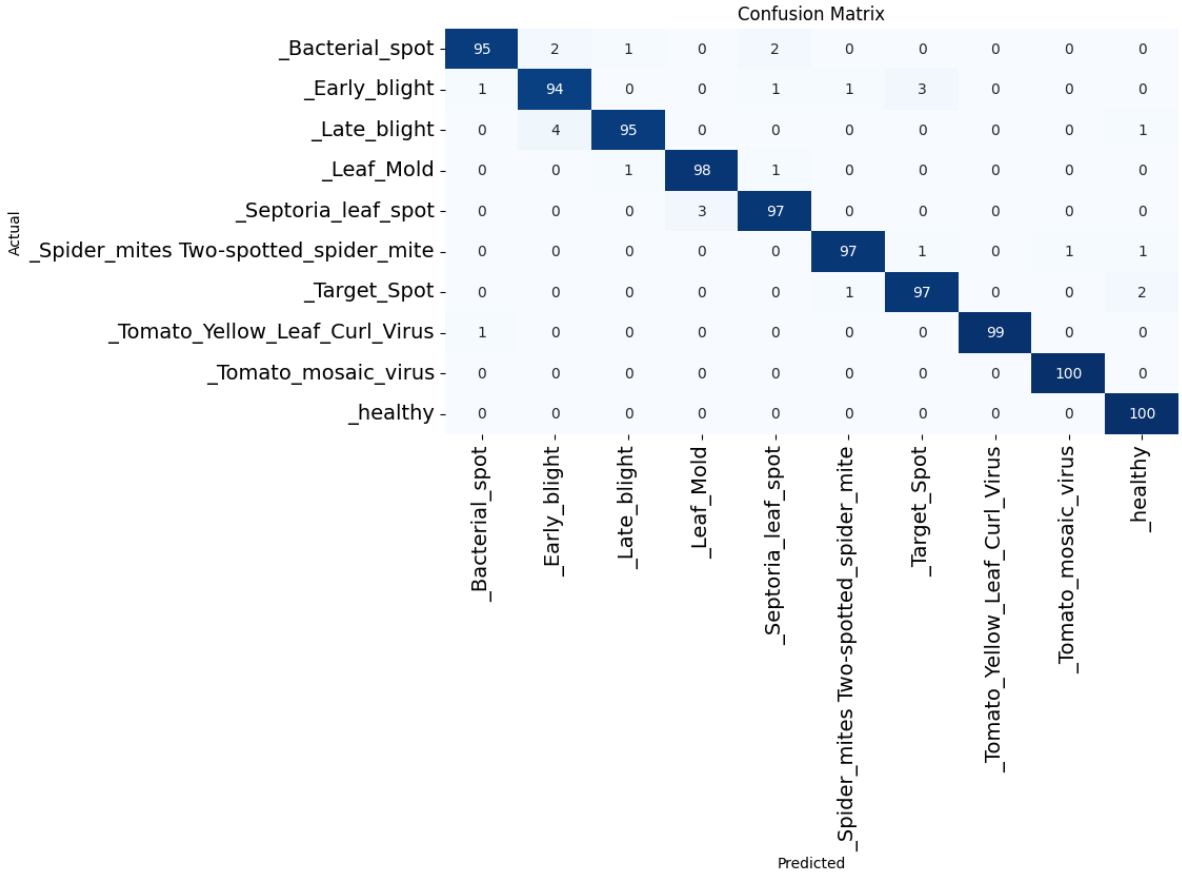

Figure 4.19: Confusion Matrix Vgg19 Tomato Dataset

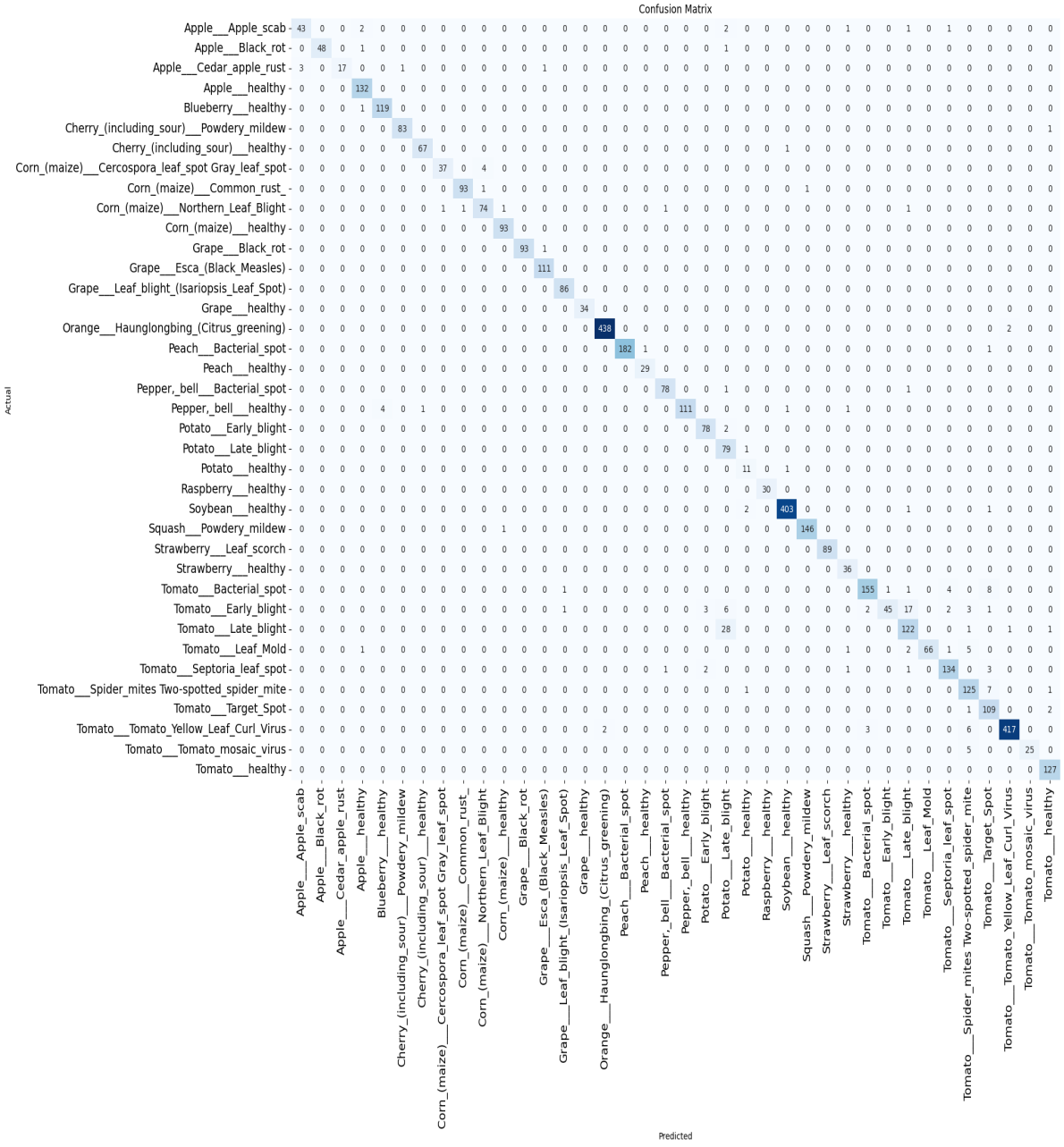

Figure 4.20: Confusion Matrix Vgg19 PlantVillage Dataset

4.3.3 Modified Inception pretrained model

We tuned Inception model by removing the top layer (instruction in keras : include-top=False) and unfrozen the model(instruction in keras trainable = true), the input data of this model (tomato dataset) has a shape (128,128,3) and (256,256,3) with plantVillage dataset it was normalized and augmented by a horizontal flip, rotation, zoom and rescaling values.
we added a batch normalization layer followed by two fully connected layers the first with 256

units and the second with 512 units then we added a dropout with 40% , and changed the output layer to have ten units (in case of tomato Dataset) and 38 units (in case of plantVillage dataset) according to the number of classes . We will be using the Rectified linear unit (ReLU) activation function for all the layers except the final output layer with a softmax function, using Early Stopping 2.2.5.3 The training was done during 16 epochs in **14 minutes** with tomato Dataset, and for PlantVillage Dataset it took 15 epochs in **2 hours 26 minutes** using google colab pro 3.4.6.1, the models are optimized by Adam Optimizer with learning rate equal to 0.001, The hyperparameters of the architecture are described below.

Parameter	**Tomato Dataset**	**PlantVillage Dataset**
Number of epochs	16	15
batch size	30	30
Optimizer	Adam	Adam
Dropout rate	0.4	0.4
Learning rate	0.001	0.001

Table 4.5: hyperparameters of Inception

- We obtained from this experiment an accuracy of 97.3% on validation set and 98.7% on test set with tomato dataset.

- We obtained from this experiment an accuracy of 99.67% on validation set and 99.31% on test set with PlantVillage dataset.

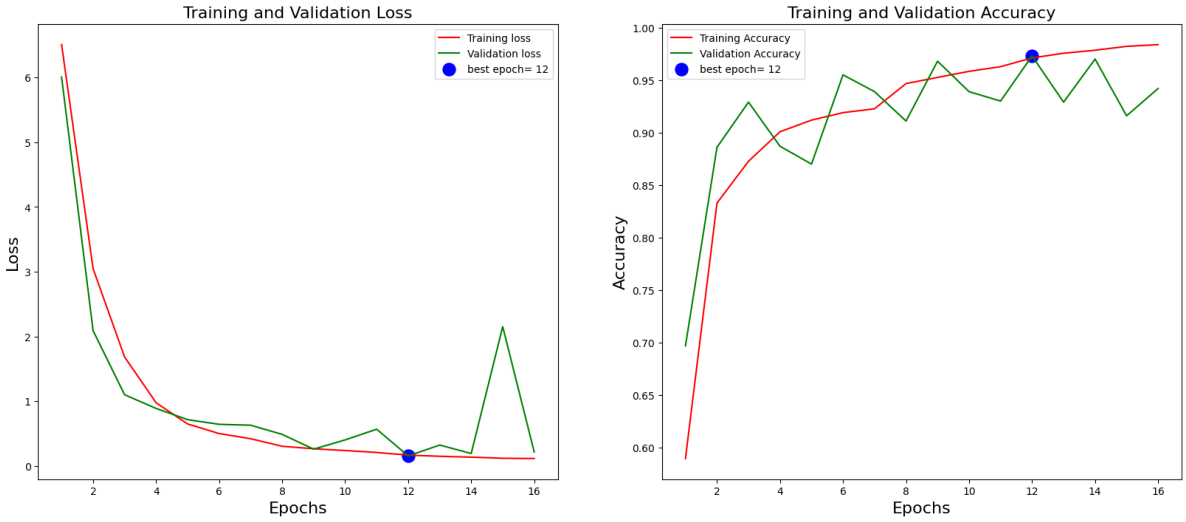

Figure 4.21: Training and validation loss and accuracy Inception Tomato Dataset

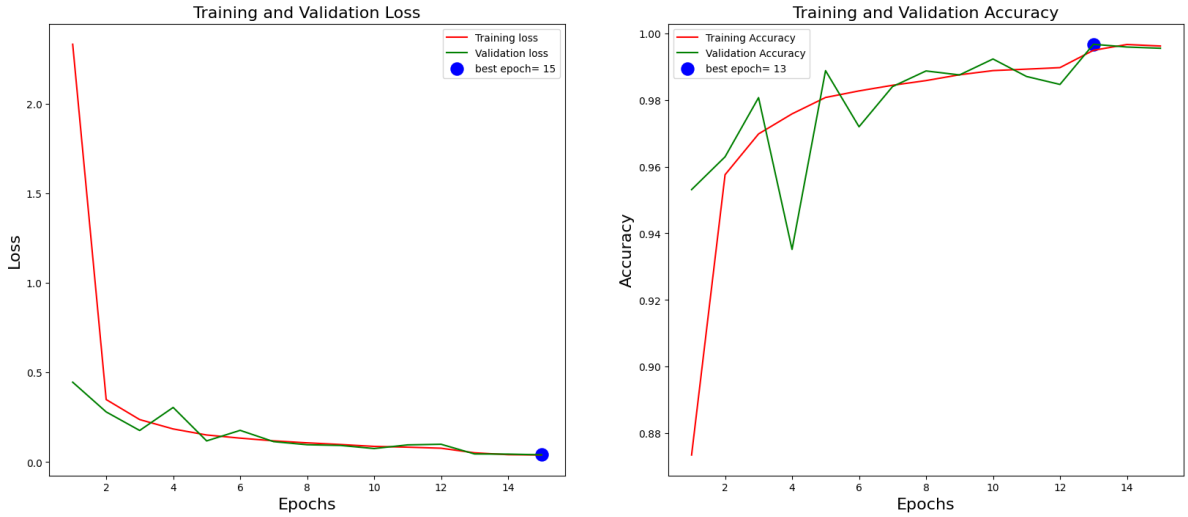

Figure 4.22: Training and validation loss and accuracy Inception PlantVillage Dataset

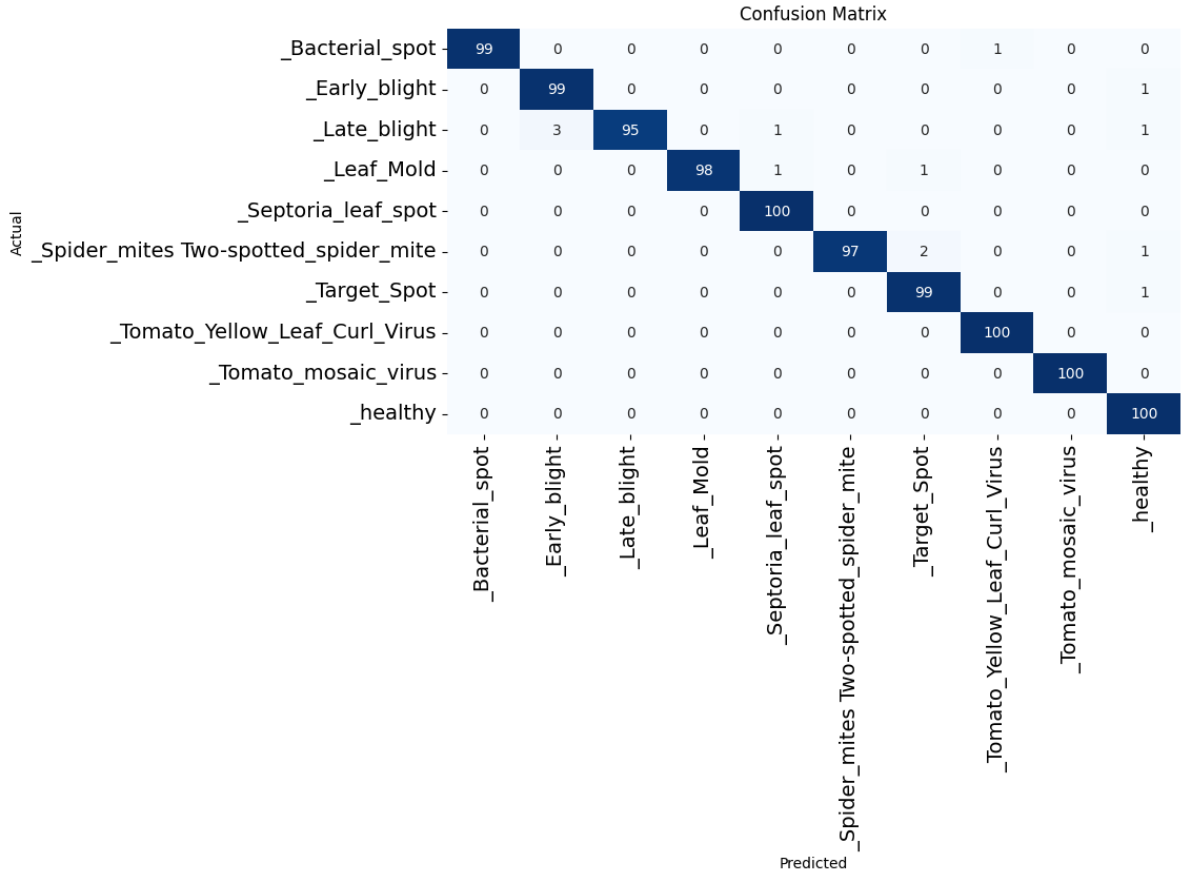

Figure 4.23: Confusion Matrix Inception Tomato Dataset

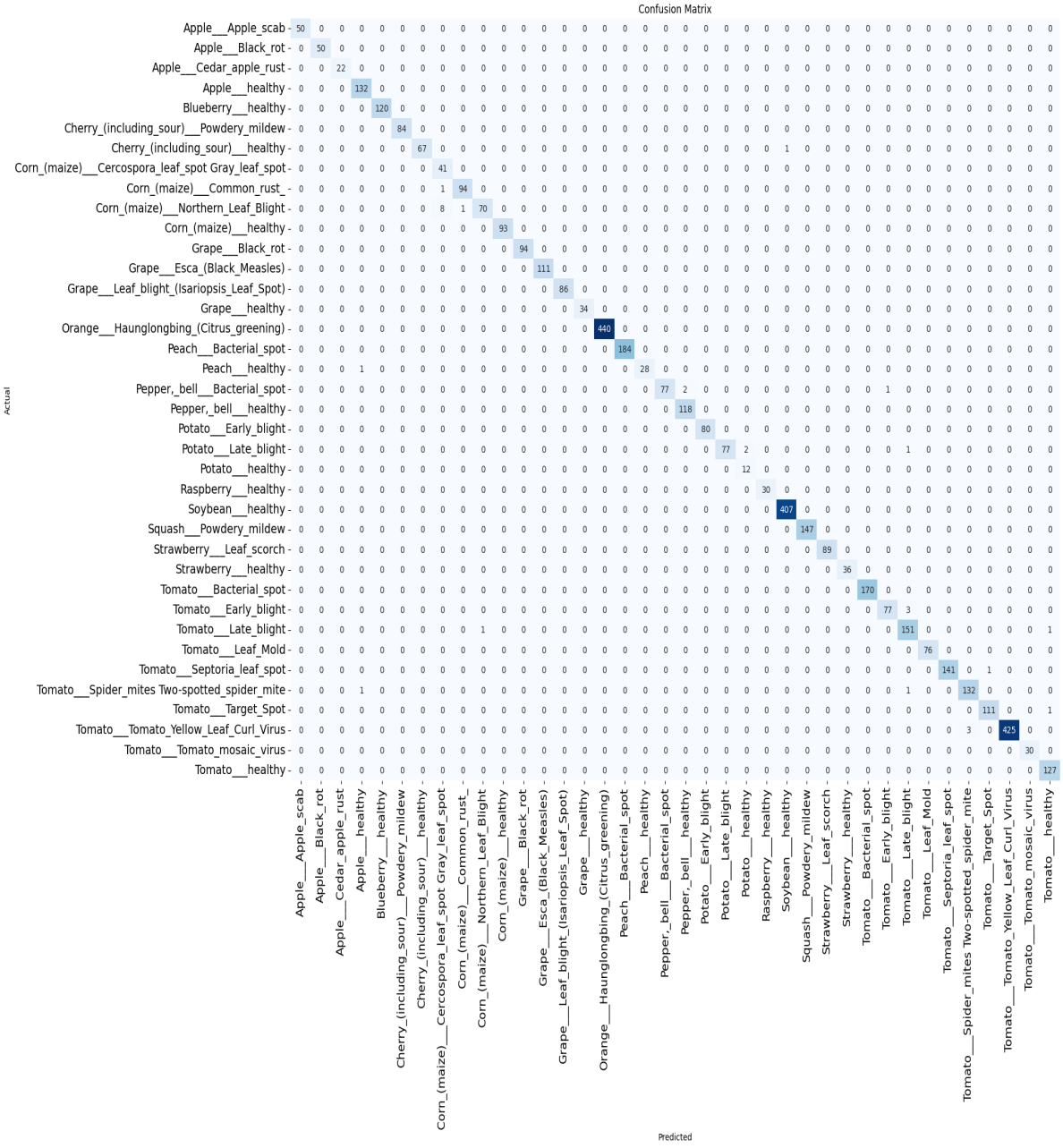

Figure 4.24: Confusion Matrix Inception PlantVillage Dataset

4.3.4 Modified ResNet pretrained model

ResNet [42], proposed by Microsoft researchers is similar to VGG but deeper; it introduced a new concept called residual learning that helps go very deeply. Residual learning allows training efficiently very deep networks having up to a depth of 152 layers.

