CSE1901 - Technical Answers to Real World Problems (TARP)

Project Report

DISEASE DETECTION IN PLANTS

By

19BCE1680	Bipasa Mukherjee
19BCE1532	Hima Rani Mathews
19BCE1778	Talatala Rahul Reddy
19BCE1298	Manav Rakesh Kumar Modi
19BCE1616	Achintya Kumar Satyayan

B. Tech Computer Science and Engineering

Submitted to

S.A. Sajidha

School of Computer Science and Engineering

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

April 2022

DECLARATION BY THE TEAM

We hereby declare that the report titled " **Disease Detection In Plants**" submitted by our team to VIT Chennai is a record of bonafide work undertaken by our team under the supervision of Dr., Assistant Professor Senior, SCOPE, Vellore Institute of Technology, Chennai.

Signature of the Candidate

Bipasa Mukherjee (19BCE1680)

Hima Rani Mathews (19BCE1532)

Talatala Rahul Reddy (19BCE1778)

Manav Rakesh Kumar Modi (19BCE1298)

Achintya Kumar Satyayan (19BCE1616)

CERTIFICATE

Certified that this project report entitled “**Disease Detection In Plants**” is a bonafide work of **Bipasa Mukherjee (19BCE1680), Hima Rani Mathews (19BCE1532), Talatala Rahul Reddy (19BCE1778), Manav Rakesh Kumar Modi (19BCE1298) and Achintya Kumar Satyayan (19BCE1616)** and they carried out the project work under my supervision and guidance for CSE1901 - Technical Answers to Real-World Problems (TARP).

S.A. Sajidha

SCOPE, VIT Chennai

ACKNOWLEDGEMENT

We wish to express our sincere thanks and a deep sense of gratitude to our project guide, Disease Detection In Plants, the School of Computer Science and Engineering for her consistent encouragement and valuable guidance offered to us throughout the project work.

We are extremely grateful to Dr. Ganesan R, the Dean, School of Computer Science and Engineering (SCOPE), Vellore Institute of Technology, Chennai, for extending the facilities of the School towards our project and for his unstinting support.

We express our thanks to our Head of the Department for his support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and the wisdom imparted to us throughout the courses.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

ABSTRACT

Agriculture is one of the most important parts of our Indian economy and to get the maximum yield out of it farmers need to take good care of crops, that is why it is necessary to have an early disease detection system in place. Plant disease is the breakdown of a plant's normal state, which disrupts and changes its functionality. Such disorders are primarily caused by pathogens. Tomato being one of the a rich source of minerals, vitamins and organic acid, essential amino acids and dietary fibers in our diet are being frequently infected by a dangerous fungal disease called early blight resulting in 100% production loss to farmers. In the modern technological era, several procedures for detecting plant diseases have been developed for agricultural applications. Earlier approaches included the usage of a simple standalone Convolution Neural Network (CNN) algorithm, the model was able to produce a decent accuracy of 93%. We suggest in our paper to investigate the tomato plant disease using the Plant Village dataset which consists of 23,000 images for 9 diseased classes. To begin, the input photographs are preprocessed, and also the target image region is split from the originals. Second, the images have been further processed by incorporating a deep-learning framework and transfer learning where VGG-19 is used for feature extraction. Later these features are utilized to detect and classify the type of tomato plant disease. The results illustrate that the proposed model's predictions are 96% accurate in 9 epochs.

CONTENTS

Declaration i

Certificate ii

Acknowledgement iii

Abstract iv

1 Introduction 1

1.1 Objective and goal of the project: . . . .…….1

1.2 Problem Statement. …. 1

1.3 Motivation . . . . …………...2

1.4 Challenges . . . . …………....2

2 Literature Survey 3

3 Requirements Specification 10

3.1 Hardware Requirements . . . . … 10

3.2 Software Requirements 10

4 System Design 11

5 Implementation of System 14

6 Results & Discussion 17

7 Conclusion and Future Work 19

8 References 20

9 Appendix 21

1. Introduction

1.1 Objective and Goal of the project

This paper presents an overview of how the employment of a deep learning framework and transfer learning techniques in various systems might improve disease detection accuracy in tomato plants. Bacterial Spot, Early Blight, Late Blight, Leaf Mold, Septoria Leaf spot, two-spotted spider mite, Target Spot, Mosaic Virus & Yellow Leaf Curl Virus are a few of the dominant diseases that deteriorate the tomato plants. These irregular or dissimilarities on the leaves are identified using object-recognition models like VGG-19 which is an architecture used in CNN for detecting and classifying diseases in tomato plants, as the name suggests VGG-19 has 19 layers that are deep trained on the ImageNet to get the features extracted from the image. The images are processed in the hidden layers which contain various image processing layers like pooling, padding, convolutional layers, etc., for training the model. The trained model is used to detect tomato plant disease using a VGG-19 model architecture used in CNN and also recommends the appropriate remedy for the same.

1.2 Problem Statement

Agriculture's evolution thousands of years ago resulted in the domestication of today's key food crops and animals. Food insecurity, among which plant diseases are indeed a key cause, is one of the greatest worldwide concerns that humanity faces today. India is a country where more than 70 percent of people are directly dependent on farming and hence economic growth and development of farmers in a country like India largely depend upon the yields that they grow. Tomatoes have become an essential vegetable crop around the globe, contributing to around 15% of total vegetable consumption and the per capita consumption of almost 20 kg per year.

Fresh tomato production in the world surpasses 170 million tonnes per year, placing it number 1 among vegetable crop production. Tomato disease is the leading cause of global tomato production decline, according to United Nations Food And agriculture survey statistics, with an annual deficit rate of nearly 8%–10%. Most tomato infections, on the other hand, begin in the leaves and spread throughout the plant.