We tuned Resnet model by removing the top layer (instruction in keras : include-top=False)

and unfrozen the model(instruction in keras trainable = true), the input data of this model (tomato dataset) has a shape (128,128,3) and (256,256,3) with plantVillage dataset it was normalized and augmented by a horizontal flip, rotation, zoom and rescaling values.
we added a batch normalization layer followed by two fully connected layers the first with 256 units and the second with 512 units then we added a dropout with 40% , and changed the output layer to have ten units (in case of tomato Dataset) and 38 units (in case of plantVillage dataset) according to the number of classes . We will be using the Rectified linear unit (ReLU) activation function for all the layers except the final output layer with a softmax function, using Early Stopping 2.2.5.3 The training was done during 11 epochs in **17 minutes** with tomato Dataset, and for PlantVillage Dataset it took 15 epochs in **4 hours** using google colab pro 3.4.6.1, the models are optimized by Adam Optimizer with learning rate equal to 0.001, The hyperparameters of the architecture are described below.

Parameter	**Tomato Dataset**	**PlantVillage Dataset**
Number of epochs	11	15
batch size	30	30
Optimizer	Adam	Adam
Dropout rate	0.4	0.4
Learning rate	0.001	0.001

Table 4.6: hyperparameters of ResNet

- We obtained from this experiment an accuracy of 94% on validation set and 97.2% on test set with tomato dataset.

- We obtained from this experiment an accuracy of 99.52% on validation set and 99.45% on test set with PlantVillage dataset.

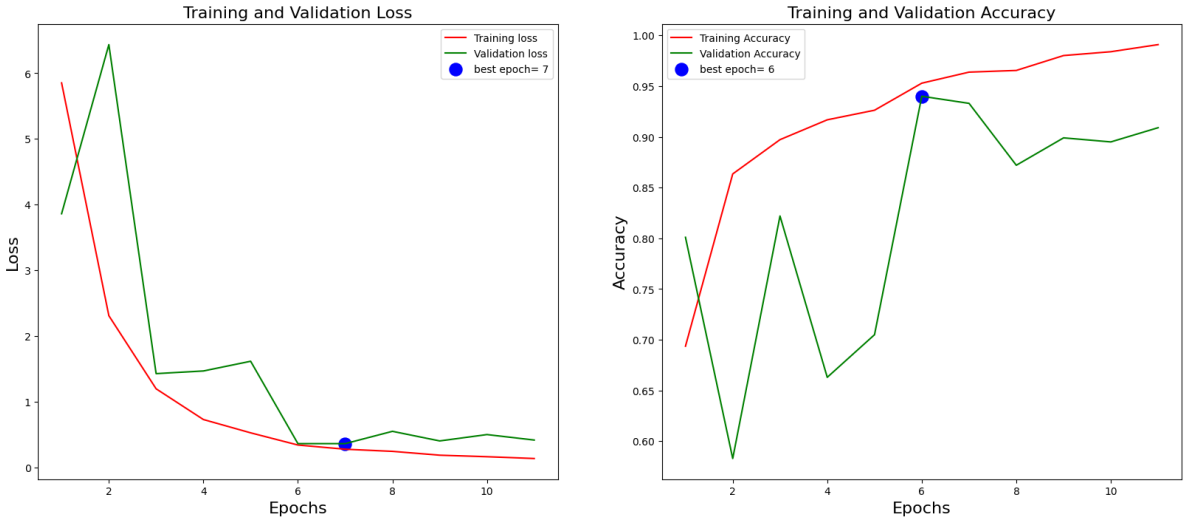

Figure 4.25: Training and validation loss and accuracy ResNet Tomato Dataset

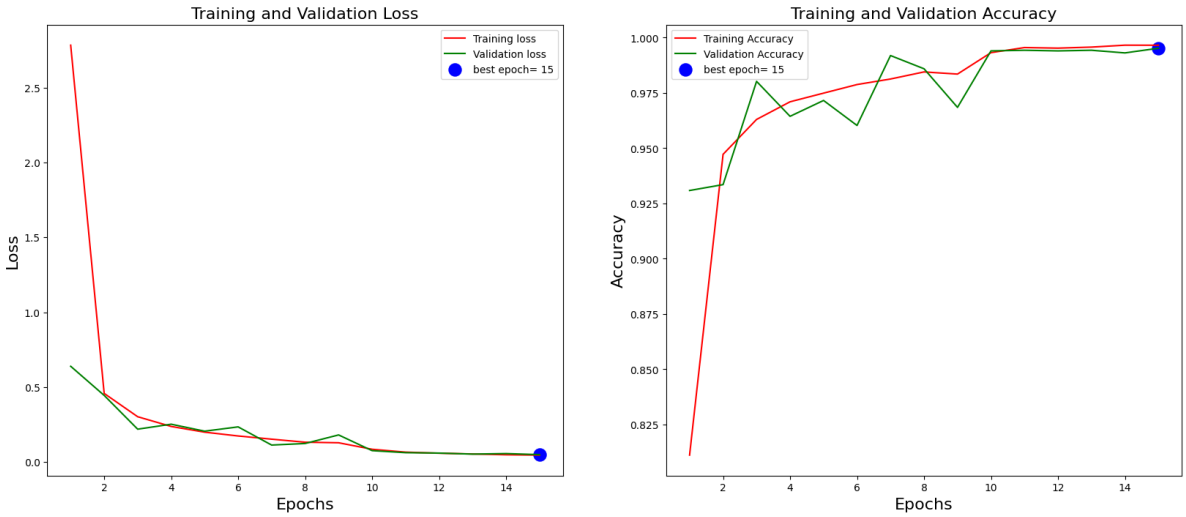

Figure 4.26: Training and validation loss and accuracy ResNet PlantVillage Dataset

4.3.5 Modified Xception pretrained model

We tuned Xception model by removing the top layer (instruction in keras : include-top=False) and unfrozen the model(instruction in keras trainable = true), the input data of both models has a shape of (256,256,3) it was normalized and augmented by a horizontal flip, rotation, zoom and rescaling values.

we added a batch normalization layer followed by two fully connected layers the first with 256 units and the second with 512 units then we added a dropout with 40% , and changed the output layer to have ten units (in case of tomato Dataset) and 38 units (in case of plantVillage dataset)

according to the number of classes . We will be using the Rectified linear unit (ReLU) activation function for all the layers except the final output layer with a softmax function, using Early Stopping 2.2.5.3 The training was done during 26 epochs in **1 hours and 43 minutes** with tomato Dataset, and for PlantVillage Dataset it took 22 epochs in **6 hours** using google colab pro 3.4.6.1, the models are optimized by Adam Optimizer with learning rate equal to 0.001, The hyperparameters of the architecture are described below.

Parameter	**Tomato Dataset**	**PlantVillage Dataset**
Number of epochs	26	22
batch size	30	30
Optimizer	Adam	Adam
Dropout rate	0.4	0.4
Learning rate	0.001	0.001

Table 4.7: hyperparameters of Xception

- We obtained from this experiment an accuracy of 99.3% on validation set and 99.8% on test set with tomato dataset.

- We obtained from this experiment an accuracy of 99.79% on validation set and 99.52% on test set with PlantVillage dataset.

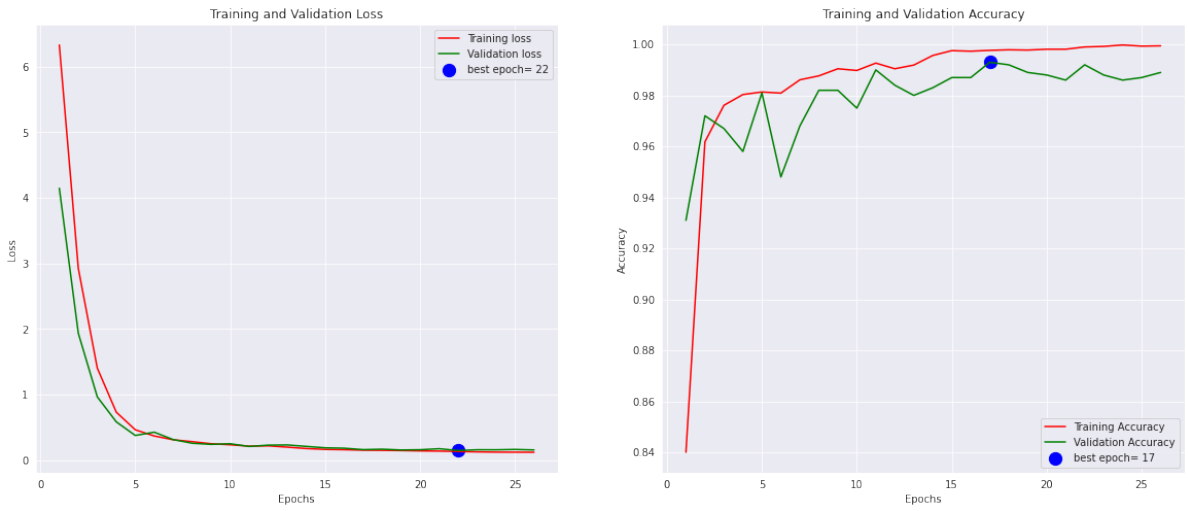

Figure 4.27: Training and validation loss and accuracy Xception Tomato Dataset

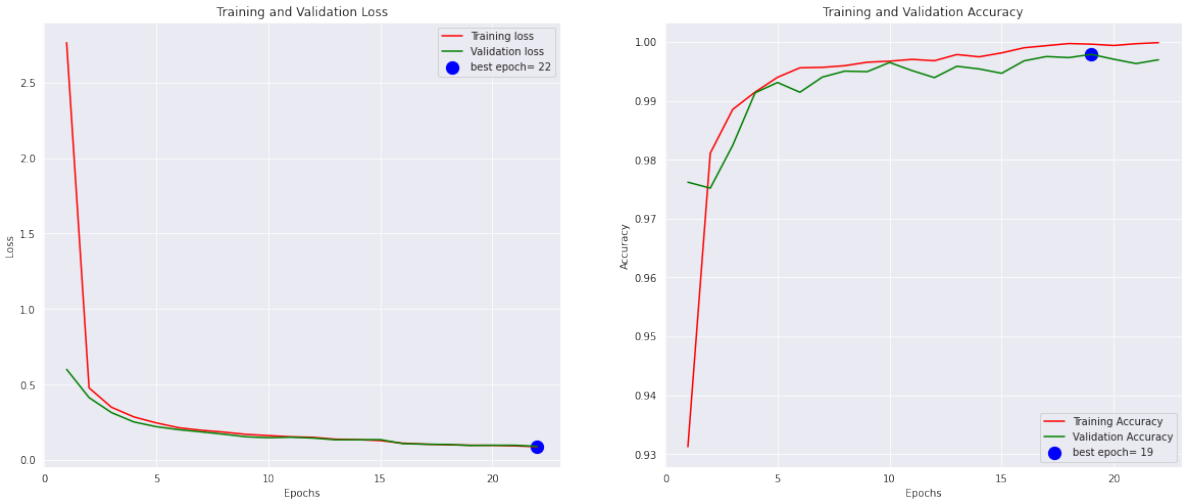

Figure 4.28: Training and validation loss and accuracy Xception PlantVillage Dataset

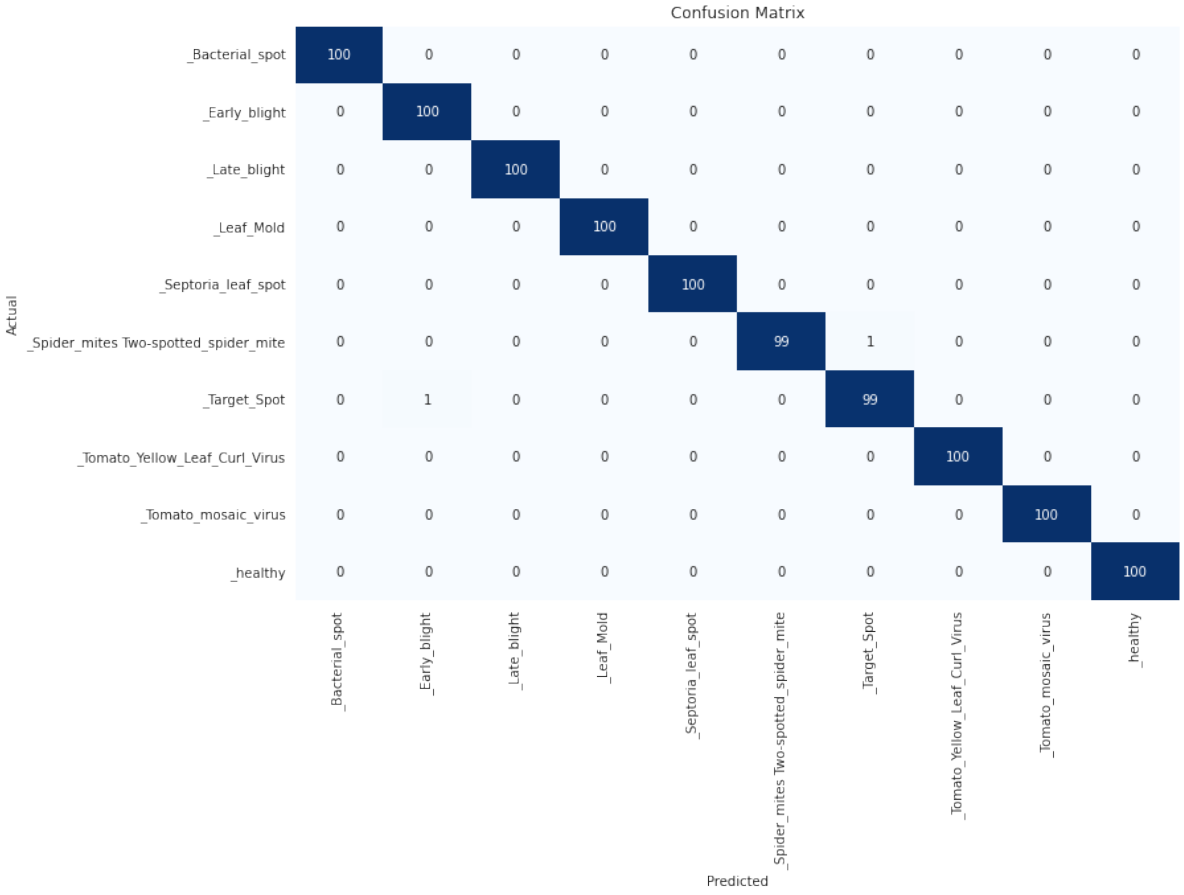

Figure 4.29: Confusion Matrix Xception Tomato Dataset

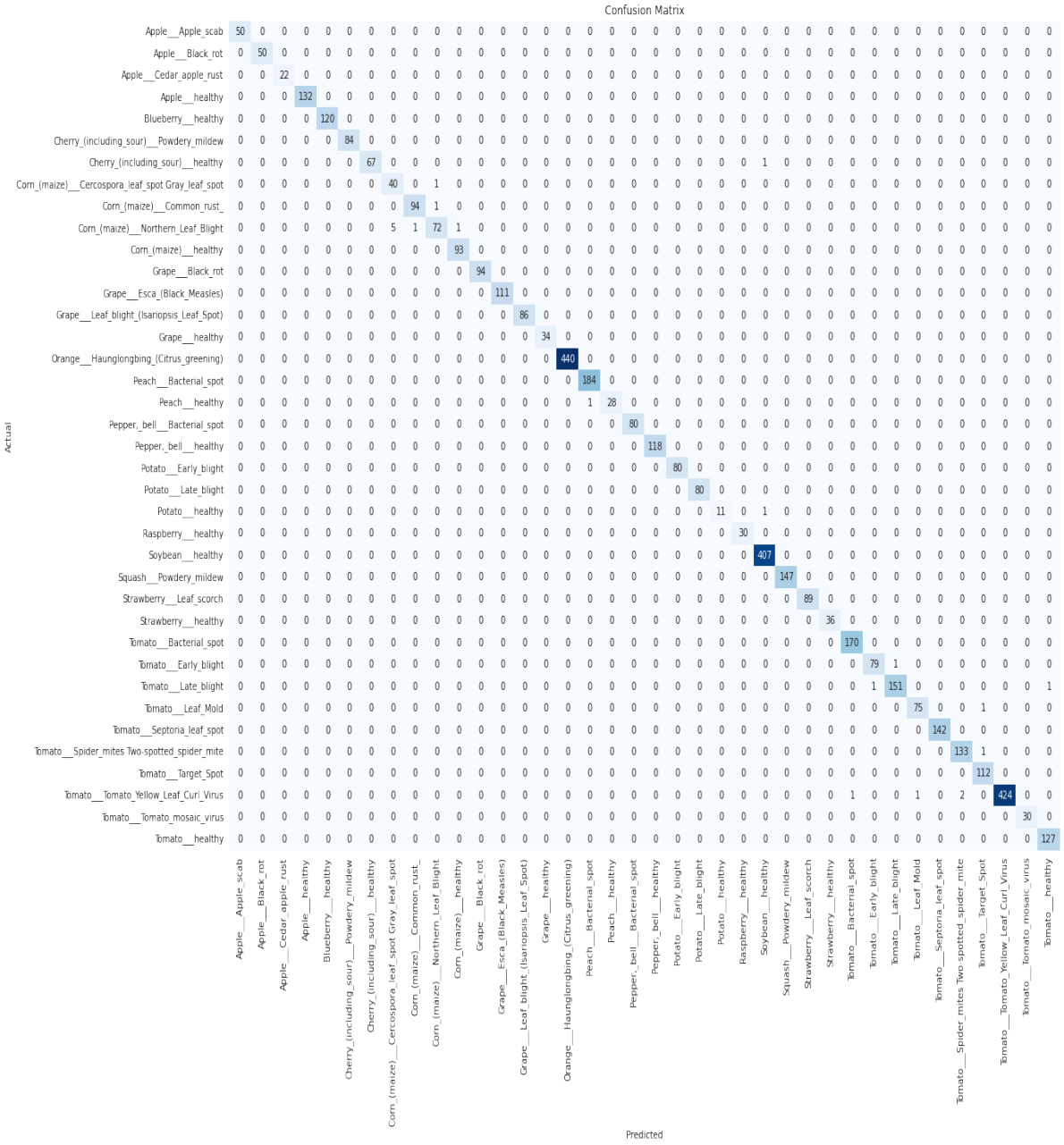

Figure 4.30: Confusion Matrix Xception PlantVillage Dataset

4.3.6 Modified MobileNet pretrained model

We tuned MobileNet model by removing the top layer (instruction in keras : include-top=False) and unfrozen the model(instruction in keras trainable = true), the input data of this model (tomato dataset) has a shape (128,128,3) and (256,256,3) with plantVillage dataset it was normalized and augmented by a horizontal flip, rotation, zoom and rescaling values.
we added a batch normalization layer followed by two fully connected layers the first with 256

units and the second with 512 units then we added a dropout with 40% , and changed the output layer to have ten units (in case of tomato Dataset) and 38 units (in case of plantVillage dataset) according to the number of classes . We will be using the Rectified linear unit (ReLU) activation function for all the layers except the final output layer with a softmax function, using Early Stopping 2.2.5.3 The training was done during 24 epochs in **23 minutes** with tomato Dataset, and for PlantVillage Dataset it took 20 epochs in **3 hours 50 minutes** using google colab pro 3.4.6.1, the models are optimized by Adam Optimizer with learning rate equal to 0.001, The hyperparameters of the architecture are described below.

Parameter	**Tomato Dataset**	**PlantVillage Dataset**
Number of epochs	24	20
batch size	30	30
Optimizer	Adam	Adam
Dropout rate	0.4	0.4
Learning rate	0.001	0.001

Table 4.8: hyperparameters of MobileNet

- We obtained from this experiment an accuracy of 96.8% on validation set and 97.9% on test set with tomato dataset.

- We obtained from this experiment an accuracy of 99.46% on validation set and 99.01% on test set with PlantVillage dataset.

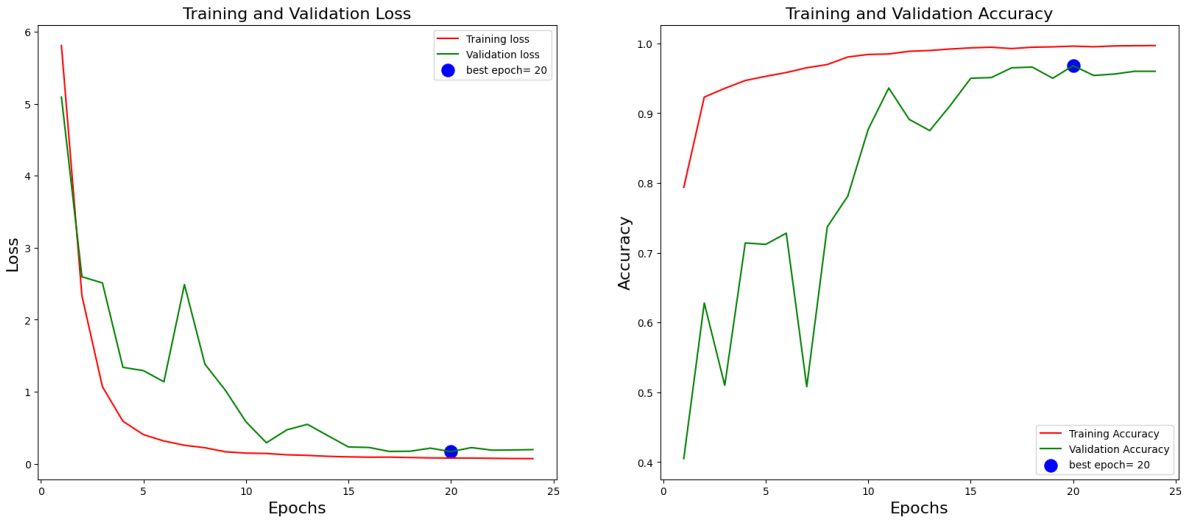

Figure 4.31: Training and validation loss and accuracy MobileNet Tomato Dataset

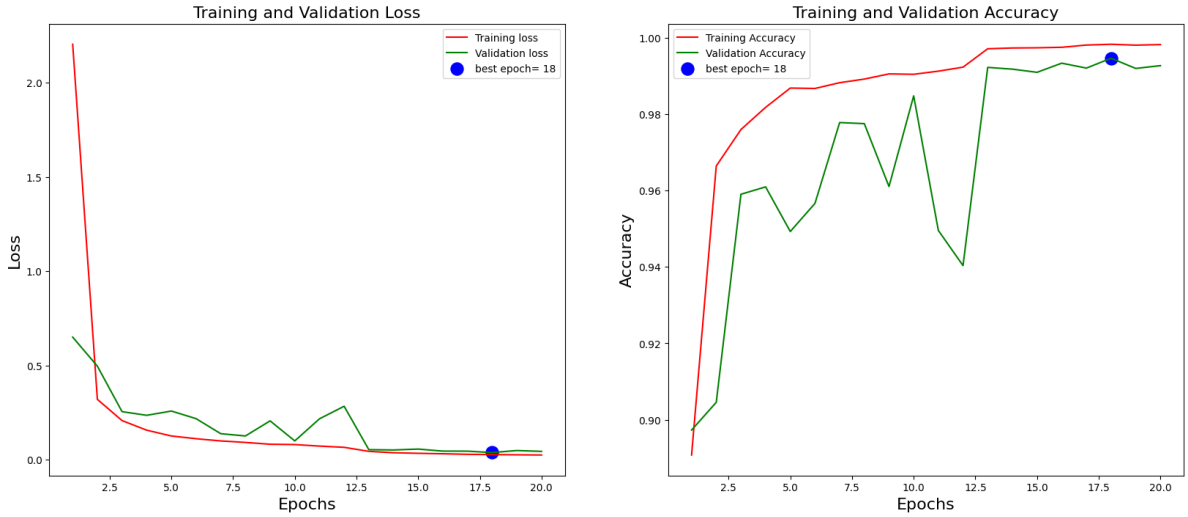

Figure 4.32: Training and validation loss and accuracy MobileNet PlantVillage Dataset

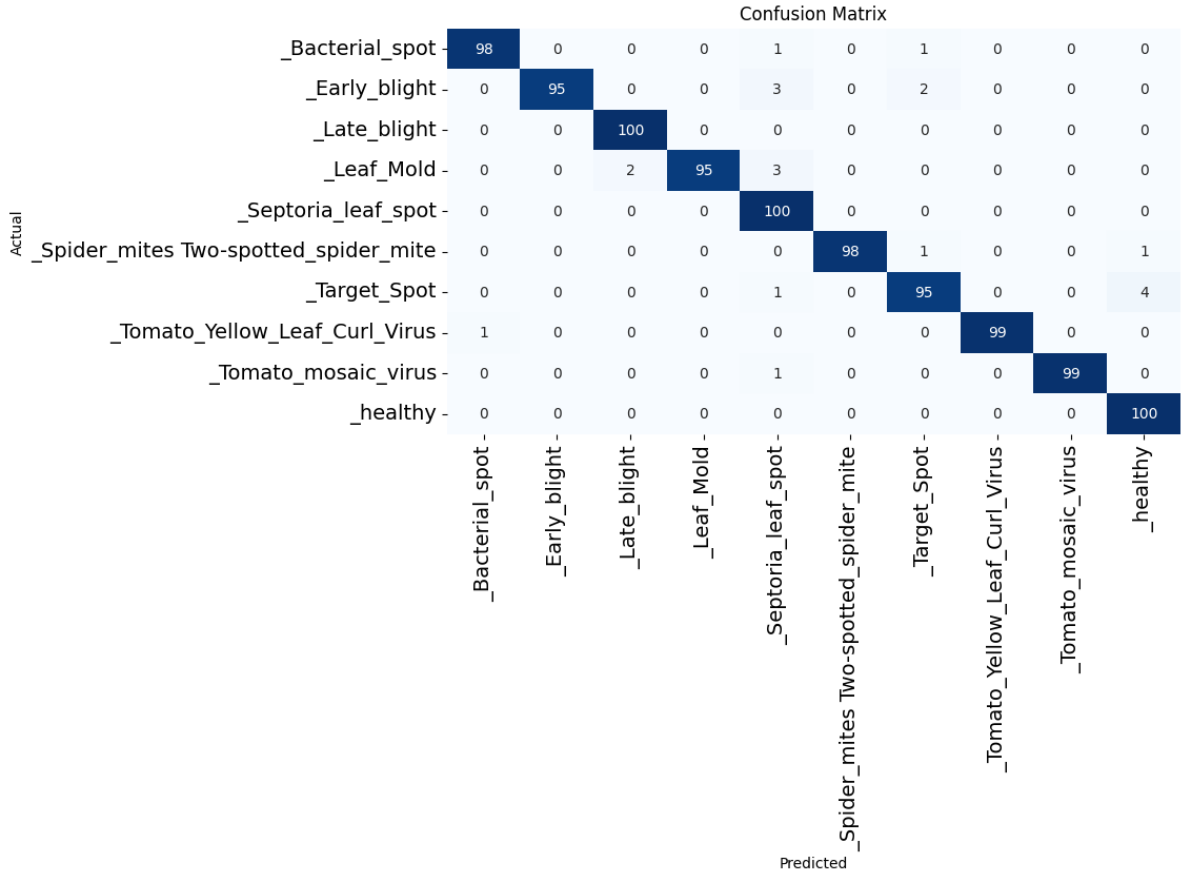

Figure 4.33: Confusion Matrix MobileNet Tomato Dataset

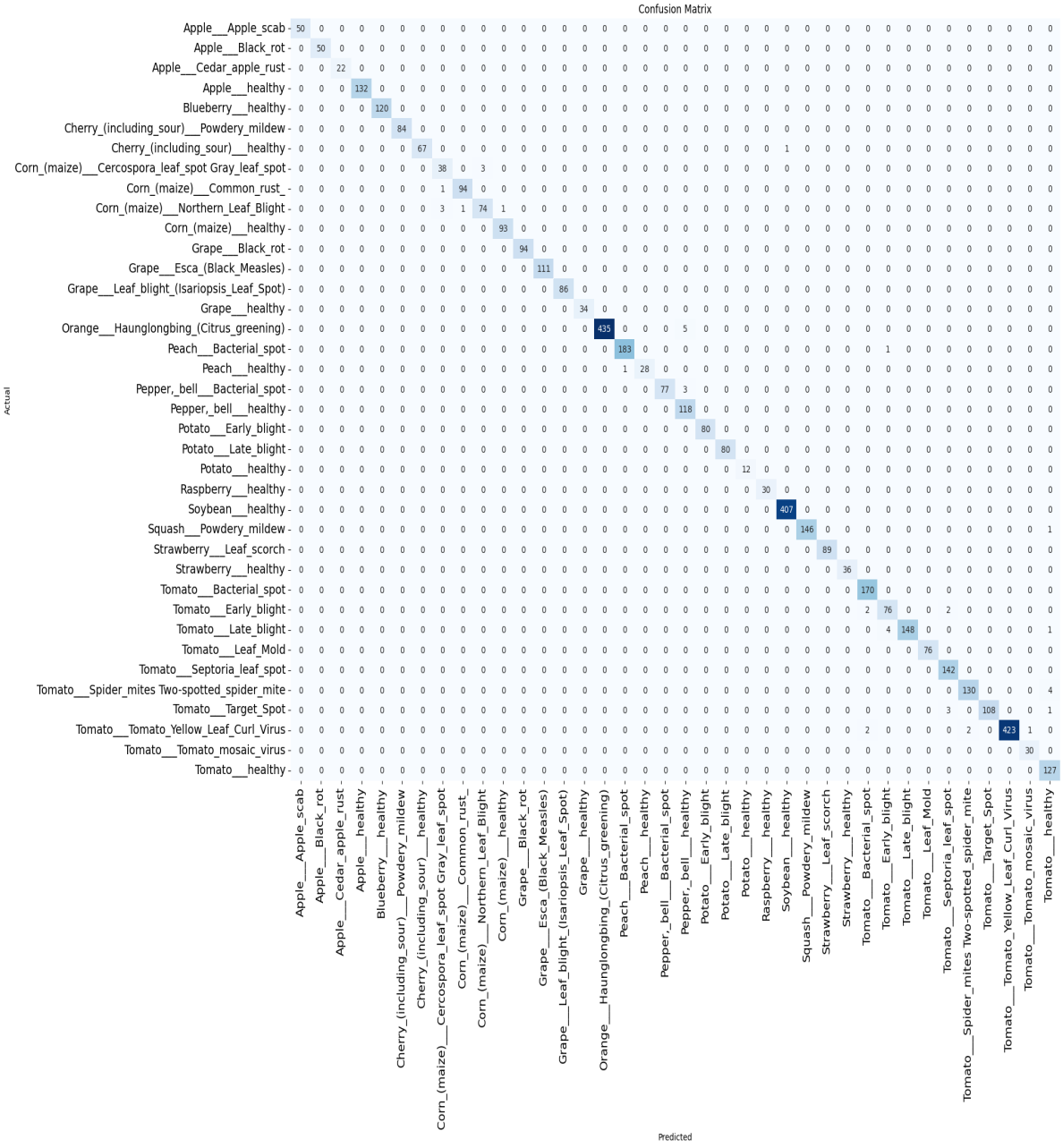

Figure 4.34: Confusion Matrix MobileNet PlantVillage Dataset

4.3.7 Modified DenseNet pretrained model

We tuned DenseNet model by removing the top layer (instruction in keras : include-top=False) and unfrozen the model(instruction in keras trainable = true), the input data of both models has a shape of (256,256,3) it was normalized and augmented by a horizontal flip, rotation, zoom and rescaling values.

we added a batch normalization layer followed by two fully connected layers the first with 256

units and the second with 512 units then we added a dropout with 40% , and changed the output layer to have ten units (in case of tomato Dataset) and 38 units (in case of plantVillage dataset) according to the number of classes . We will be using the Rectified linear unit (ReLU) activation function for all the layers except the final output layer with a softmax function, using Early Stopping 2.2.5.3 The training was done during 37 epochs in **3 hours and 27 minutes** with tomato Dataset, and for PlantVillage Dataset it took 15 epochs in **3 hours and 15 minutes** using google colab pro 3.4.6.1, the models are optimized by Adam Optimizer with learning rate equal to 0.001, The hyperparameters of the architecture are described below.

Parameter	**Tomato Dataset**	**PlantVillage Dataset**
Number of epochs	37	15
batch size	30	30
Optimizer	Adam	Adam
Dropout rate	0.4	0.4
Learning rate	0.001	0.001

Table 4.9: hyperparameters of DenseNet

- We obtained from this experiment an accuracy of 99.5% on validation set and 99.8% on test set with tomato dataset.

- We obtained from this experiment an accuracy of 99.86% on validation set and 99.59% on test set with PlantVillage dataset.

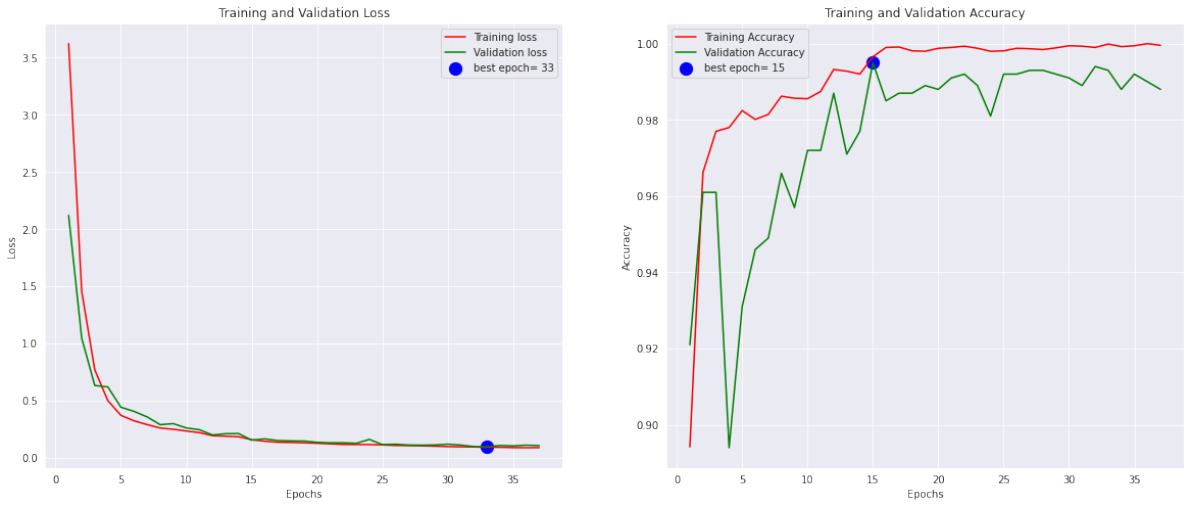

Figure 4.35: Training and validation loss and accuracy DenseNet Tomato Dataset

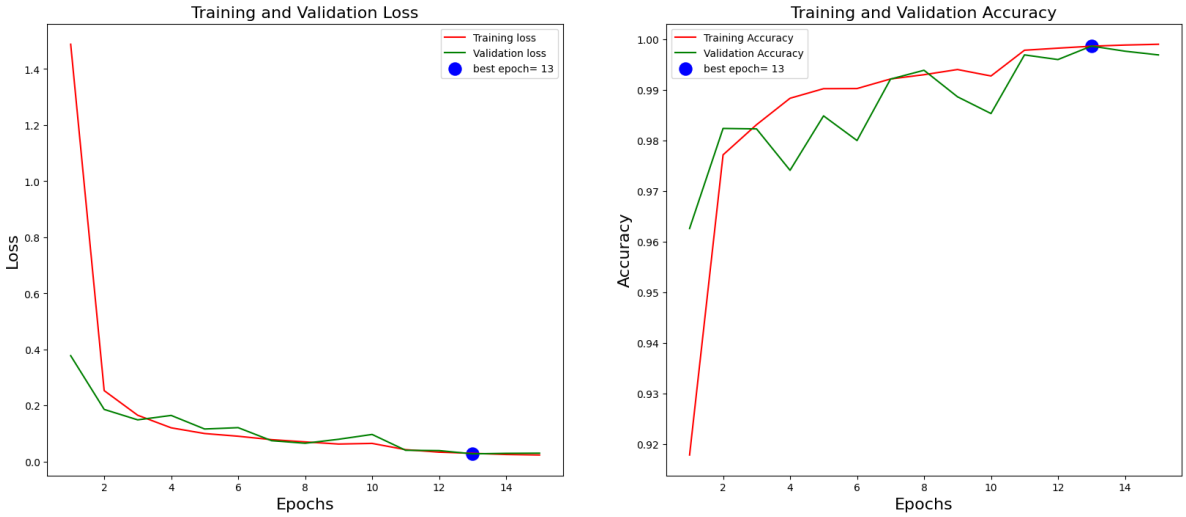

Figure 4.36: Training and validation loss and accuracy DenseNet PlantVillage Dataset