1.3 Motivation

Automatically identifying tomato leaf diseases and providing a suitable growth environment can assist to enhance tomato production management and give a good development environment. Several research works have been conducted to boost the survival rate of tomatoes using an early diagnosis of diseases and subsequent disease control using deep learning technology. El Hassouni et al. suggested a CNN model for classifying ten of the most frequent tomato plant diseases. They obtained a 90.3% overall accuracy using an adaption of the MobileNet model [1]. To create features of tomato plant illnesses, Alvaro Fuentes et al [2] employed image processing and spectroscopic approaches on numerous pictures. On several images of diseased plant leaves, Rajvinder Kaur et al [3] have used a Support Vector Machine classifier, K-Nearest Neighbor (KNN), Neural Network, and obtained an accuracy of 91%. A neural network that used transfer learning as that of an AlexNet-based deep learning mechanism to assess and categorize tomato plant leaf health reached an accuracy of 95.75% [4]. The overall recognition rate of 10 types of tomato leaves is improved when Wang et al. [5] implement transfer learning to the native Alex Net network. Rangarajan et al. [6] integrated the core AlexNet, VGG16 framework with transfer learning to achieve a 97 percent accuracy on the 7 segmented tomato diseased leaves. The impacts of weight, variance and learning rate on disease diagnosis accuracy and speed are investigated. As a result, we would like to build a CNN model based on the VGG-19 architecture that would aid us in identifying and classifying diseases on unhealthy tomato plants.

1.4 Challenges

Due to many aspects such as segmentation sensitivity toward the ROI (Region of Interest), dynamic environment with diverse real-life conditions, identification of multiple diseases in a single leaf, and so on, configuring the VGG-19 architecture on CNN model was not delivering the desired results. Plant diseases and pests identification in a genuine complex natural environment has a number of obstacles, including a tiny difference between the lesion region and the backdrop, low contrast, substantial fluctuations in the extent of the lesion area and various kinds, and a lot of noise mostly in lesion image. There are also a lot of distractions when gathering photographs of plant diseases and pests in natural light. Traditional classical approaches typically appear ineffective at this period, and better detection results remain difficult to achieve.

2. Literature Survey

Numerous studies have focused on this topic, particularly early work is based on the use of image processing and machine learning for disease detection and classification. The problem is that these classifiers lack automation because they use hand-crafted features designed by experts to extract features relevant to image classification. To overcome this limit, in recent years, many researchers adopted deep learning, able to use the acquired images directly, without the aid of craft functions. Below table shows various contributions of research in recent years and their outcomes.

Serial No	**Author**	**Dataset**	**Description**	**Methodology**	**Result**
1	Rajasekaran, C., et al, IEEE, 2020	Plant Village Dataset	Turmeric Plant Diseases Detection and Classification using Artificial Intelligence.	Compared VGG-16 (Visual Geometry Group) architecture in CNN and AlexNet architecture for diseased image classification and detection using the Raspberry Pi hardware module.	The overall accuracy obtained by VGG-16 was 0.9624 and loss was 0.8719 which was better accuracy value as compared to Alex-net where it had an overall accuracy of 0.0525 and loss of 40.4076.

2	KP, A., & Anitha, J. (2021)	Plant Village Dataset	Plant disease classification using deep learning	The CNN model and pre-trained models such as VGG, ResNet, and DenseNet models are trained using the dataset.	The DenseNet model achieved the highest accuracy of 98.27%.
3	Li, X., & Rai, L. (2020)	Plant Village Dataset	Apple Leaf Disease Identification and Classification using ResNet Models.	Image segmentation SVM classifier and ResNet and VGG convolutional neural network model were used for comparison and improvement where the grayscale symbiosis matrix (GLCM) was used to extract statistical texture features, contrast, correlation, energy and uniformity, mean, standard deviation, entropy, skew and energy, a total of 10 features.	The ResNet-18 and ResNet-34 networks train and classify the original image and the image after image segmentation and extraction, with an accuracy of 99% and 97%.

4	Muammer Türkoğlu, Davut Hanbay	Images of plant diseases common to the Malatya, Bingöl, and Elazığ regions of Turkey obtained with a Nikon 7200d camera. Each image in this dataset consists of 4000 × 6000 resolution and three-channel (RGB) color images	Plant disease and pest detection using deep learning-based features	Transfer learning and deep feature extraction methods are used and the pretrained deep models are considered in the presented work for feature extraction and for further fine-tuning. The obtained features using deep feature extraction are then classified by support vector machine (SVM), extreme learning machine (ELM), and K-nearest neighbor (KNN) methods.	The fc6 layers of the AlexNet, VGG16, and VGG19 models produced better accuracy scores when compared to the other layers.

5	R. Sujatha , Jyotir Moy Chatterjee , NZ Jhanjhi, Sarfraz Nawaz Brohi	Gathered manually with the guidance of experts and citrus research center, Government of Punjab, Pakistan located in Sargodha city.	Performance of deep learning vs machine learning in plant leaf disease detection	Comparing the performance of ML (Support Vector Machine (SVM), Random Forest (RF), Stochastic Gradient Descent (SGD)) & DL (Inception-v3, VGG-16, VGG-19) in terms of citrus plant disease detection	DL methods perform better than that of ML methods in case of disease detection as follows: RF-76.8% > SGD-86.5% > SVM87% > VGG-19–87.4% > Inception-v3–89% > VGG-16–89.5%. From the result, it is seen RF is giving the least CA whereas VGG-16 is giving the best in terms of CA.

6	Krishnaswamy Rangarajan, A., Purushothaman, R.	A dataset for these diseases have been created with the images of isolated leaf samples using different smartphone cameras in laboratory conditions.	Disease Classification in Eggplant Using Pre-trained VGG16 and MSVM.	In this study, five major diseases due to pests and pathogens have been identified. Pre-trained Visual Geometry Group 16 (VGG16) architecture has been used and the images have been converted to other color spaces namely Hue Saturation Value (HSV), YCbCr and grayscale for evaluation. VGG16 has been used as feature extractor from 8th convolution layer and these features have been used for classifying diseases employing Multi-Class Support Vector Machine (MSVM).	The dataset created with RGB and YCbCr images in feld condition was promising with a classifcation accuracy of 99.4%.