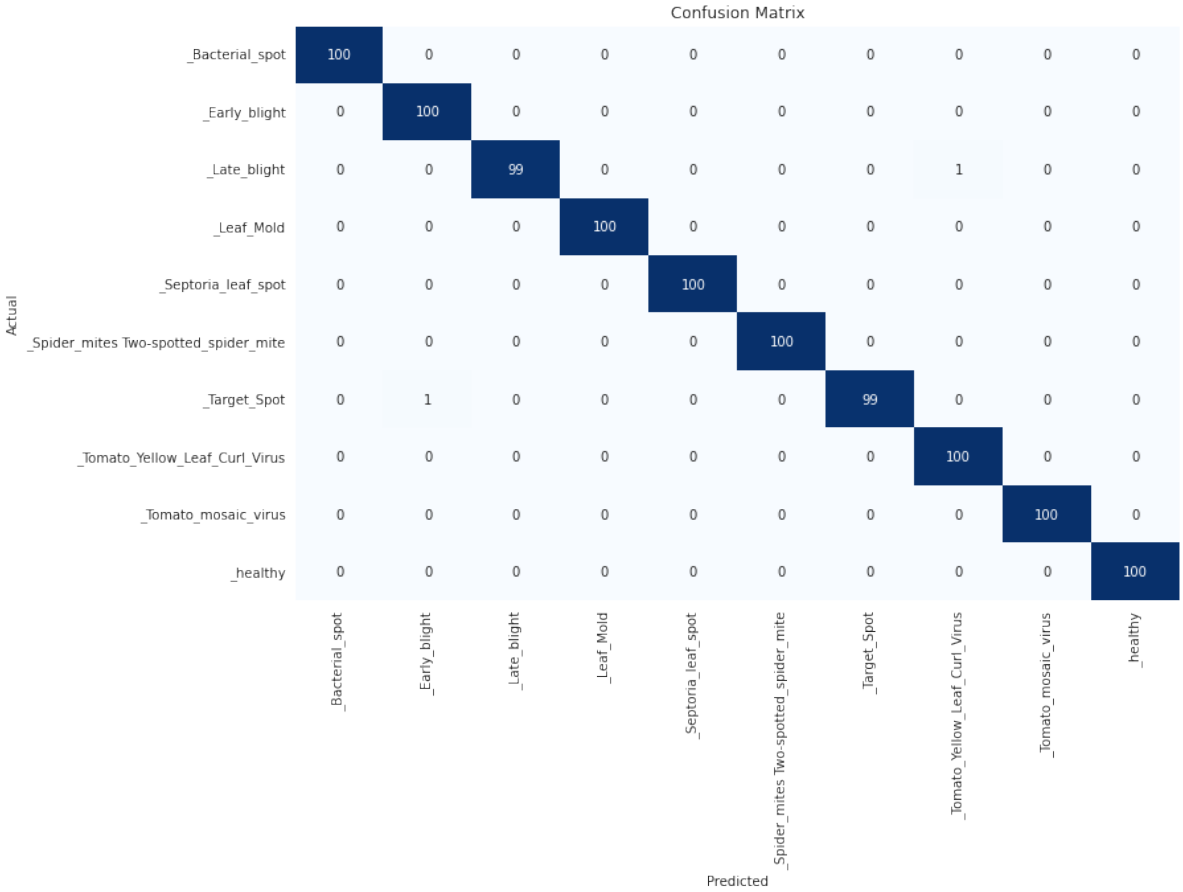

Figure 4.37: Confusion Matrix DenseNet Tomato Dataset

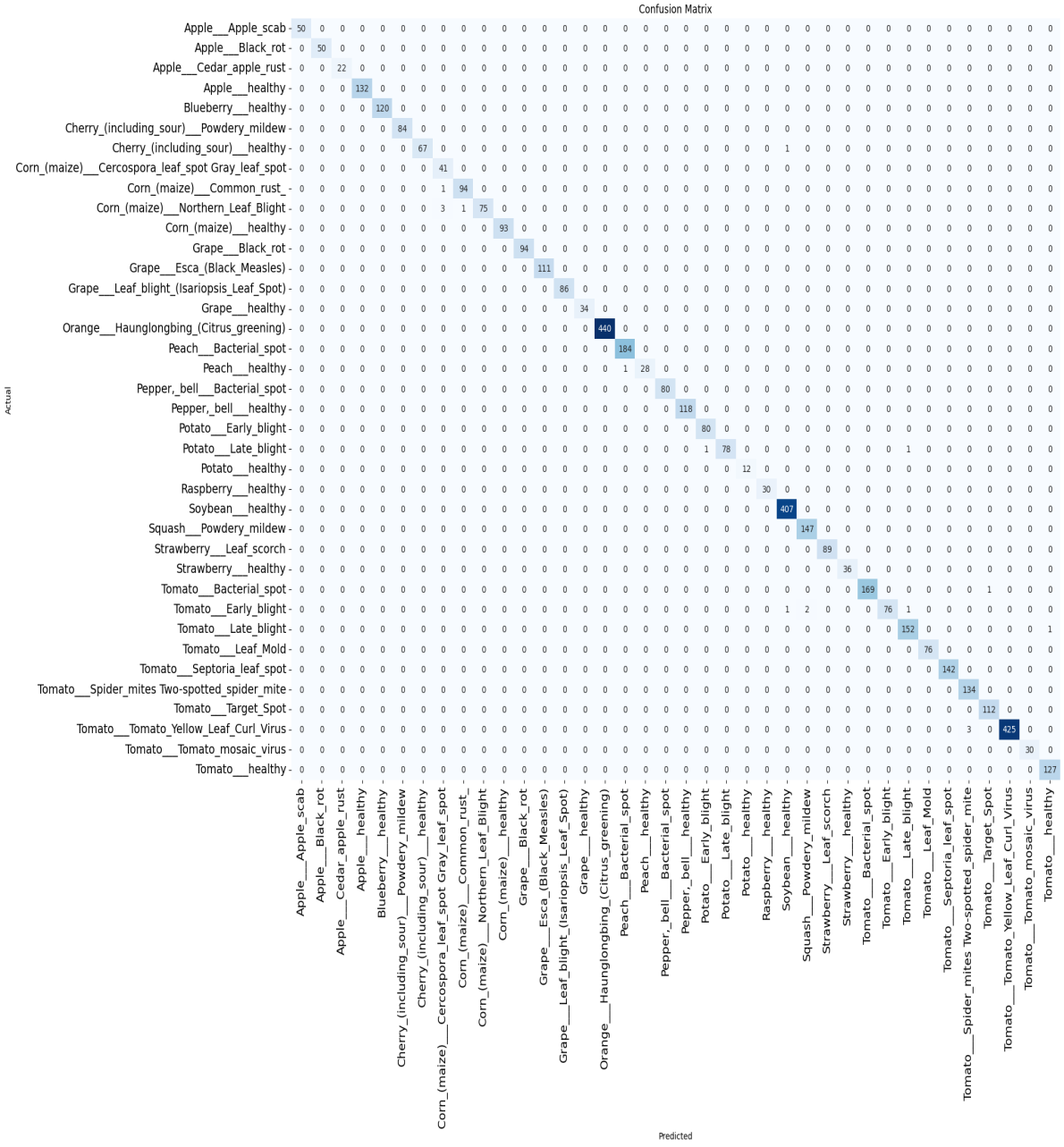

Figure 4.38: Confusion Matrix DenseNet PlantVillage Dataset

4.3.8 Modified EfficientNet B3 pretrained model

We tuned EfficientNet B3 model by removing the top layer (instruction in keras : include-top=False) and unfrozen the model(instruction in keras trainable = true), the input data of both models has a shape of (256,256,3) it was normalized and augmented by a horizontal flip, rotation, zoom and rescaling values.

we added a batch normalization layer followed by two fully connected layers the first with 256

units and the second with 512 units then we added a dropout with 40% , and changed the output layer to have ten units (in case of tomato Dataset) and 38 units (in case of plantVillage dataset) according to the number of classes . We will be using the Rectified linear unit (ReLU) activation function for all the layers except the final output layer with a softmax function, using Early Stopping 2.2.5.3 The training was done during 40 epochs in **28 minutes** with tomato Dataset, and for PlantVillage Dataset it took 25 epochs in **4 hours and 47 minutes** using google colab pro 3.4.6.1, the models are optimized by Adam Optimizer with learning rate equal to 0.001, The hyperparameters of the architecture are described below.

Parameter	**Tomato Dataset**	**PlantVillage Dataset**
Number of epochs	40	25
batch size	30	30
Optimizer	Adam	Adam
Dropout rate	0.4	0.4
Learning rate	0.001	0.001

Table 4.10: hyperparameters of EfficientNetB3

- We obtained from this experiment an accuracy of 97.8% on validation set and 97.3% on test set with tomato dataset.

- We obtained from this experiment an accuracy of 99.76% on validation set and 99.65% on test set with PlantVillage dataset.

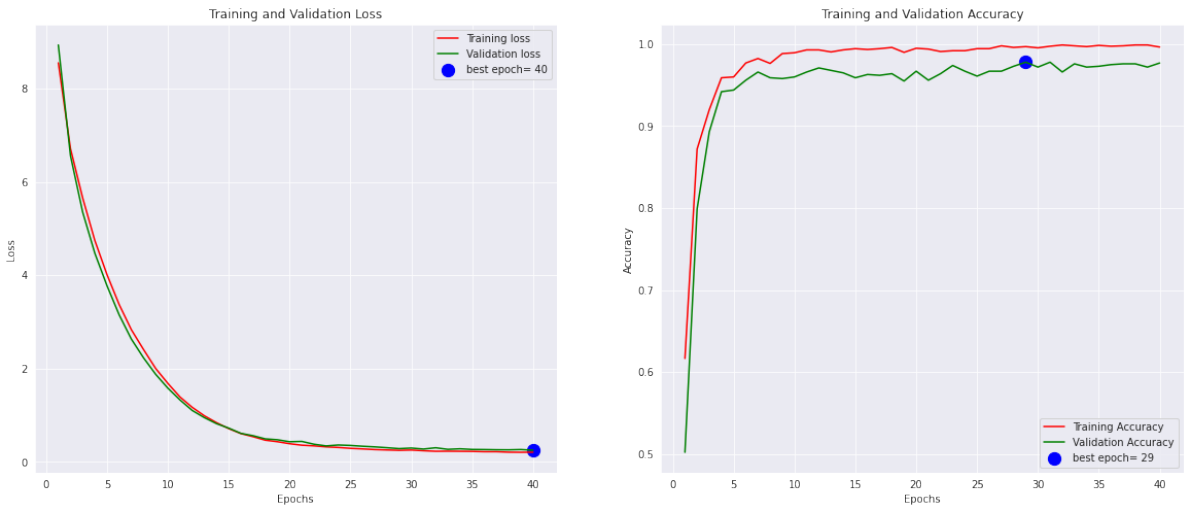

Figure 4.39: Training and validation loss and accuracy EfficientNetB3 Tomato Dataset

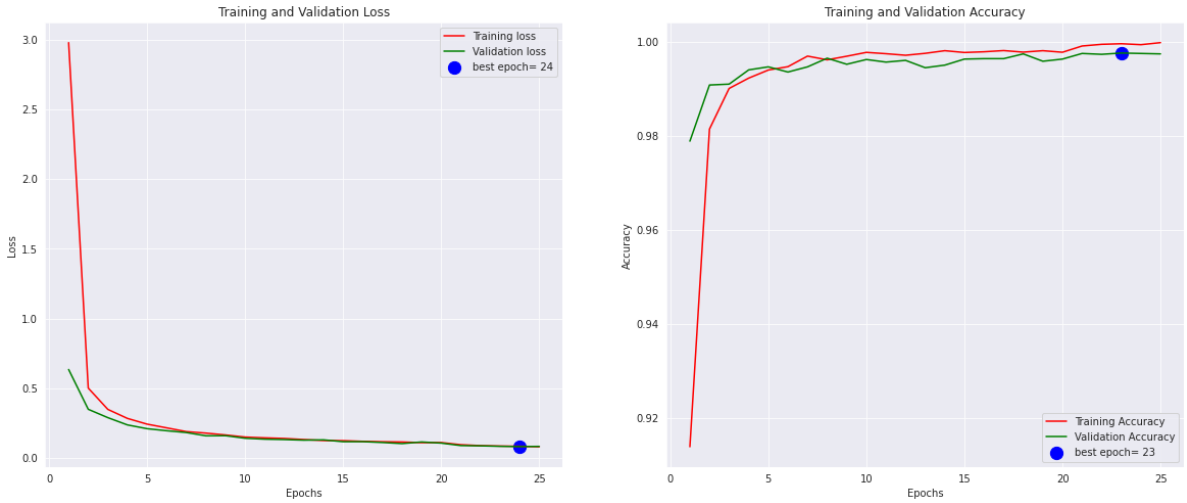

Figure 4.40: Training and validation loss and accuracy EfficientNetB3 PlantVillage Dataset

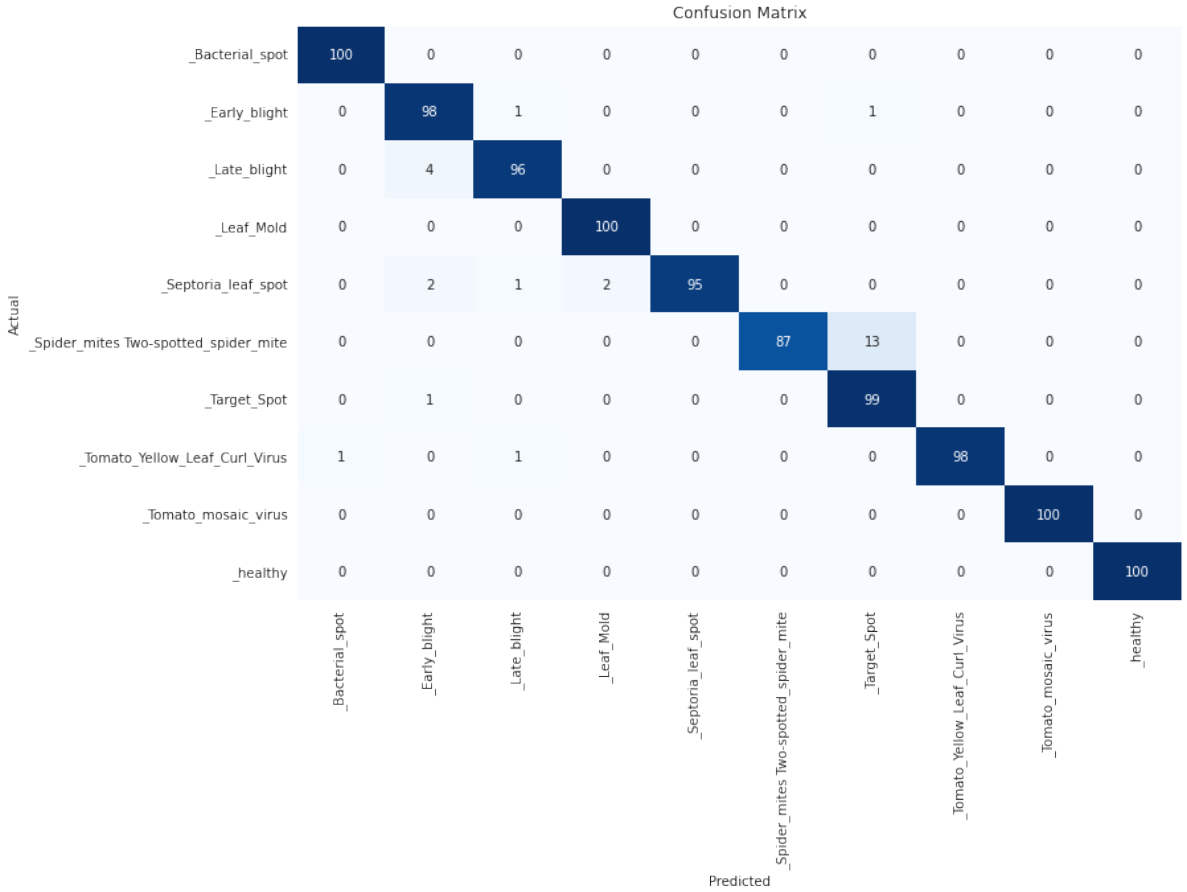

Figure 4.41: Confusion Matrix EfficientNetB3 Tomato Dataset

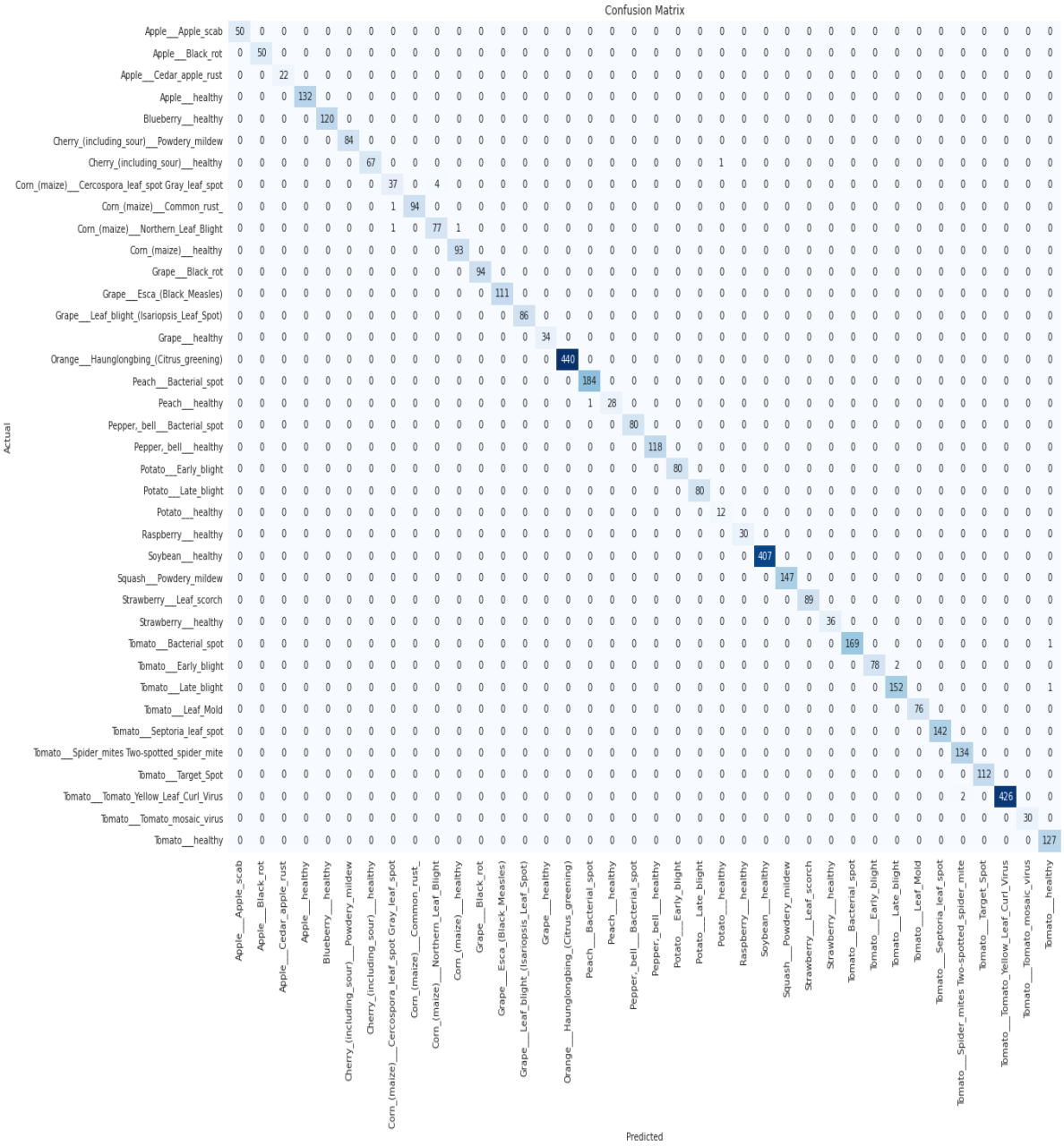

Figure 4.42: Confusion Matrix EfficientNetB3 PlantVillage Dataset

4.3.9 Modified EfficientNetB5 pretrained model

We tuned EfficientNetB5 model by removing the top layer (instruction in keras : include-top=False) and unfrozen the model(instruction in keras trainable = true), the input data of both models has a shape of (256,256,3) it was normalized and augmented by a horizontal flip, rotation, zoom and rescaling values.
we added a batch normalization layer followed by two fully connected layers the first with 256

units and the second with 512 units then we added a dropout with 40% , and changed the output layer to have ten units (in case of tomato Dataset) and 38 units (in case of plantVillage dataset) according to the number of classes . We will be using the Rectified linear unit (ReLU) activation function for all the layers except the final output layer with a softmax function, using Early Stopping 2.2.5.3 The training was done during 40 epochs in **53 minutes** with tomato Dataset, and for PlantVillage Dataset it took 10 epochs in **3 hours and 15 minutes** using google colab pro 3.4.6.1, the models are optimized by Adam Optimizer with learning rate equal to 0.001, The hyperparameters of the architecture are described below.

Parameter	**Tomato Dataset**	**PlantVillage Dataset**
Number of epochs	40	10
batch size	30	30
Optimizer	Adam	Adam
Dropout rate	0.4	0.4
Learning rate	0.001	0.001

Table 4.11: hyperparameters of EfficientNetB5

- We obtained from this experiment an accuracy of 97.5% on validation set and 98.4% on test set with tomato dataset.

- We obtained from this experiment an accuracy of 99.77% on validation set and 99.65% on test set with PlantVillage dataset.