7	V. K. Shrivastava, M. K. Pradhan and M. P. Thakur	The data set was created with images captured by two smartphone mobiles (Gionee and LYF mobile phones having 5.0 megapixel) and a digital camera (Canon PowerShot SX530HS with 16.0 megapixel)	Application of Pre-Trained Deep Convolutional Neural Networks for Rice Plant Disease Classification	Explored the performance of various pre-trained deep CNN models such as: (i) AlexNet; (ii) Vgg16; (iii) ResNet152V2; (iv) InceptionV3; (V) InceptionResNetV2; (vi) Xception; (vii) MobileNet; (viii) DenseNet169; (ix) NasNetMobile; and (x) NasNetLarge for image based rice plant disease classification	The Vgg16 model resulted in the highest classification accuracy of 93.11%.

8	Srivastava, S., Kumar, P., Mohd, N., Singh, A., & Gill, F. S. (2020).	The dataset was formed by collecting images of diseased and non-diseased sugarcane from Mawana Sugar Mill Pvt. Ltd.	A Novel Deep Learning Framework Approach for Sugarcane Disease Detection	The study comprises three scenarios based on different feature extractors namely Inception v3, VGG-16, and VGG-19. The state-of-the-art algorithms (SVM, SGD, ANN, naive Bayes, KNN, and logistic regression) are compared with deep learning algorithms like neural networks and hybrid AdaBoost. Several statistical measures such as accuracy, precision, specificity, AUC, and sensitivity are calculated using Orange software	An AUC of 90.2% is obtained using VGG-16 as the feature extractor and SVM as the classifier.

3. Requirements Specification

3.1 Hardware Requirements

- Processor : 2.5 gigahertz (GHz) frequency or above
- RAM : A minimum of 4 GB of RAM
- Hard disk : A minimum of 20 GB of available space
- Monitor : Minimum Resolution 1024 X 768
- System Type : 64-bit operating system, x64-based processor

3.2 Software Requirements

- Operating System : Windows 7 and above.
- Programming language : Python 2.7 and above.
- Platform : Jupyter Notebook
- Supporting libraries: Tensorflow, NumPy, pandas, etc.

4. System Design

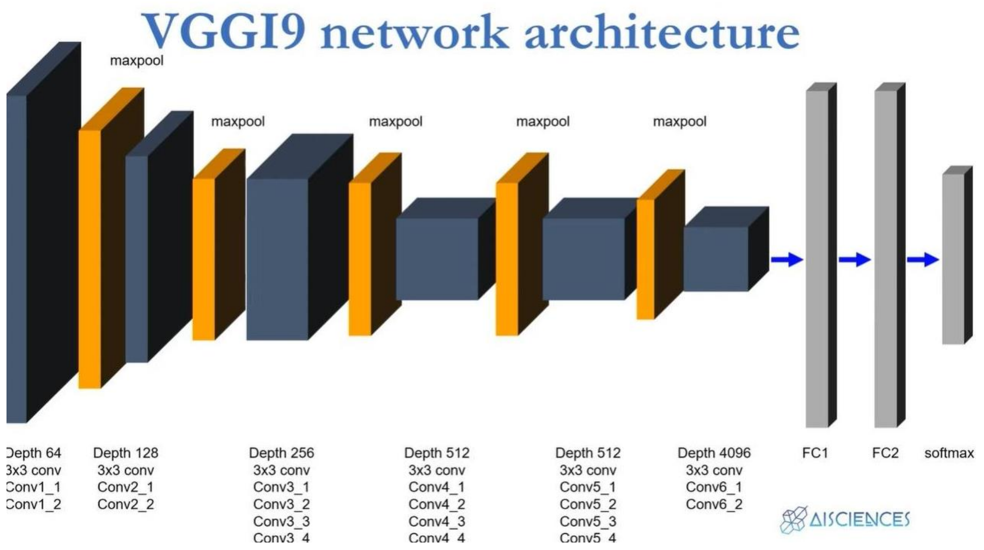

VGGI9 network architecture
maxpool
maxpool
maxpool
maxpool
maxpool
Depth 64 3x3 conv Conv1_1 Conv1_2
Depth 128 3x3 conv Conv2_1 Conv2_2
Depth 256 3x3 conv Conv3_1 Conv3_2 Conv3_3 Conv3_4
Depth 512 3x3 conv Conv4_1 Conv4_2 Conv4_3 Conv4_4
Depth 512 3x3 conv Conv5_1 Conv5_2 Conv5_3 Conv5_4
Depth 4096 3x3 conv Conv6_1 Conv6_2
FC1
FC2
softmax
AISCIENCES

Fig 1

2Conv — 1 Maxpool — 2Conv — 1 Maxpool — 4 Conv — 1 Maxpool — 4 Conv — 1 Maxpool — 4 Conv — 1Maxpool — 1 FC — 1 FC — 1 FC

A VGG-19 is a Convolutional Neural Network - That utilizes 19 layers - having been trained on million of Image samples -

and utilizes the Architectural style of:

- Zero-Center normalization* on Images
- Convolution
- ReLU
- Max Pooling
- Convolution
- etc.
- UNTIL
- Fully Connected layer
- ReLU
- Dropout

- ➢ Fully Connected
- ➢ Softmax
- ➢ Classification output

a) Zero-Center normalization is the Centralization and Normalization towards Origo. It normalizes and reduces dimensions - to keep scale centralized - in terms of when we will perform Convolutions later on. The reason this is important - is to bring everything “down in line” in a normalized, streamlined and orderly fashion - so we have some sense of normality. As in - we want the general structure of what we are parsing - to be normalized and centralized - so that we have a pre-defined boundary that we are being relative towards.

b) Convolutions is the functional operation of performing concatenation of Functional Curves - so that they “add up to encapsulate how they affect each other - in terms of multiplicative relationships”.

c) ReLU is a rectifying linear unit. A unit that is made to rectify and compensate for signal parsing errors - such as when we are forced to encapsulate epsilon (infinitely small) - and we have to work around that. So ReLU “Steers back the signal” to where it’s supposed to be headed - so that the signal does not explode or vanish. Max Pooling is when you pool together the largest sample you can find - in an average area of taking strides.

d) Strides - is the average functional kernel mapping which you have in a square diagram plot that averages out the value samplings of a certain space. This effectively takes “the biggest impact features” and there of, “the largest information to retrieve” and then ignores the smaller features. So you are left with a smaller sample space (lower dimensionality) - But you keep the coarsest features - you keep the largest representative of information.

e) A Fully Connected Layer is a fully connected layer - that is utilized for Classification. Since this is used so sparingly - it’s at the end - when the rough feature extraction and averaging of Functional relationships have had its sequence Now it summarizes to run tally on the total output.

f) Softmax is an applicative algorithm - that normalizes a distribution dynamic from K amount of composite functional concatenations. This means that instead of having a spread where the values can be anything - 0, -1, 1, 2, etc. They are all normalized to be in line to a distribution (as the Softmax acts as a regularizer over a distribution)And then generalized to be across the j-th Linear function as it has become baked into the Distribution we have regressed forward with Softmax.

g) Classification output is the output of classification where it utilizes Cross Entropy to maximize the Probability - in terms of where each K-th unit came from and where it most likely belongs. There of - it stands in relation to the previous K functional relationships where it classifies and infers the probability of placement - of each.