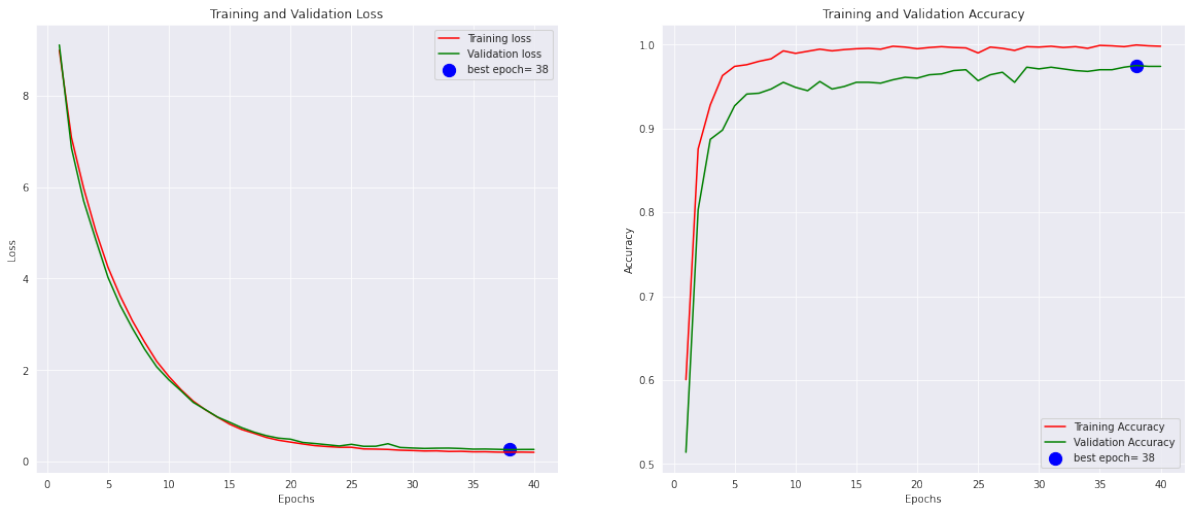

Figure 4.43: Training and validation loss and accuracy EfficientNetB5 Tomato Dataset

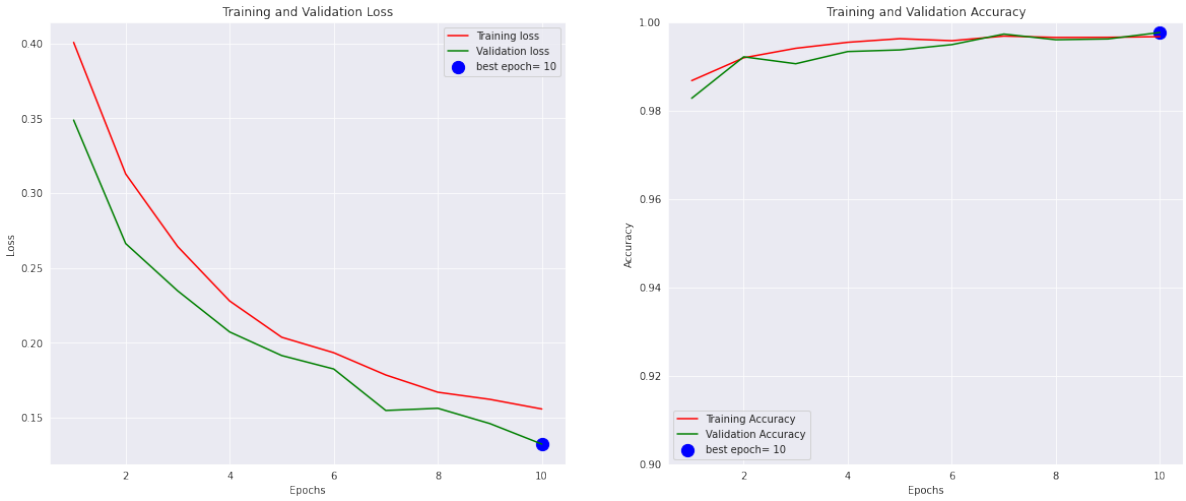

Figure 4.44: Training and validation loss and accuracy EfficientNetB5 PlantVillage Dataset

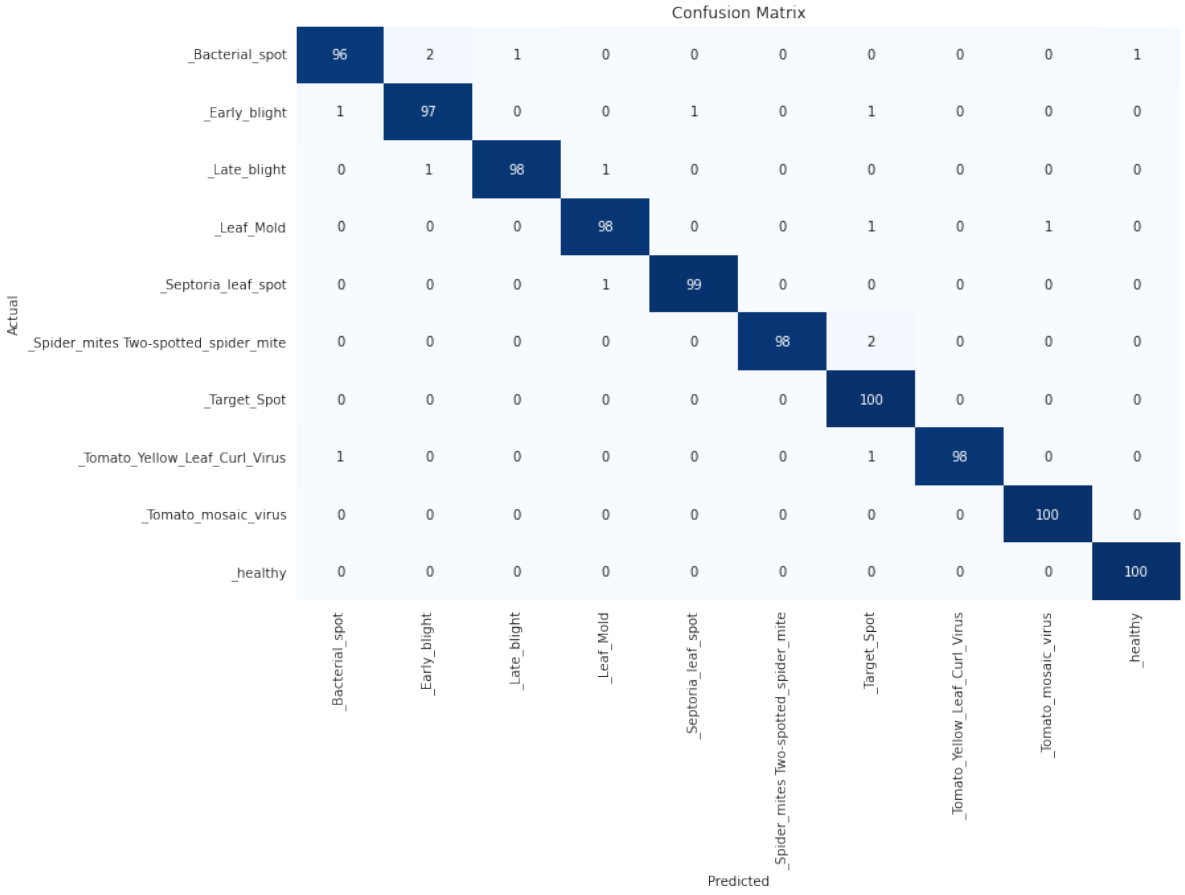

Figure 4.45: Confusion Matrix EfficientNetB5 Tomato Dataset

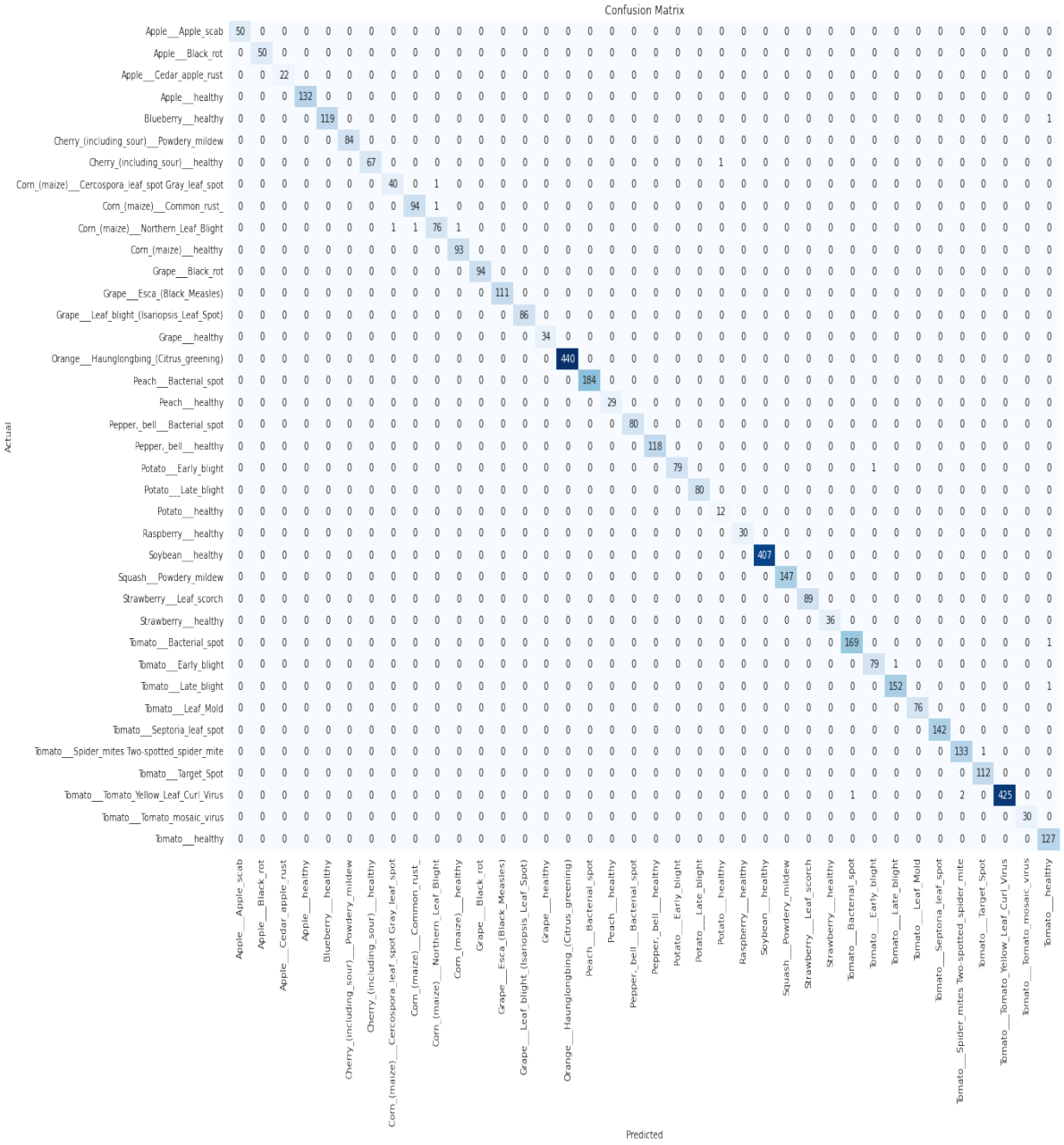

Figure 4.46: Confusion Matrix EfficientNetB5 PlantVillage Dataset

4.4 Vision Transformer

Vision Transformer (ViT) by Google Brain is basically BERT, but applied to images. It attains excellent results compared to state-of-the-art convolutional networks. In order to provide images to the model, each image is split into a sequence of fixed-size patches (typically of resolution 16x16 or 32x32), which are linearly embedded.

4.4.1 ViT B32 from keras

We tuned ViT B32 model by removing the top layer (instruction in keras : include-top=False), the input data of both models has a shape of (256,256,3) it was normalized and augmented by a random flip, random rotation and random zoom.
we added a batch normalization layer followed by two fully connected layers the first with 256 units and the second with 128 units,these two layers has a batch normalization layer in beetween, finally the output layer with ten units (in case of tomato Dataset) and 38 units (in case of plantVillage dataset) according to the number of classes . We will be using the Rectified linear unit (ReLU) activation function for all the layers except the final output layer with a softmax function, using Early Stopping 2.2.5.3 The training was done during 21 epochs in **1 hour and 7 minutes** with tomato Dataset, and for PlantVillage Dataset it took 12 epochs in **3 hours and 18 minutes** using google colab pro 3.4.6.1, the models are optimized by Adam Optimizer with learning rate equal to 0.001, The hyperparameters of the architecture are described below.

Parameter	**Tomato Dataset**	**PlantVillage Dataset**
Number of epochs	21	12
batch size	30	30
Optimizer	Adam	Adam
Dropout rate	/	/
Learning rate	0.001	0.001

Table 4.12: hyperparameters of ViT B32

- We obtained from this experiment an accuracy of 95.9% on validation set and 97.7% on test set with tomato dataset.

- We obtained from this experiment an accuracy of 97.94% on validation set and 86.44% on test set with PlantVillage dataset.

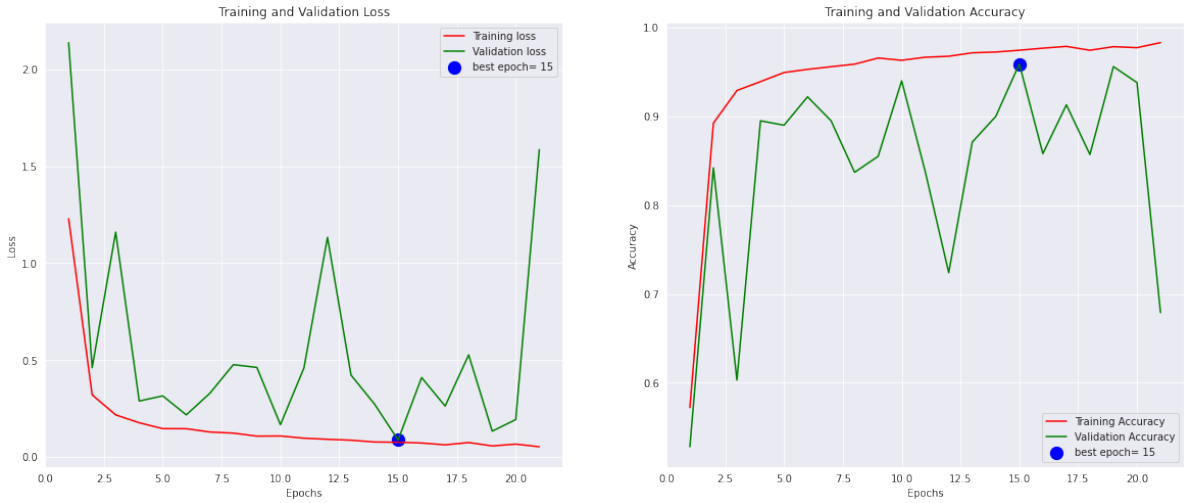

Figure 4.47: Training and validation loss and accuracy ViT B32 Tomato Dataset

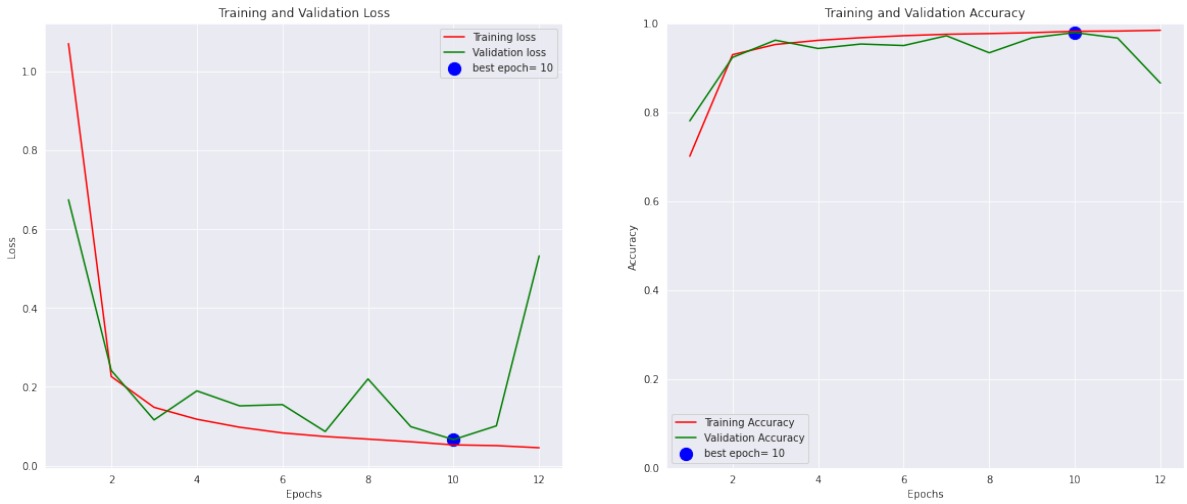

Figure 4.48: Training and validation loss and accuracy ViT B32 PlantVillage Dataset

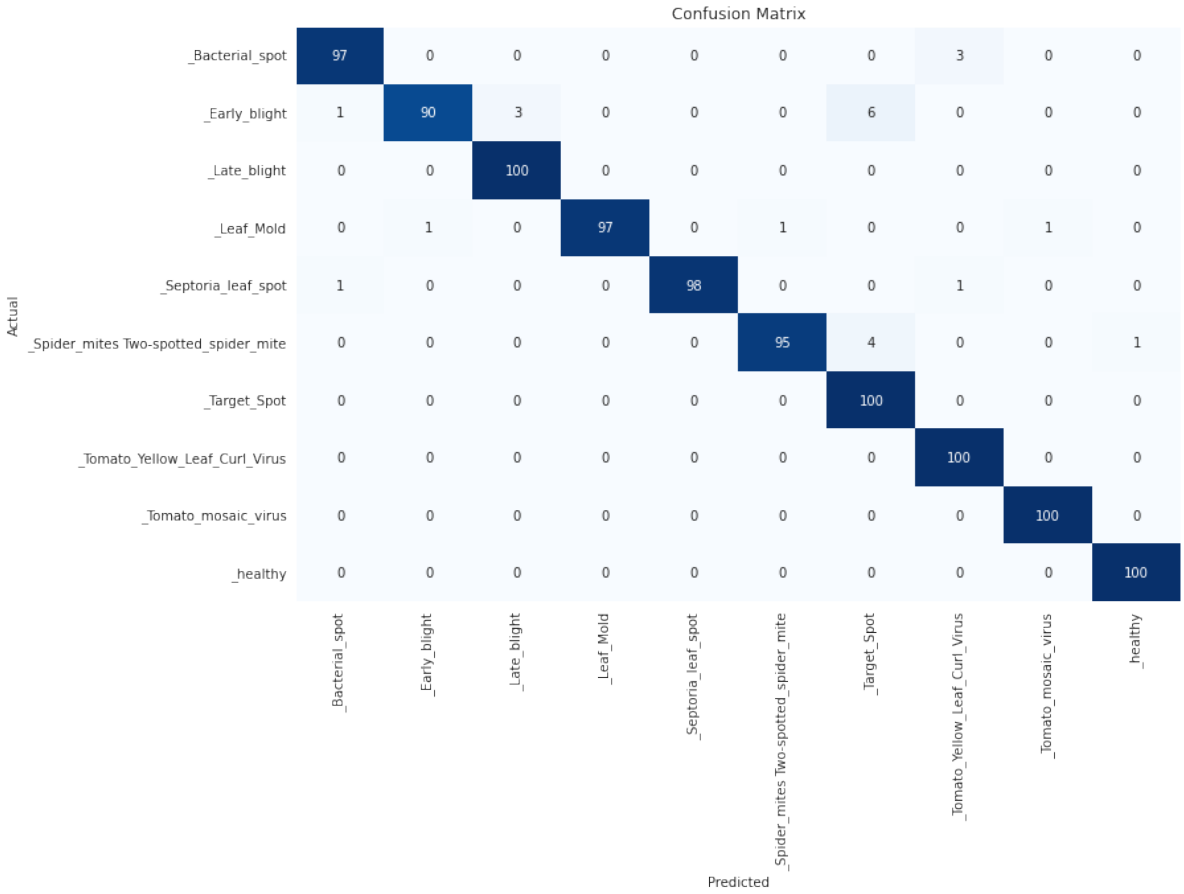

Figure 4.49: Confusion Matrix ViT B32 Tomato Dataset

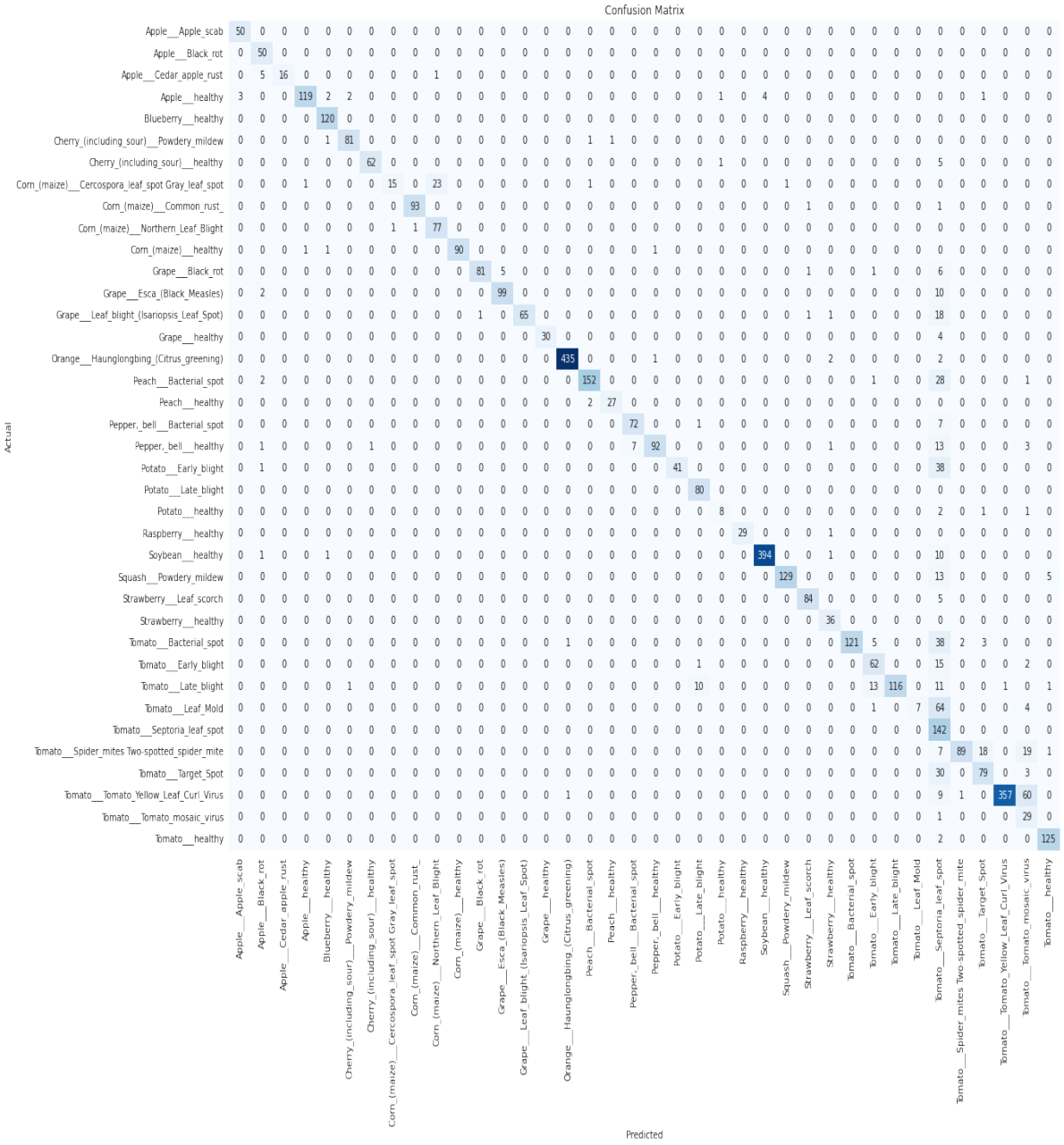

Figure 4.50: Confusion Matrix ViT B32 PlantVillage Dataset

4.4.2 ViT Base patch 16-224(Google) with Pytorch

In the begining we start by loading our dataset to [datasets type] using ImageFolder Dataset Builder, then we load the ViT Feature Extractor offered by the pre-trained model to normalize a dataset,we augment our data using random resize, random horizontal flip and center crop. we modified the top layer of the model by specifying the number of output layers corresponding to the number of our classes (10 classes in case of tomato dataset , 38 classes in case of PlantVillage

dataset). we set a list of arguments that the Trainer will use to fit and evaluate our model. the training was done in 13 minutes(tomato dataset) and 1 hour(PlantVillage datatset) during 3 epochs using google colab pro 3.4.6.1

- We obtained from this experiment an accuracy of 99.1% and a loss of 0.07 on validation set with tomato dataset.