5. Implementation of System:

Dataset

Images of Tomato disease have been taken from the Plant Village dataset. The dataset includes over 50,000 images of 14 crops, such as tomatoes, potatoes, grapes, apples, corn, blueberry, raspberry, soybeans, squash and strawberry. We selected tomatoes as our target crop.

The images of various classes of tomato diseases are as follows :

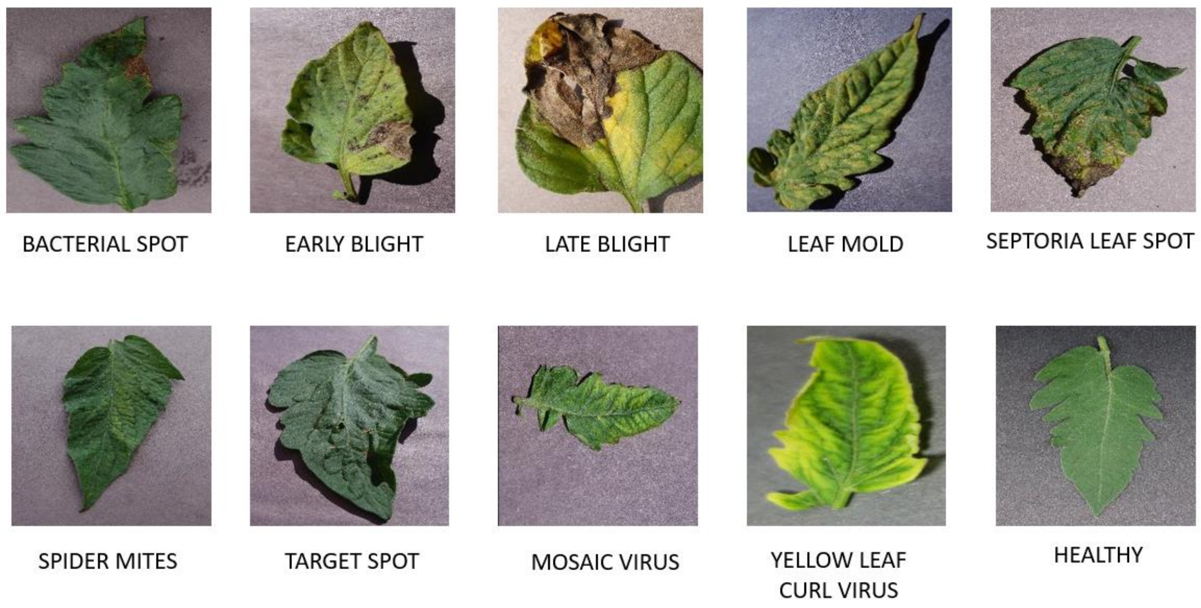


Fig. Nine Different types of diseases found in Tomato Leaf, i.e., Target Spot, 2) Mosaic virus, 3) Bacterial spot, 4) Late blight, 5) Leaf Mold, 6) Yellow Leaf Curl Virus, 7) Spider mites: Two-spotted spider mite, 8) Early blight and 9) Septoria leaf spot

There are mainly nine types of diseases in tomato: 1) Bacterial spot, 2) Early blight, 3) Late blight, 4) Leaf Mold, 5) Septoria leaf spot, 6) Spider mites: Two-spotted spider mite, 7) Target Spot, 8) Mosaic virus and 9) Yellow Leaf Curl Virus. In the proposed work, there are 18345 images in the training dataset and 4585 images in the validation dataset belonging to 10 different classes. Augmentation was done using the Augmentor package of python and it helps to build similar new images by rotating, flipping, cropping and resizing the existing images.

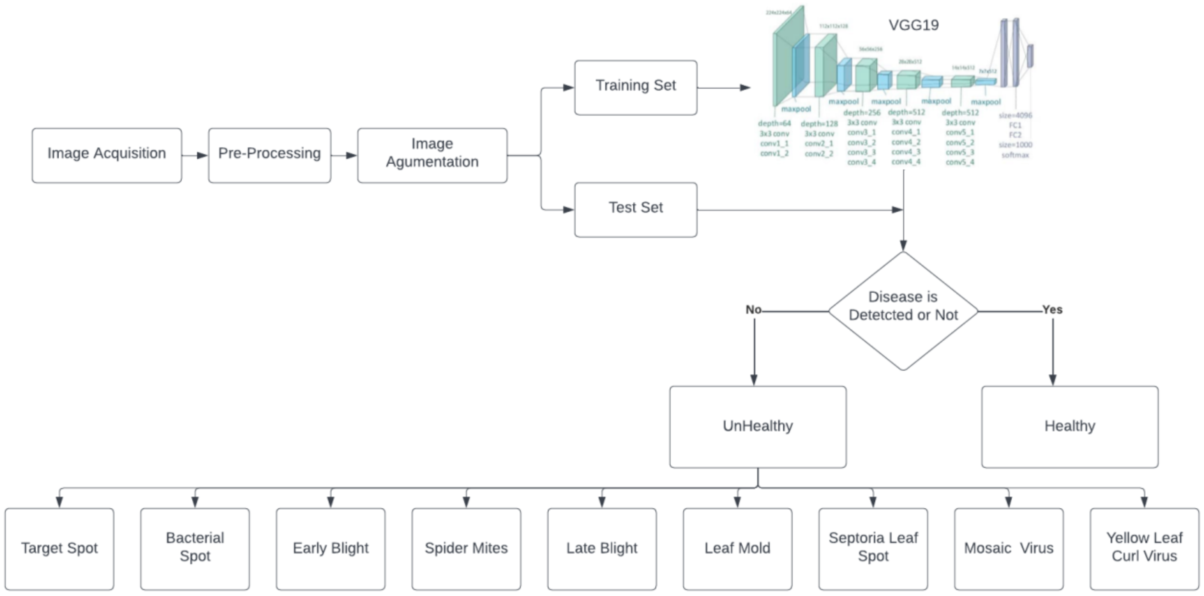


Fig 2

Figure 2 above shows the detailed flowchart of our proposed model.