- We obtained from this experiment an accuracy of 99.7% and a loss of 0.01 on validation set with PlantVillage dataset.

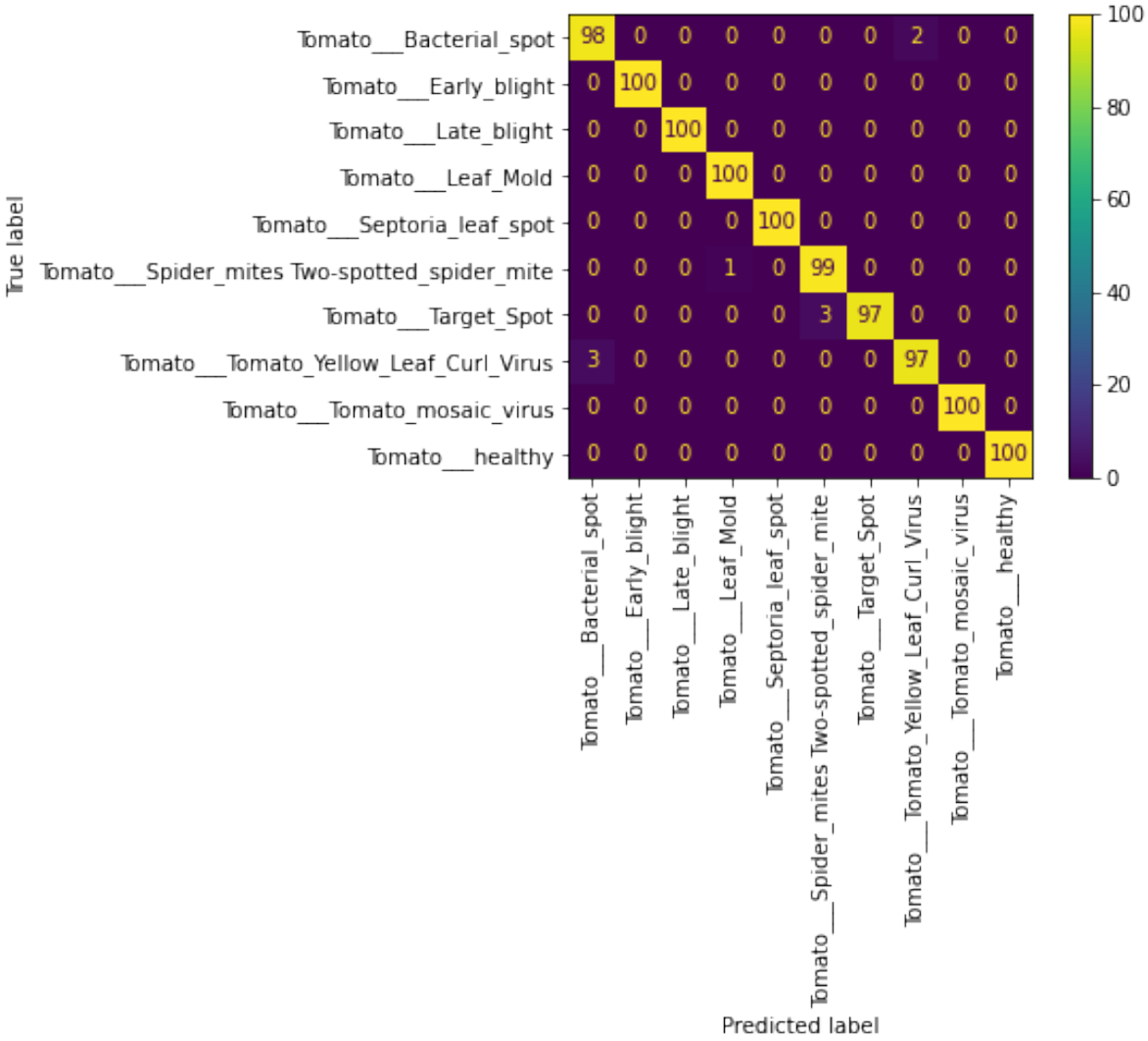

Figure 4.51: Confusion Matrix ViT Base patch 16-224 Tomato Dataset

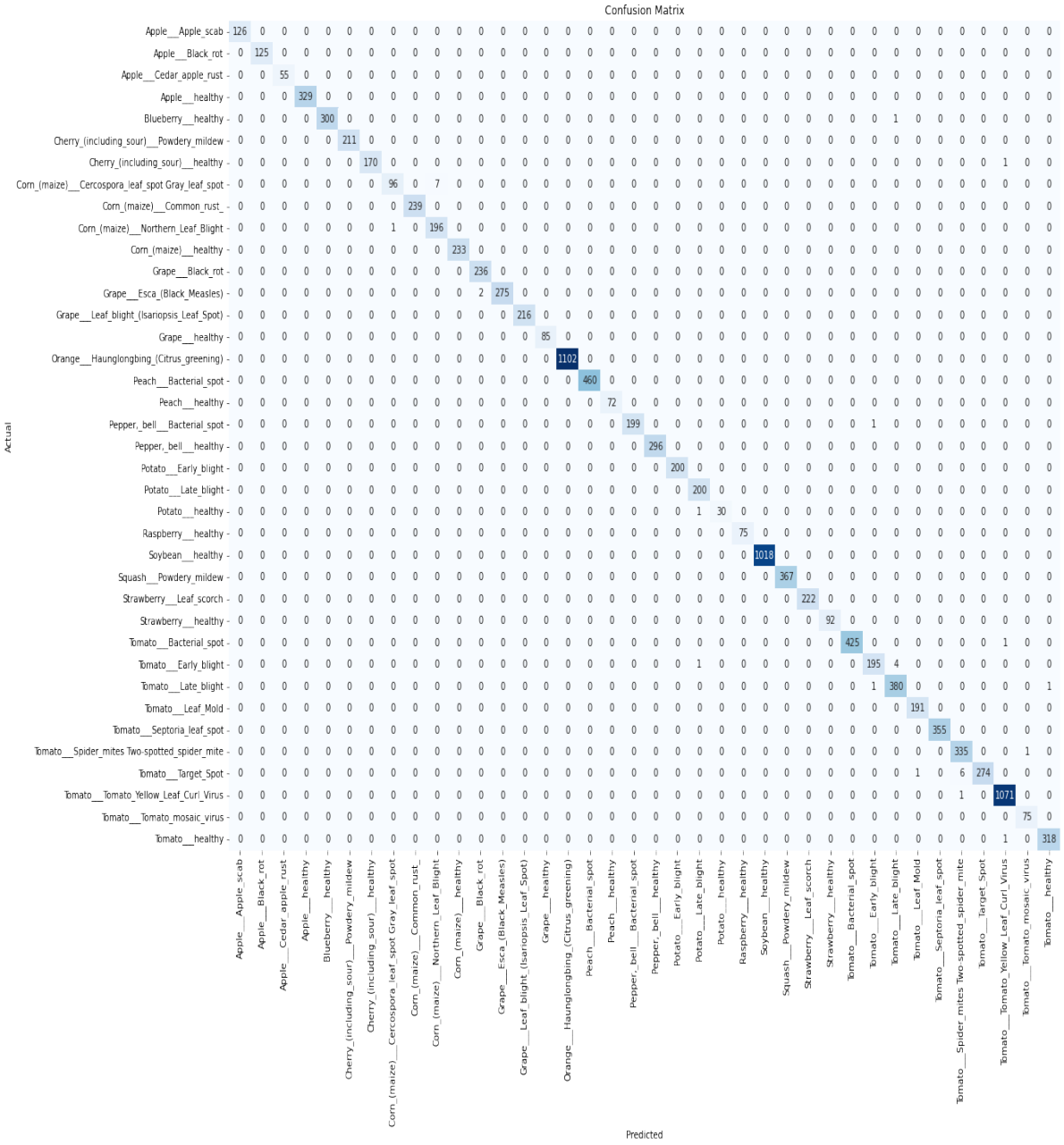

Figure 4.52: Confusion Matrix ViT Base patch 16-224 PlantVillage Dataset

4.4.3 ViT BeiT patch 16-224(Microsoft) with Pytorch

In the begining we start by loading our dataset to [datasets type] using ImageFolder Dataset Builder, then we load the ViT Feature Extractor offered by the pre-trained model to normalize a dataset,we augment our data using random resize, random horizontal flip and center crop. we modified the top layer of the model by specifying the number of output layers corresponding to the number of our classes (10 classes in case of tomato dataset , 38 classes in case of PlantVillage

dataset). we set a list of arguments that the Trainer will use to fit and evaluate our model. the training was done in 14 minutes(tomato dataset) and 1 hour(PlantVillage datatset) during 3 epochs using google colab pro 3.4.6.1

- We obtained from this experiment an accuracy of 98.9% and a loss of 0.04 on validation set with tomato dataset.

- We obtained from this experiment an accuracy of 99.69% and a loss of 0.01 on validation set with PlantVillage dataset.

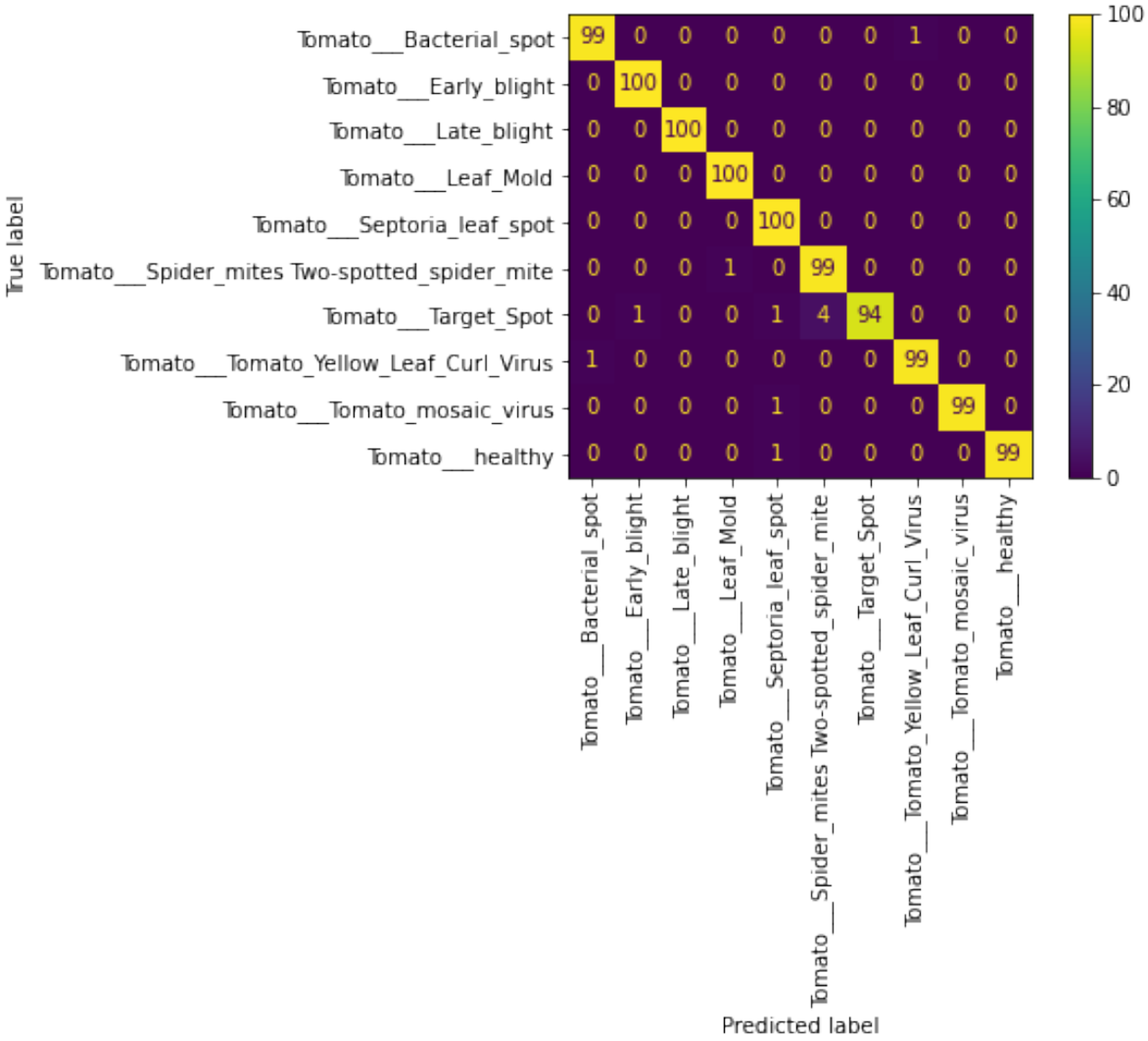

Figure 4.53: Confusion Matrix ViT BeiT patch 16-224 Tomato Dataset

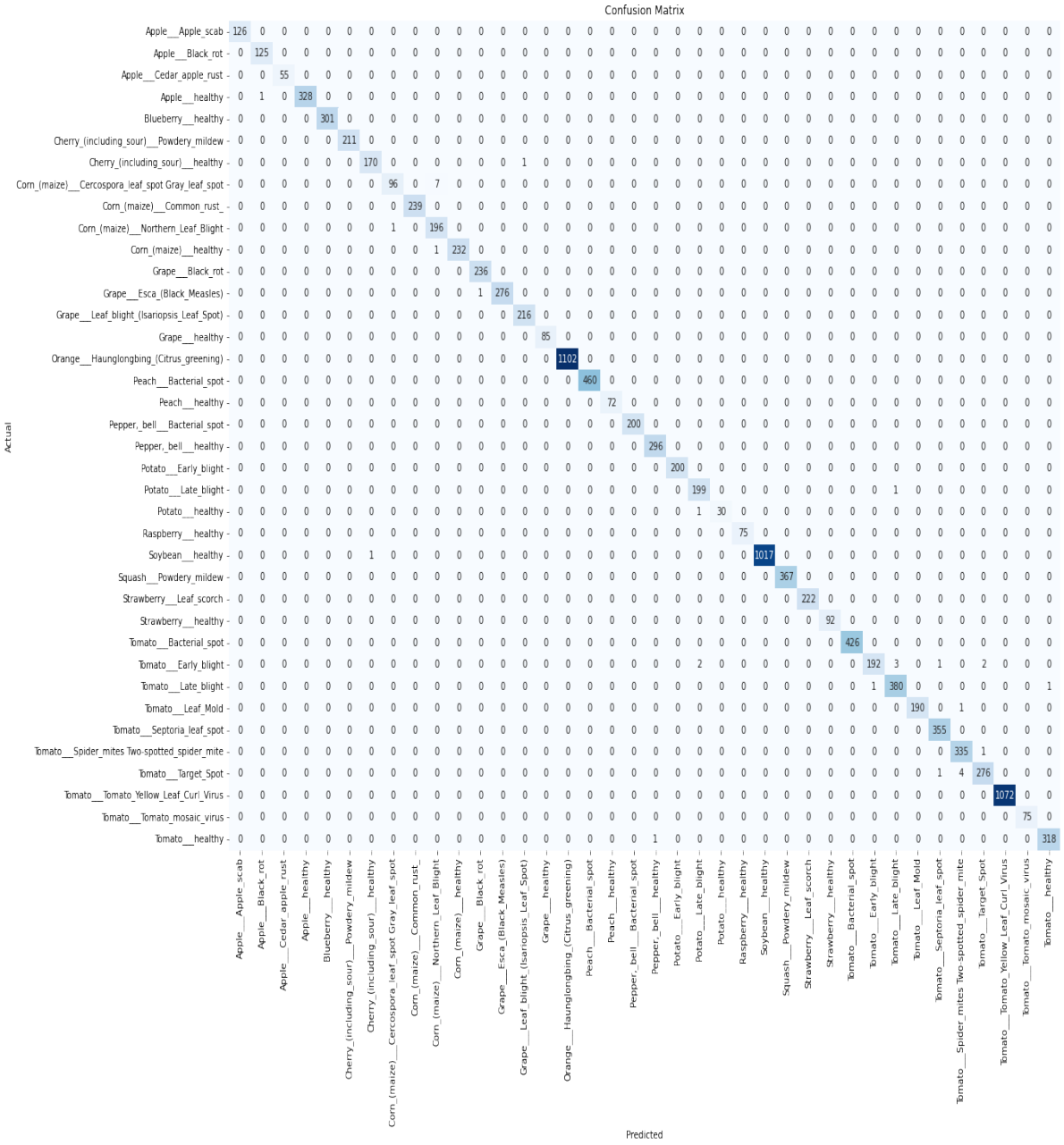

Figure 4.54: Confusion Matrix ViT BeiT patch 16-224 PlantVillage Dataset

4.4.4 ViT DeiT patch 16-224(Facebook) with Pytorch

In the begining we start by loading our dataset to [datasets type] using ImageFolder Dataset Builder, then we load the ViT Feature Extractor offered by the pre-trained model to normalize a dataset,we augment our data using random resize, random horizontal flip and center crop. we modified the top layer of the model by specifying the number of output layers corresponding to the number of our classes (10 classes in case of tomato dataset , 38 classes in case of PlantVillage

dataset). we set a list of arguments that the Trainer will use to fit and evaluate our model. the training was done in 14 minutes(tomato dataset) and 58 minutes(PlantVillage datatset) during 3 epochs using google colab pro 3.4.6.1

- We obtained from this experiment an accuracy of 99.2% and a loss of 0.02 on validation set with tomato dataset.

- We obtained from this experiment an accuracy of 99.73% and a loss of 0.01 on validation set with PlantVillage dataset.

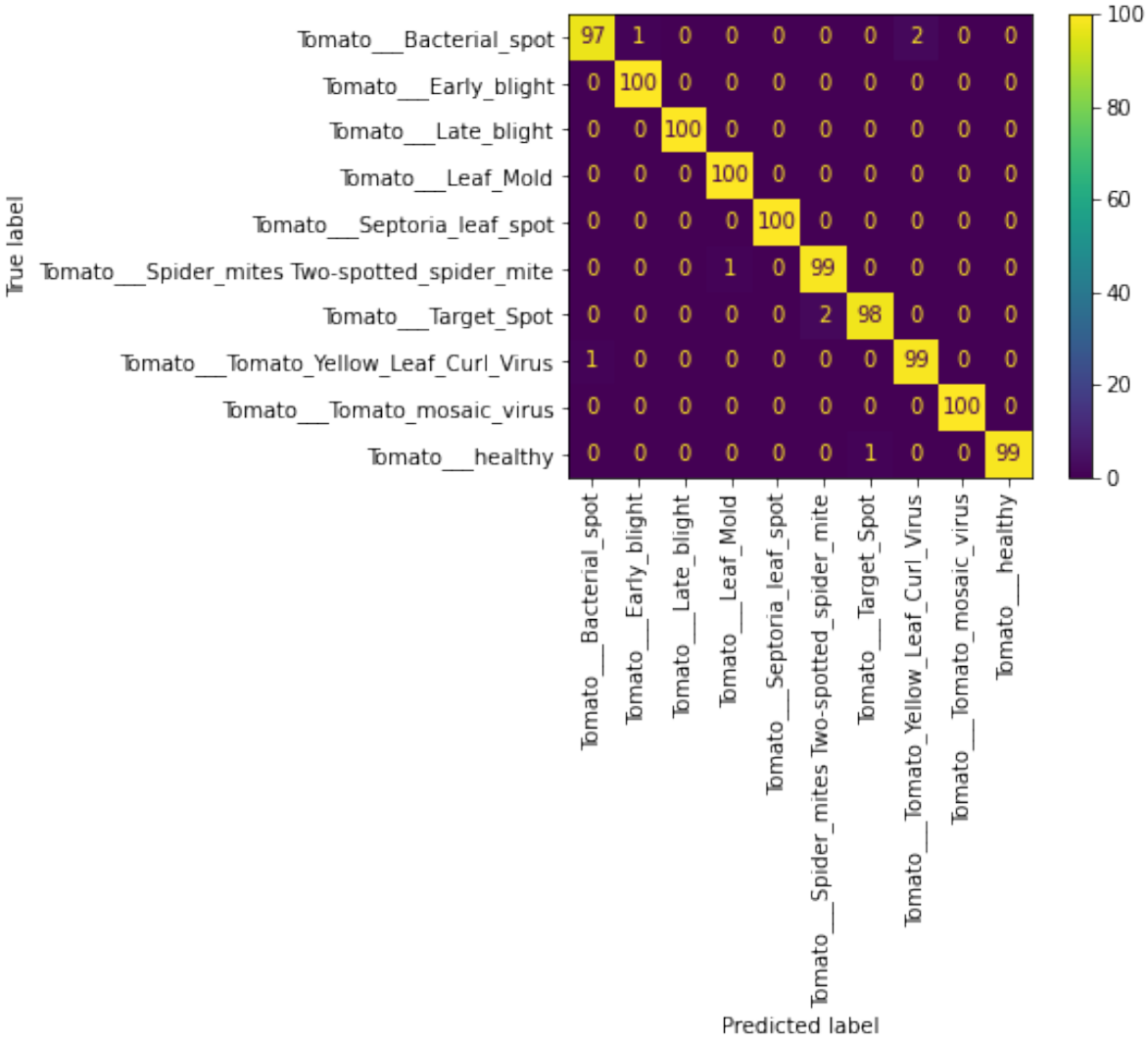

Figure 4.55: Confusion Matrix ViT DeiT patch 16-224 Tomato Dataset

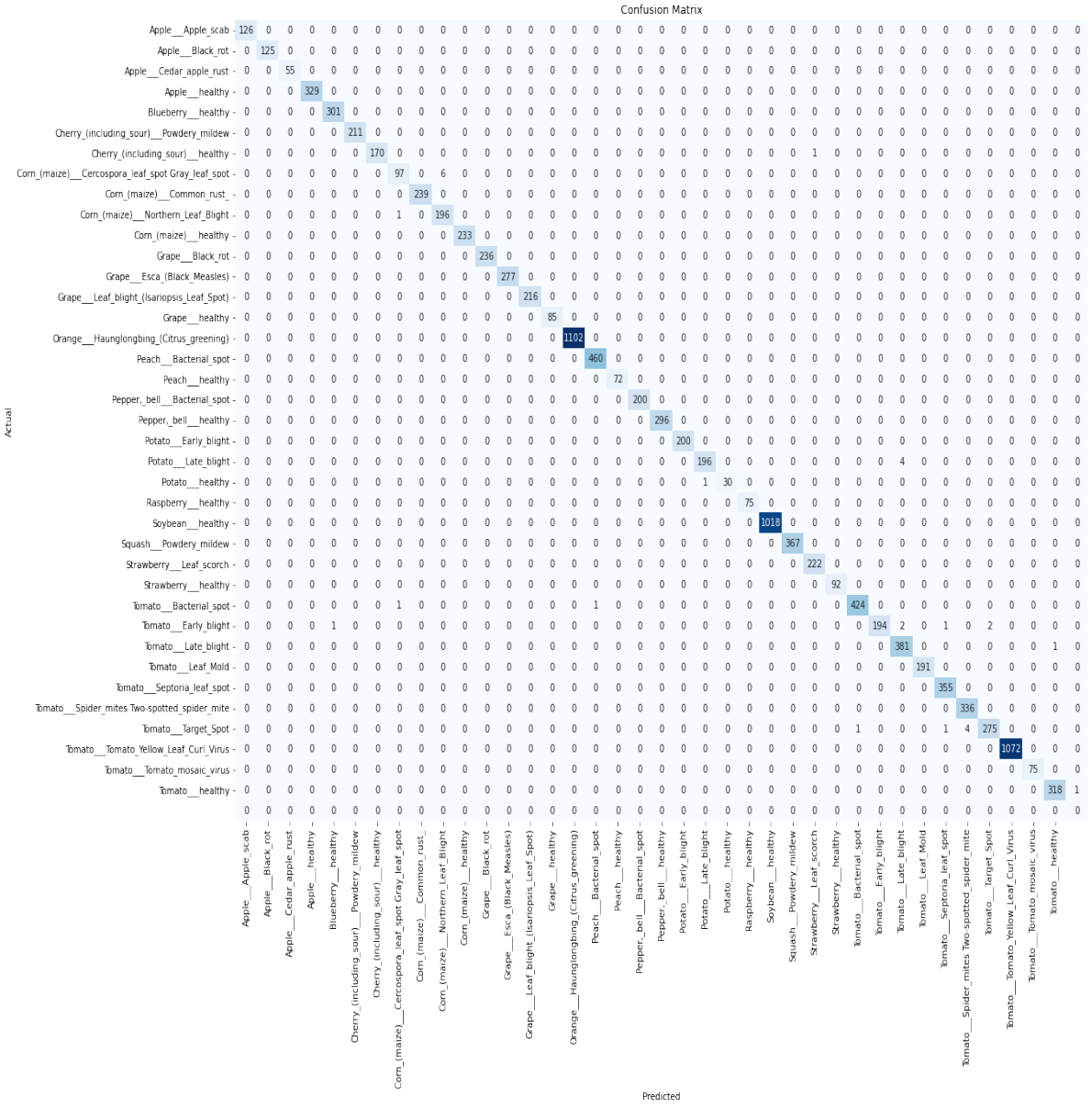

Figure 4.56: Confusion Matrix ViT DeiT patch 16-224 PlantVillage Dataset

4.5 Object Detecion using Yolo V5

In this section we will implement Yolo V5 to detect plant-leaf diseases. First we clone the yolo5 project from github, then we download the PlantDoc3.4.9.1 dataset from Robotflow website3.4.9, the dataset is already labeled with bounding boxes for each image to use in detection purpose, next we unfrozen the model then we train it within 40 minutes during 100 epochs using google colab pro 3.4.6.1

- We reached in this experiment a recall of 65.4%, a precision of 65.3% and mAP@.5 of 0.639 on validation set.

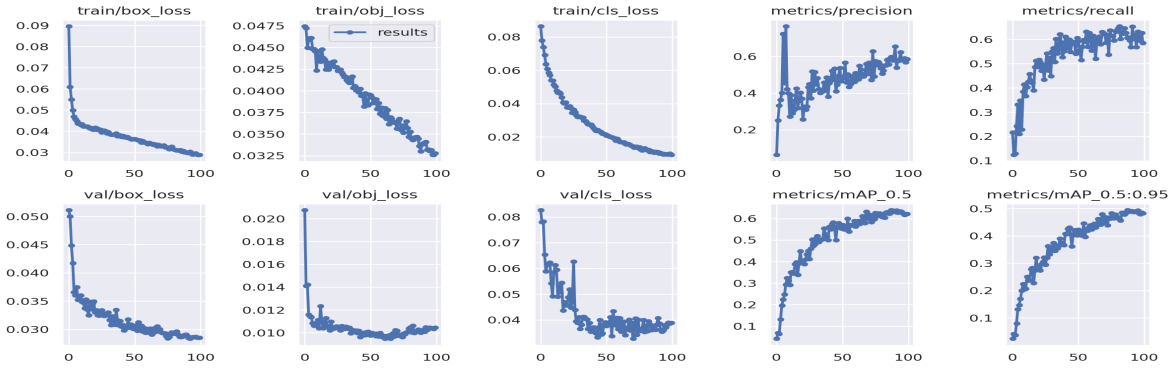

Figure 4.57: Train Results Metrics

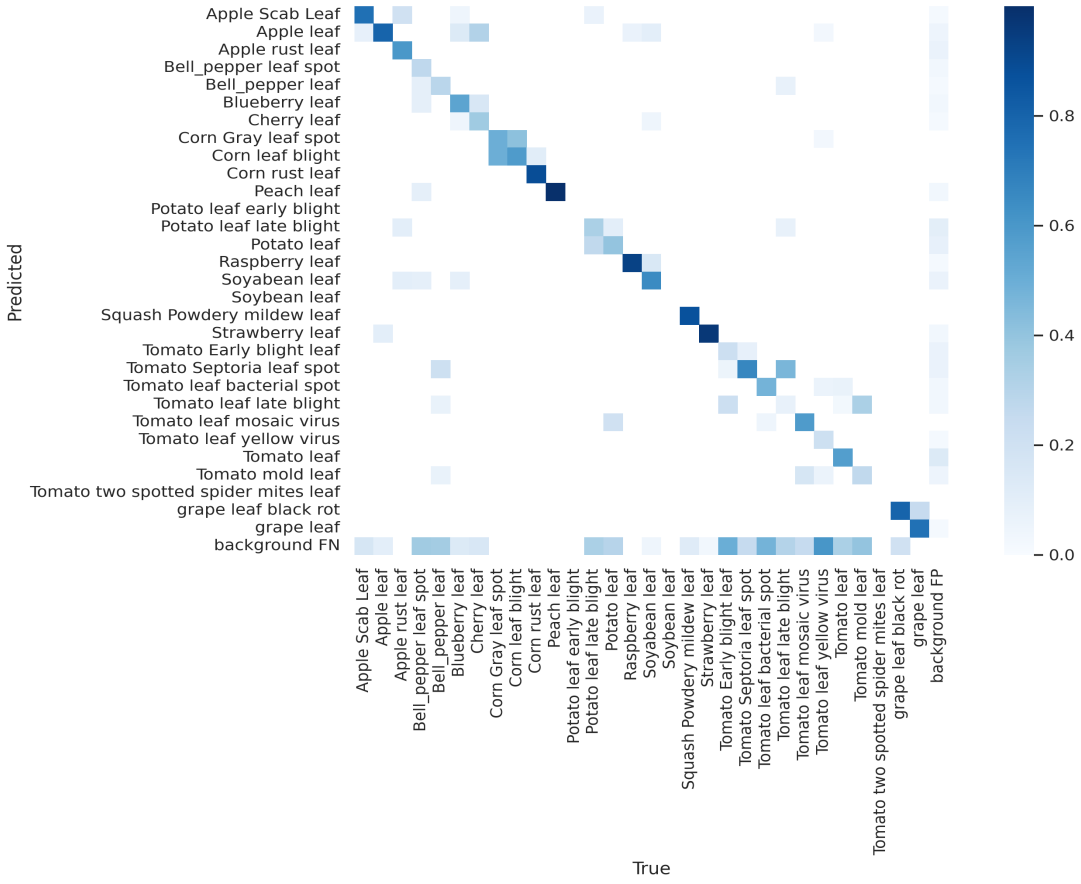

Figure 4.58: Confusion Matrix Yolo V5

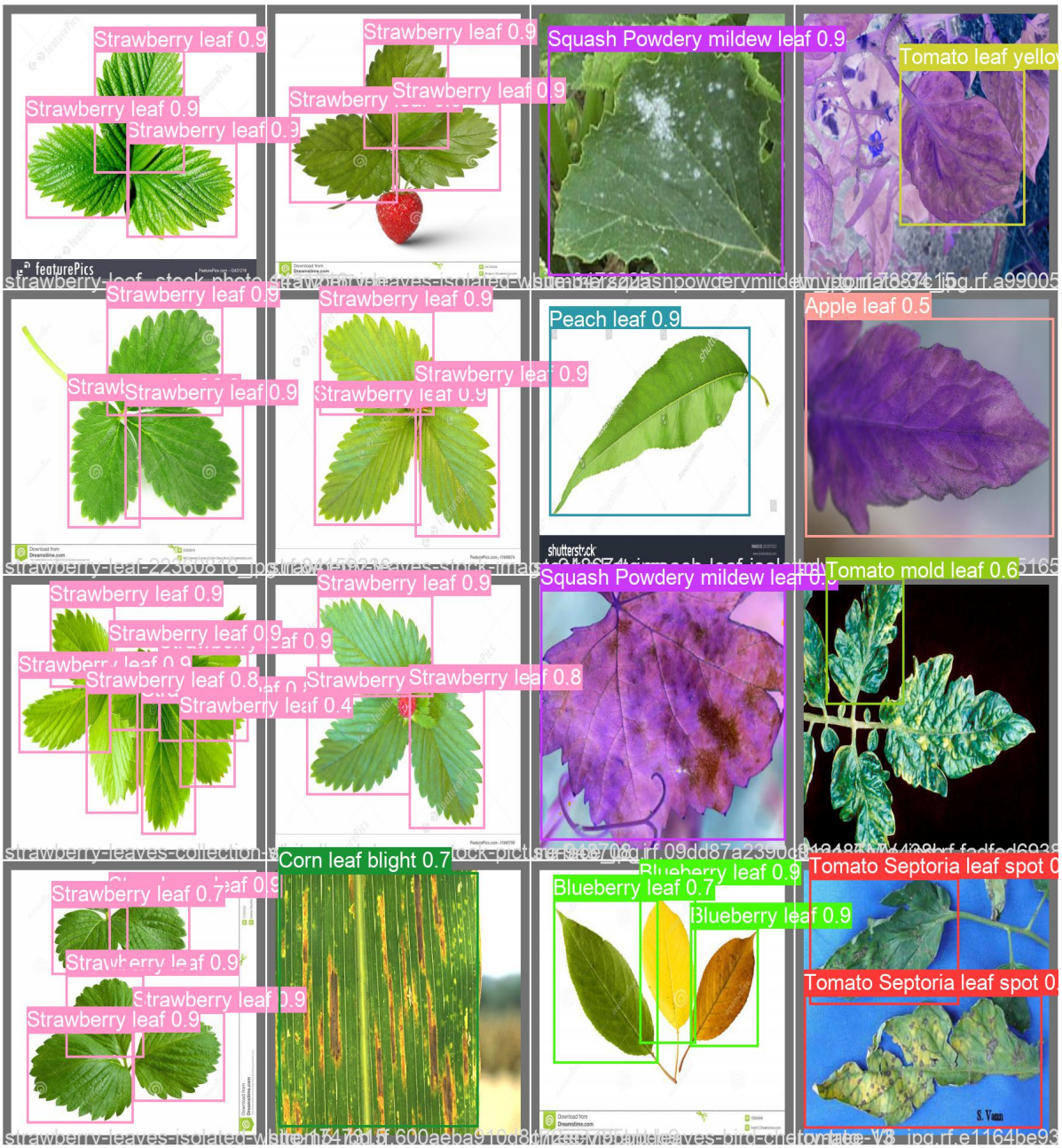

Figure 4.59: Yolo V5 Predictions

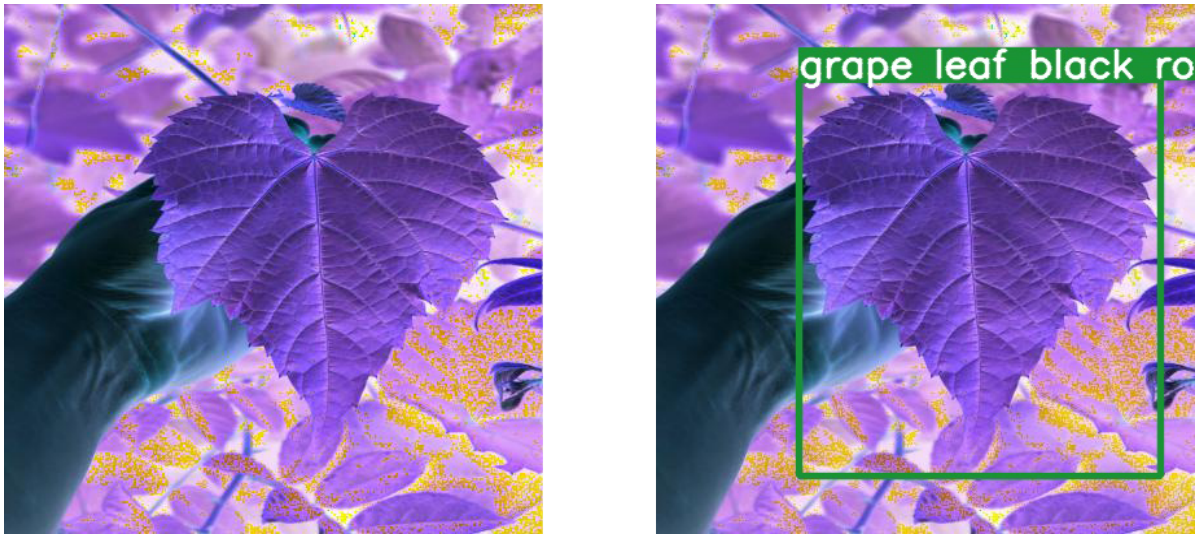

Figure 4.60: Yolo V5 Prediction Test

4.6 Results comparisons

In this section we will compare our different obtained results from our proposed algoritms and deep convolutions netwroks with Tomato dataset then PlantVillage dataset.