In this detection, we will use the data set of normal and diseases tomates. The data set contains the directory trains and validation. Subdirectory inside these contains the images used to train, test, and validate the model. In total, we have 23,000 images. After understanding the data set we moved towards the training part. Firstly we will import all the useful libraries.

We have imported the library like NumPy, which is used for array processing, pandas, which is used in data frame creation and operation, OS, which is used in path and directory related operation. We will keep the size of all images the same, hence, resizing all the images.

We will declare the train and test path into the variable. After that in our case, we are using transfer learning techniques like VGG19 to get better accuracy. To perform this we have to import the VGG19 model. The input shape parameter will have plus 3 because of three RGB channels. The weights are taken from imagenet competition. Furthermore, the include_top is kept False because we only need to classify among two classes. To not retrain the pre-trained layers we have to make all the layers trainable as False. To get all the categories under the train folder path in a list we use a glob method.

By keeping include_top as False we have not specified the last layer in the model. Here, we will add our customized last layers. After specifying the last layer, we will now create an object model which has standard VGG input, which we have defined above, and prediction as output.Then we will compile the defined model object. After compiling, we load the data set with some pre-processing and image augmentation so our model gets more images to learn from. We use an image data generator. This helps to rescale images and generate more images by performing horizontal flip, shearing, and zooming on given images.Then we create a training set to feed into model training. This objective is achieved by using the flow from the directory function. Now, we will train the model with 10 epochs.

6. Result and Discussion

The model has trained and the architecture is tested successfully by providing new input images to classify whether its disease affected or not. And now we will implement a model to our field for disease identification. The accuracy and loss graph of training and validation for trained models using the data set is shown in Fig. 3. After successfully training the model on nine epochs it gives validation accuracy around 96%. A comparison accuracy chart is shown in Fig 4.

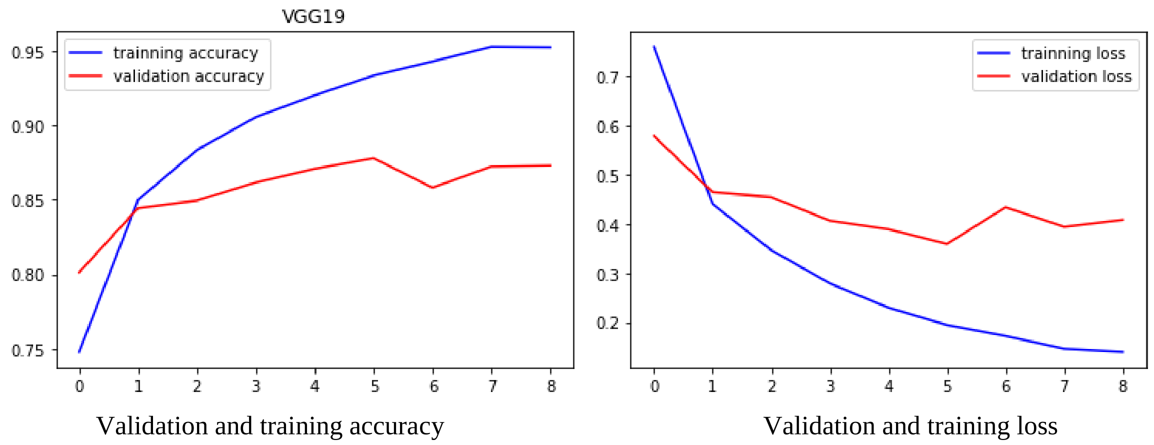


Fig 3

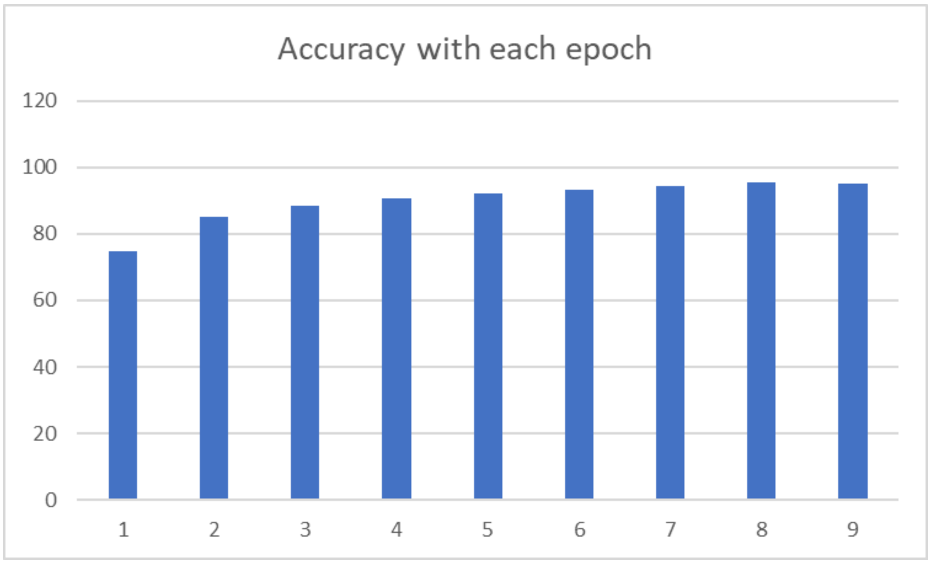
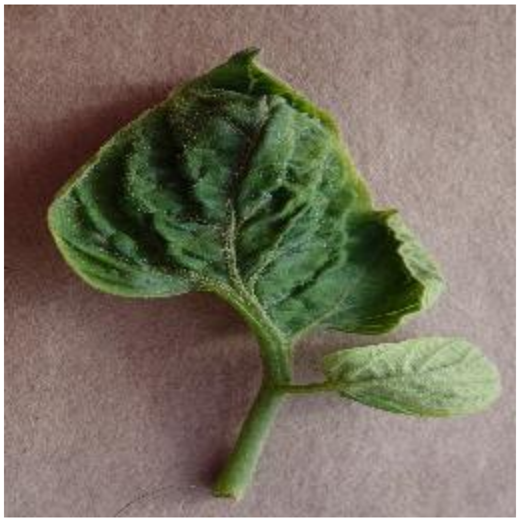


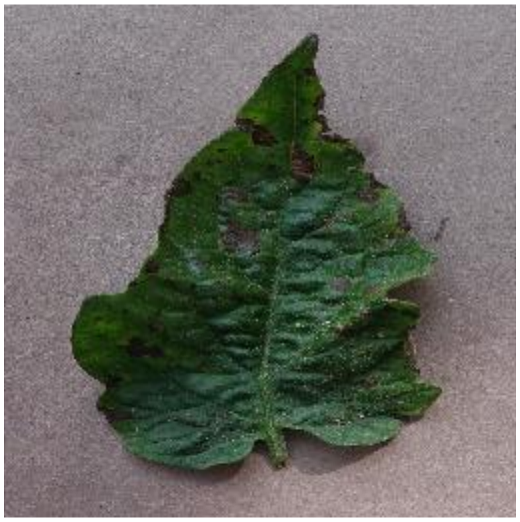
Fig 4

After the model is trained on simulation we validate by testing our model by giving some input image apart from testing data and training data but the image format is the same as testing and training data. We have testing our model by set of two different input images with disease as in Fig. 5 and without disease as in Fig. 6 and the output is shown below.



```
'Tomato___Tomato_Yellow_Leaf_Curl_Virus'
```

Fig 5



```
'Tomato___Early_blight'
```

Fig 6

Here a comparison chart is provided which shows the accuracy of other works with ours.

Comparison Chart

Papers	**Accuracy**
Turmeric Plant Diseases Detection and Classification using Artificial Intelligence.	95%
Tomato diseases Classification Based on VGG and Transfer Learning	97%
Apple Leaf Disease Identification and Classification using ResNet Models.	95%
Disease Classification in Eggplant Using Pre-trained VGG16 and MSVM.	98%
Application of Pre-Trained Deep Convolutional Neural Networks for Rice Plant Disease Classification	97%
Proposed Model	96%

7. Conclusion

Convolutional Neural Network helps to identify the infected and non infected leaf images based on model trained accuracy level at early. So, this VGG-19 architecture acts as an efficient method in the process of identifying and treating the disease much earlier than over affecting other plants in agricultural land. Thus, we can improve our agricultural yield and protect our land health from usage of unnecessary fertilizer and pesticides. The overall accuracy obtained by VGG-19 was 96%.

8. References

[1] A. Elhassouny and F. Smarandache,” Smart mobile application to recognize tomato leaf diseases using Convolutional Neural Networks,” 2019 International Conference of Computer Science and Renewable Energies (ICCSRE), Agadir, Morocco, 2019, pp. 1-4, doi: 10.1109/ICCSRE.2019.8807737.

[2] Fuentes, A., Park, D.S., Yoon, S., Youngki, H. and Lee, Y., Characteristics of Tomato Plant Diseases.

[3] Kaur, L., & Laxmi, V. (2016). Detection of Unhealthy Region of plant leaves using Neural Network. Dis Manag, 1(05), 34-42.

[4] Sangeetha, R.; Rani, M. Tomato Leaf Disease Prediction Using Transfer Learning. In Proceedings of the International Advanced Computing Conference 2020, Panaji, India, 5–6 December 2020.

[5] Wang, Y.; Zhang, H.; Liu, Q.; Zhang, Y. Image classification of tomato leaf diseases based on transfer learning. J. China Agric. Univ. 2019, 24, 124–130.

[6] Rangarajan, A.K.; Purushothaman, R.; Ramesh, A. Tomato crop disease classification using pre-trained deep learning algorithm. Procedia Comput. Sci. 2018, 133, 1040–1047

APPENDIX

TOMATO LEAF DISEASE DETECTION USING VGG19

Importing all dependencies

```
In [1]: import numpy as np # linear algebra
        import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
        import os
```

Adding the path

```
In [2]: path = "H:/3rd Year/6th SEM/TAA1- Tarp/Tomato/New Plant Diseases Dataset(Augmented)"
        os.listdir(path)

Out[2]: ['train', 'valid']
```

Join train and test with path

```
In [3]: train_path = os.path.join(path, "train")
        print(os.listdir(train_path))
        print("*"*100)
        test_path = os.path.join(path, "valid")
        print(os.listdir(test_path))

        ['Tomato___Bacterial_spot', 'Tomato___Early_blight', 'Tomato___healthy', 'Tomato___Late_blight', 'Tomato___Leaf_Mold', 'Tomato_
        __Septoria_leaf_spot', 'Tomato___Spider_mites Two-spotted_spider_mite', 'Tomato___Target_Spot', 'Tomato___Tomato_mosaic_virus',
        'Tomato___Tomato_Yellow_Leaf_Curl_Virus']
        ****************************************************************************************************
        ['Tomato___Bacterial_spot', 'Tomato___Early_blight', 'Tomato___healthy', 'Tomato___Late_blight', 'Tomato___Leaf_Mold', 'Tomato_
        __Septoria_leaf_spot', 'Tomato___Spider_mites Two-spotted_spider_mite', 'Tomato___Target_Spot', 'Tomato___Tomato_mosaic_virus',
        'Tomato___Tomato_Yellow_Leaf_Curl_Virus']
```