4.6.1 Summarized Models - Tomato Dataset

Algorithm	Validation Accuracy	Test Accuracy
CNN From scratch	87.9%	90.4%
CNN with Attention	90.7%	92.7%
VGG-16	97.5%	97.8%
VGG-19	94.7%	97.2%
InceptionV3	97.3%	98.7%
ResNet152V2	94%	97.2%
Xception	99.3%	99.8%
MobileNetV2	96.8%	97.9%
DenseNet201	99.5%	99.8%
EfficientNet B3	97.8%	97.3%
EfficientNet B5	97.5%	98.4%
ViT B32	95.9 %	97.7 %
ViT base (Google)	99.1	/
ViT BeiT (Microsoft)	98.9	/
ViT DeiT (Facebook)	99.2	/

Table 4.13: Summarized Models - Tomato Dataset

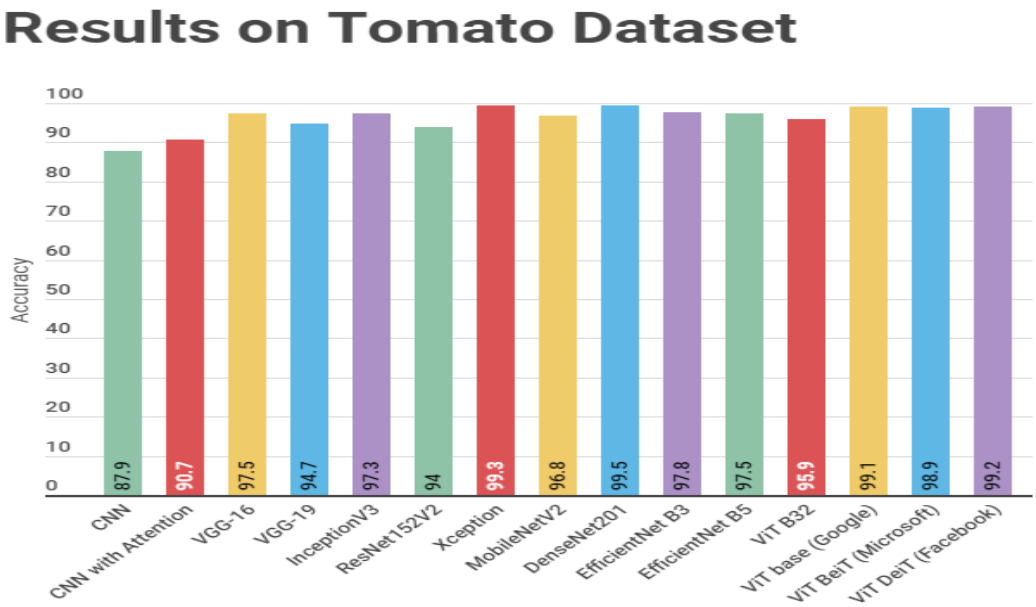

Figure 4.61: Results on Tomato Dataset

4.6.2 Summarized Models - PlantVillage Dataset

Algorithm	Validation Accuracy	Test Accuracy
CNN From scratch	96.57%	96.5%
CNN with Attention	97.74%	97.49%
VGG-16	99.13%	98.37%
VGG-19	97.64%	95.9%
InceptionV3	99.67%	99.31%
ResNet152V2	99.52%	99.45%
Xception	99.79%	99.52%
MobileNetV2	99.46%	99.01%
DenseNet201	99.86%	99.59%
EfficientNet B3	99.76%	99.65%
EfficientNet B5	99.77%	99.65%
ViT B32	97.94 %	86.44 %
ViT Base (Google)	99.7	/
ViT BeiT (Microsoft)	99.69	/
ViT DeiT (Facebook)	99.73	/

Table 4.14: Summarized Models - PlantVillage Dataset

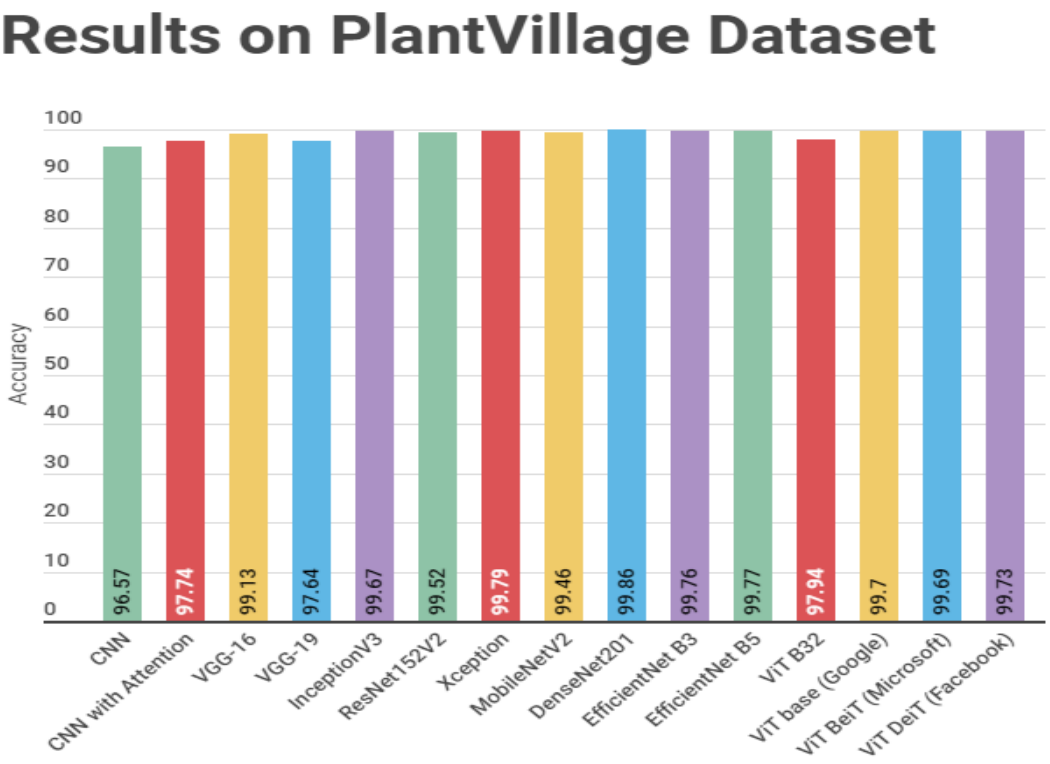

Figure 4.62: Results on PlantVillage Dataset

4.7 Summary

This chapter provided an overview of the results obtained from our proposed models.
The gradcam technique shows that the cnn based attention is determining the location of particular better than cnn do which gives better results.
The pretrained models made a good results and some models outperformed all other classifiers.
The ViT models gave a very interesting and good results so we can say that the transformers are very helpful in such cases.
Object detection was so accurate and precised for detecting the plant leaf disease and classify them with acceptable precision which can be better.

CONCLUSION AND FUTURE WORK

This work represents case study and aims to develop and propose a new novel classification model based on deep learning to automatically detect and classify plant leaf diseases better than the diagnosis of experts in pathology and also previous published papers.

For future , we would study further many related problems and test our models in different datasets to diagnose more plant leaf diseases.

In order to implement the result of our work, we are building a mobile app -**DeePlant**-, the application will use the recent technologies in detecting and classifying plant leaf disease such as Transformers, YoLo ViT model. And here is our proposed architecture and UI for the app.

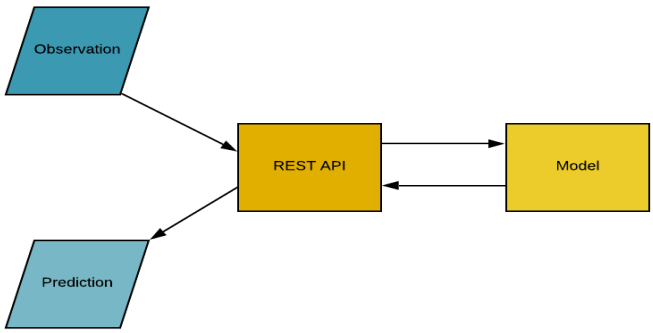

Figure 4.63: Back end Architecture for DeePlant App

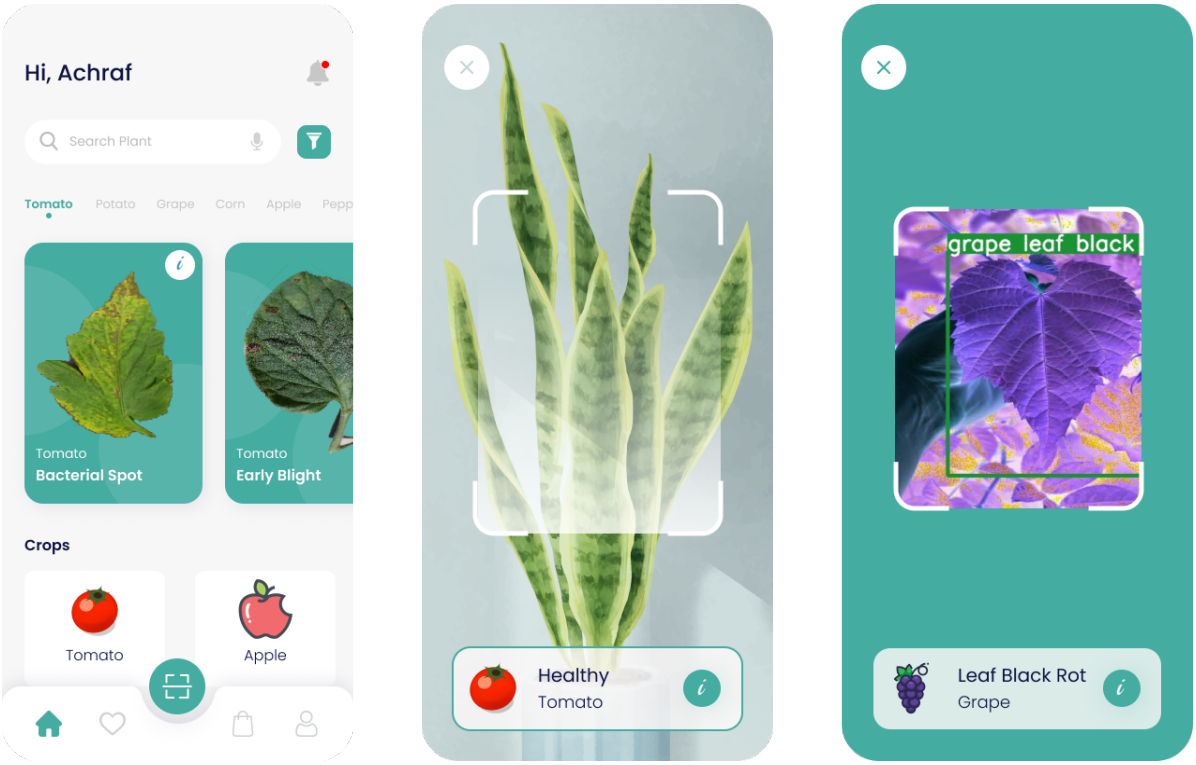

Figure 4.64: UI of DeePlant App

BIBLIOGRAPHY

[1] Leon T. Lucas George B. Lucas, C. Lee Campbell.
Introduction to Plant Diseases.
Springer New York, NY, 1992.

[2] Baleev D. N. Ivanova M. I. Sokolova L. M. Karakozova M. V Nazarov, P. A.
Infectious plant diseases: Etiology, current status, problems and prospects in plant protection.
Acta Naturae, 12, 2020.

[3] Rose C Gergerich and Valerian V Dolja.
Introduction to plant viruses, the invisible foe.
The plant health instructor, 478, 2006.

[4] H Czosnek and H Laterrot.
A worldwide survey of tomato yellow leaf curl viruses.
Archives of virology, 142(7):1391–1406, 1997.

[5] Arthur Samuel.
Some studies in machine learning using the game of checkers. ii—recent progress.
IBM Journal of Research and Development, 3:206 – 226, 02 2000.

[6] E. Alpaydin.
Introduction to Machine Learning, second edition.
Adaptive Computation and Machine Learning series. MIT Press, 2009.

[7] J. Han, J. Pei, and M. Kamber.
Data Mining: Concepts and Techniques.
The Morgan Kaufmann Series in Data Management Systems. Elsevier Science, 2011.

[8] `https://www.coursera.org/learn/machinelearning`.
[Accessed 26-Jun-2022].

[9] Sotiris B Kotsiantis, Ioannis Zaharakis, P Pintelas, et al.
Supervised machine learning: A review of classification techniques.
Emerging artificial intelligence applications in computer engineering, 160(1):3–24, 2007.

[10] Vladimir Vapnik.
Statistical learning theory. john wiley&sons.
Inc., New York, 1, 1998.

[11] J. Han, J. Pei, and M. Kamber.
Data Mining, Southeast Asia Edition.
The Morgan Kaufmann Series in Data Management Systems. Elsevier Science, 2006.

[12] Pranjali B Padol and Anjali A Yadav.
Svm classifier based grape leaf disease detection.
In *2016 Conference on advances in signal processing (CASP)*, pages 175–179. IEEE, 2016.

[13] Sai Arivazhagan, R Newlin Shebiah, S Ananthi, and S Vishnu Varthini.
Detection of unhealthy region of plant leaves and classification of plant leaf diseases using texture features.
Agricultural Engineering International: CIGR Journal, 15(1):211–217, 2013.

[14] Shima Ramesh, Ramachandra Hebbar, M Niveditha, R Pooja, N Shashank, PV Vinod, et al.
Plant disease detection using machine learning.
In *2018 International conference on design innovations for 3Cs compute communicate control (ICDI3C)*, pages 41–45. IEEE, 2018.

[15] Ms Deepika Chauhan et al.
Detection of maize disease using random forest classification algorithm.
Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12(9):715–720, 2021.

[16] AL Mila, AL Carriquiry, and XB Yang.
Logistic regression modeling of prevalence of soybean sclerotinia stem rot in the north-central region of the united states.
Phytopathology, 94(1):102–110, 2004.

[17] Divyansh Tiwari, Mritunjay Ashish, Nitish Gangwar, Abhishek Sharma, Suhanshu Patel, and Suyash Bhardwaj.
Potato leaf diseases detection using deep learning.
In *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 461–466. IEEE, 2020.

[18] Amanda Ramcharan, Kelsee Baranowski, Peter McCloskey, Babuali Ahmed, James Legg, and David P Hughes.
Deep learning for image-based cassava disease detection.
Frontiers in plant science, 8:1852, 2017.

[19] MP Vaishnnave, K Suganya Devi, P Srinivasan, and G Arut Perum Jothi.
Detection and classification of groundnut leaf diseases using knn classifier.
In *2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN)*, pages 1–5. IEEE, 2019.

[20] B Rajesh, M Vishnu Sai Vardhan, and L Sujihelen.
Leaf disease detection and classification by decision tree.
In *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184)*, pages 705–708. IEEE, 2020.

[21] Kawcher Ahmed, Tasmia Rahman Shahidi, Syed Md Irfanul Alam, and Sifat Momen.
Rice leaf disease detection using machine learning techniques.
In *2019 International Conference on Sustainable Technologies for Industry 4.0 (STI)*, pages 1–5. IEEE, 2019.

[22] Irina Rish et al.
An empirical study of the naive bayes classifier.
In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

[23] Kshyanaprava Panda Panigrahi, Himansu Das, Abhaya Kumar Sahoo, and Suresh Chandra Moharana.
Maize leaf disease detection and classification using machine learning algorithms.
In *Progress in Computing, Analytics and Networking*, pages 659–669. Springer, 2020.

[24] Tom M Mitchell et al.
Machine learning, 1997.

[25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville.
Deep feedforward networks.
Deep learning, (1), 2016.

[26] David E Rumelhart, Richard Durbin, Richard Golden, and Yves Chauvin.
Backpropagation: The basic theory.
Backpropagation: Theory, architectures and applications, pages 1–34, 1995.

[27] Geoffrey Hinton.
Neural networks for machine learning.
University of Toronto via Coursera https://www.classcentral.com/course/coursera-neural-networks-formachine- learning-398.

[28] Diederik P Kingma and Jimmy Ba.
Adam: A method for stochastic optimization.

arXiv preprint arXiv:1412.6980, 2014.

[29] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov.
Dropout: a simple way to prevent neural networks from overfitting.
The journal of machine learning research, 15(1):1929–1958, 2014.

[30] Andrew NG.
Machine learning course in coursera.
https://www.coursera.org/learn/machinelearning.

[31] Sergey Ioffe and Christian Szegedy.
Batch normalization: Accelerating deep network training by reducing internal covariate shift.
In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[32] Lutz Prechelt.
Early stopping-but when?
In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 1998.

[33] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner.
Gradient-based learning applied to document recognition.
Proceedings of the IEEE, 86(11):2278–2324, 1998.

[34] Dominik Scherer, Andreas Müller, and Sven Behnke.
Evaluation of pooling operations in convolutional architectures for object recognition.
In *International conference on artificial neural networks*, pages 92–101. Springer, 2010.

[35] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon.
CBAM: convolutional block attention module.
CoRR, abs/1807.06521, 2018.

[36] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra.
Grad-cam: Visual explanations from deep networks via gradient-based localization.
In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[37] Jia Deng.
W., dong, r. socher, l.
In *J., Li„ K., Li„ and, L., Fei, Fei., Imagenet:, A large-scale, hierarchical, image, database., In, IEEE, Conference, on Computer, Vision, and Pattern, Recogni, tion,(CVPR)„ pages*, pages 248–255, 2009.

[38] Karen Simonyan and Andrew Zisserman.
Very deep convolutional networks for large-scale image recognition.
arXiv preprint arXiv:1409.1556, 2014.

[39] Muhammad Mateen, Junhao Wen, Sun Song, and Zhouping Huang.
Fundus image classification using vgg-19 architecture with pca and svd.
Symmetry, 11(1):1, 2018.

[40] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al.
Imagenet large scale visual recognition challenge.
International journal of computer vision, 115(3):211–252, 2015.

[41] François Chollet.
Xception: Deep learning with depthwise separable convolutions.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[42] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun.
Deep residual learning for image recognition.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[43] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen.
Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation.
CoRR, abs/1801.04381, 2018.

[44] Mingxing Tan and Quoc V. Le.
Efficientnet: Rethinking model scaling for convolutional neural networks.
CoRR, abs/1905.11946, 2019.

[45] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger.
Densely connected convolutional networks.
CoRR, abs/1608.06993, 2016.

[46] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby.
An image is worth 16x16 words: Transformers for image recognition at scale.
CoRR, abs/2010.11929, 2020.

[47] Vision Transformers (ViT) in Image Recognition - 2022 Guide - viso.ai — viso.ai.
https://viso.ai/deep-learning/vision-transformer-vit/.
[Accessed 23-Jun-2022].

[48] Melike Sardogan, Adem Tuncer, and Yunus Ozen.
Plant leaf disease detection and classification based on cnn with lvq algorithm.
In *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, pages 382–385. IEEE, 2018.

[49] Sharada P Mohanty, David P Hughes, and Marcel Salathé.
Using deep learning for image-based plant disease detection.
Frontiers in plant science, 7:1419, 2016.

[50] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton.
Imagenet classification with deep convolutional neural networks.
Advances in neural information processing systems, 25, 2012.

[51] Junde Chen, Jinxiu Chen, Defu Zhang, Yuandong Sun, and Yaser Ahangari Nanehkaran.
Using deep transfer learning for image-based plant disease identification.
Computers and Electronics in Agriculture, 173:105393, 2020.

[52] Bin Liu, Yun Zhang, DongJian He, and Yuxiang Li.
Identification of apple leaf diseases based on deep convolutional neural networks.
Symmetry, 10(1):11, 2017.

[53] Mohammed Brahimi, Kamel Boukhalfa, and Abdelouahab Moussaoui.
Deep learning for tomato diseases: classification and symptoms visualization.
Applied Artificial Intelligence, 31(4):299–315, 2017.

[54] Amit Yali, Felzenszwalb Pedro, and Girshick Ross.
Object Detection, pages 1–9.
Springer International Publishing, Cham, 2020.

[55] YOLOv5 Documentation — docs.ultralytics.com.
https://docs.ultralytics.com/.
[Accessed 26-Jun-2022].

[56] GitHub - ultralytics/yolov5: YOLOv5 in PyTorch > ONNX > CoreML > TFLite — github.com.
https://github.com/ultralytics/yolov5.
[Accessed 26-Jun-2022].

[57] Tomato leaf disease detection — kaggle.com.
https://www.kaggle.com/datasets/kaustubhb999/tomatoleaf?fbclid=IwAR1W8JQgkE1Gdi_kj_hzY7MQaigdmqFP5L-WkFGQMR9PcUuZYuPzi5bC4e4.

[Accessed 21-Jun-2022].

[58] Futura.
Futura, Explorer le monde — futura-sciences.com.
https://www.futura-sciences.com/.
[Accessed 08-Jun-2022].

[59] TensorFlow — tensorflow.org.
https://www.tensorflow.org/.
[Accessed 11-Jun-2022].

[60] Keras Team.
Keras documentation: About Keras — keras.io.
https://keras.io/about/.
[Accessed 11-Jun-2022].

[61] deepAI - x4EBA;x5DE5;x667A;x80FD;x3001;x533A;x5757;x94FE;x3001;x673A;x5668;x4EBA;
— deepAi.com.
www.deepAi.com.
[Accessed 26-Jun-2022].

[62] Matplotlib x2014; Visualization with Python — matplotlib.org.
https://matplotlib.org/.
[Accessed 11-Jun-2022].

[63] Google Colab Tutorial — tutorialspoint.com.
https://www.tutorialspoint.com/google_colab/.
[Accessed 12-Jun-2022].

[64] Ishan Chawla.
Google Colab vs Google Colab Pro on Multidimensional TFT Architectures — Ishankc.
https://medium.com/@Ishankc/google-colab-vs-google-colab-pro-on-multidimensional-tft-archi
[Accessed 21-Jun-2022].

[65] Hugging Face – The AI community building the future. — huggingface.co.
https://huggingface.co/.
[Accessed 24-Jun-2022].

[66] Roboflow: Give your software the power to see objects in images and video — roboflow.com.
https://roboflow.com/.
[Accessed 24-Jun-2022].

[67] http://www.scikitlearn.org.
[Accessed 21-Jun-2022].

[68] https://www.google.com/MachineLearningMastery/.
What is a Confusion Matrix in Machine Learning - Machine Learning Mastery — machinelearningmastery.com.
https://machinelearningmastery.com/confusion-matrix-machine-learning/.
[Accessed 21-Jun-2022].

[69] Waseem Rawat and Zenghui Wang.
Deep convolutional neural networks for image classification: A comprehensive review.
Neural computation, 29(9):2352–2449, 2017.