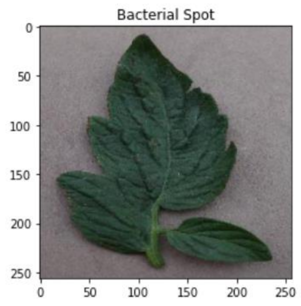
Applying glob function

```
In [4]: from glob import glob
        folders = glob("H:/3rd Year/6th SEM/TAA1- Tarp/Tomato/New Plant Diseases Dataset(Augmented)/train/*")
        folders

Out[4]: ['H:/3rd Year/6th SEM/TAA1- Tarp/Tomato/New Plant Diseases Dataset(Augmented)/train\\Tomato___Bacterial_spot',
         'H:/3rd Year/6th SEM/TAA1- Tarp/Tomato/New Plant Diseases Dataset(Augmented)/train\\Tomato___Early_blight',
         'H:/3rd Year/6th SEM/TAA1- Tarp/Tomato/New Plant Diseases Dataset(Augmented)/train\\Tomato___healthy',
         'H:/3rd Year/6th SEM/TAA1- Tarp/Tomato/New Plant Diseases Dataset(Augmented)/train\\Tomato___Late_blight',
         'H:/3rd Year/6th SEM/TAA1- Tarp/Tomato/New Plant Diseases Dataset(Augmented)/train\\Tomato___Leaf_Mold',
         'H:/3rd Year/6th SEM/TAA1- Tarp/Tomato/New Plant Diseases Dataset(Augmented)/train\\Tomato___Septoria_leaf_spot',
         'H:/3rd Year/6th SEM/TAA1- Tarp/Tomato/New Plant Diseases Dataset(Augmented)/train\\Tomato___Spider_mites Two-spotted_spider_m
        ite',
         'H:/3rd Year/6th SEM/TAA1- Tarp/Tomato/New Plant Diseases Dataset(Augmented)/train\\Tomato___Target_Spot',
         'H:/3rd Year/6th SEM/TAA1- Tarp/Tomato/New Plant Diseases Dataset(Augmented)/train\\Tomato___Tomato_mosaic_virus',
         'H:/3rd Year/6th SEM/TAA1- Tarp/Tomato/New Plant Diseases Dataset(Augmented)/train\\Tomato___Tomato_Yellow_Leaf_Curl_Virus']
```

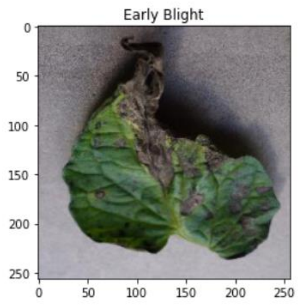
```
In [5]: import matplotlib.pyplot as plt
        plt.imshow(plt.imread("H:/3rd Year/6th SEM/TAA1- Tarp/Tomato/New Plant Diseases Dataset(Augmented)/train/Tomato___Bacterial_spot/
        plt.title("Bacterial Spot")

Out[5]: Text(0.5, 1.0, 'Bacterial Spot')
```



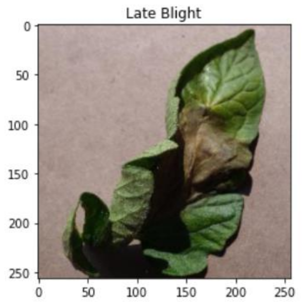
```
In [6]: plt.imshow(plt.imread("H:/3rd Year/6th SEM/TAA1- Tarp/Tomato//New Plant Diseases Dataset(Augmented)/train/Tomato___Early_blight/(
        plt.title("Early Blight")

Out[6]: Text(0.5, 1.0, 'Early Blight')
```



In [7]:
```
plt.imshow(plt.imread("H:/3rd Year/6th SEM/TAA1- Tarp/Tomato/New Plant Diseases Dataset(Augmented)/train/Tomato___Late_blight/00
plt.title("Late Blight")
```

Out[7]:
```
Text(0.5, 1.0, 'Late Blight')
```



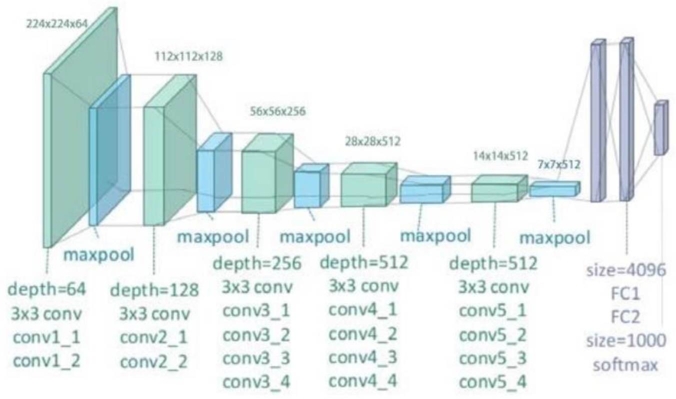
Importing necessary packages

In [8]:
```
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
```

Image size

In [9]:
```
SIZE = [128, 128]
```

Importing VGG19



In [10]:
```
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.vgg19 import VGG19
```

vgg19 with input shape and weight is imagenet

In [11]:
```
vg19 = VGG19(input_shape=SIZE + [3], weights="imagenet", include_top=False)
```

In [12]:
```
for layer in vg19.layers:
    layer.trainable = False
```

In [13]:
```
x = Flatten()(vg19.output)
```

In [14]:
```
prediction = Dense(len(folders), activation="softmax")(x)

modelvg = Model(inputs=vg19.input, outputs=prediction)
```

VGG19 model summary

```
In [15]: modelvg.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 128, 128, 3)]     0

 block1_conv1 (Conv2D)       (None, 128, 128, 64)      1792

 block1_conv2 (Conv2D)       (None, 128, 128, 64)      36928

 block1_pool (MaxPooling2D)  (None, 64, 64, 64)        0

 block2_conv1 (Conv2D)       (None, 64, 64, 128)       73856

 block2_conv2 (Conv2D)       (None, 64, 64, 128)       147584

 block2_pool (MaxPooling2D)  (None, 32, 32, 128)       0

 block3_conv1 (Conv2D)       (None, 32, 32, 256)       295168

 block3_conv2 (Conv2D)       (None, 32, 32, 256)       590080

 block3_conv3 (Conv2D)       (None, 32, 32, 256)       590080

 block3_conv4 (Conv2D)       (None, 32, 32, 256)       590080

 block3_pool (MaxPooling2D)  (None, 16, 16, 256)       0

 block4_conv1 (Conv2D)       (None, 16, 16, 512)       1180160

 block4_conv2 (Conv2D)       (None, 16, 16, 512)       2359808

 block4_conv3 (Conv2D)       (None, 16, 16, 512)       2359808

 block4_conv4 (Conv2D)       (None, 16, 16, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 8, 8, 512)         0

 block5_conv1 (Conv2D)       (None, 8, 8, 512)         2359808

 block5_conv2 (Conv2D)       (None, 8, 8, 512)         2359808

 block5_conv3 (Conv2D)       (None, 8, 8, 512)         2359808

 block5_conv4 (Conv2D)       (None, 8, 8, 512)         2359808

 block5_pool (MaxPooling2D)  (None, 4, 4, 512)         0

 flatten (Flatten)           (None, 8192)              0

 dense (Dense)               (None, 10)                81930

=================================================================
Total params: 20,106,314
Trainable params: 81,930
Non-trainable params: 20,024,384
_________________________________________________________________
```

Compile model

```
In [16]: modelvg.compile(loss="categorical_crossentropy", metrics=["accuracy"], optimizer="adam")
```

Data Augmentation

```
In [17]: train_datagen_vg19 = ImageDataGenerator(rescale=1./255)

         test_datagen_vg19 = ImageDataGenerator(rescale=1./255)
```

```
In [18]: trainning_set_vg19 = train_datagen_vg19.flow_from_directory(train_path,
                                                             target_size=(128, 128),
                                                             batch_size=16,
                                                             class_mode="categorical", shuffle=True)
```

```
Found 18345 images belonging to 10 classes.
```

```
In [19]: testing_set_vg19 = test_datagen_vg19.flow_from_directory(test_path,
                                                             target_size=(128, 128),
                                                             batch_size=16,
                                                             class_mode="categorical", shuffle=False)
```

```
Found 4585 images belonging to 10 classes.
```

Model fit_generator

In [20]:
```
import tensorflow as tf

callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=3)

r_vg19 = modelvg.fit_generator(trainning_set_vg19,
                       validation_data=testing_set_vg19,
                       epochs=10,
                       callbacks=[callback]
                       )
```

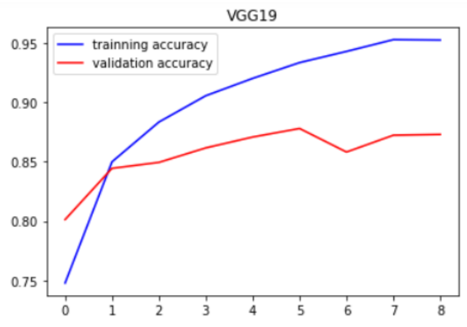
```
<ipython-input-20-11692eb06477>:5: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Pl
ease use `Model.fit`, which supports generators.
  r_vg19 = modelvg.fit_generator(trainning_set_vg19,
```

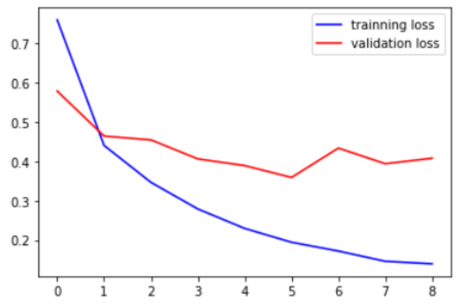
```
Epoch 1/10
1147/1147 [==============================] - 2289s 2s/step - loss: 0.7595 - accuracy: 0.7478 - val_loss: 0.5785 - val_accuracy:
0.8013
Epoch 2/10
1147/1147 [==============================] - 2038s 2s/step - loss: 0.4402 - accuracy: 0.8500 - val_loss: 0.4641 - val_accuracy:
0.8445
Epoch 3/10
1147/1147 [==============================] - 1901s 2s/step - loss: 0.3463 - accuracy: 0.8835 - val_loss: 0.4543 - val_accuracy:
0.8495
Epoch 4/10
1147/1147 [==============================] - 2531s 2s/step - loss: 0.2787 - accuracy: 0.9057 - val_loss: 0.4061 - val_accuracy:
0.8617
Epoch 5/10
1147/1147 [==============================] - 2383s 2s/step - loss: 0.2293 - accuracy: 0.9203 - val_loss: 0.3890 - val_accuracy:
0.8709
Epoch 6/10
1147/1147 [==============================] - 2260s 2s/step - loss: 0.1940 - accuracy: 0.9336 - val_loss: 0.3588 - val_accuracy:
0.8781
Epoch 7/10
1147/1147 [==============================] - 1848s 2s/step - loss: 0.1717 - accuracy: 0.9430 - val_loss: 0.4335 - val_accuracy:
0.8582
Epoch 8/10
1147/1147 [==============================] - 1850s 2s/step - loss: 0.1456 - accuracy: 0.9530 - val_loss: 0.3937 - val_accuracy:
0.8724
Epoch 9/10
1147/1147 [==============================] - 2058s 2s/step - loss: 0.1392 - accuracy: 0.9526 - val_loss: 0.4077 - val_accuracy:
0.8731
```

Visualization for VGG19

In [21]:
```
import matplotlib.pyplot as plt
accuracy = r_vg19.history['accuracy']
val_accuracy = r_vg19.history['val_accuracy']
loss = r_vg19.history['loss']
val_loss = r_vg19.history['val_loss']
epochs = range(len(accuracy))
plt.title("VGG19")
plt.plot(epochs, accuracy, "b", label="trainning accuracy")
plt.plot(epochs, val_accuracy, "r", label="validation accuracy")
plt.legend()
plt.show()

plt.plot(epochs, loss, "b", label="trainning loss")
plt.plot(epochs, val_loss, "r", label="validation loss")
plt.legend()
plt.show()
```





```
In [22]: y_pred = modelvg.predict(testing_set_vg19)
```

```
In [23]: y_pred
```

```
Out[23]: array([[9.9912912e-01, 3.4328410e-04, 3.2787542e-05, ..., 4.6819999e-04,
                 1.2042792e-11, 6.8430789e-14],
                [9.9869722e-01, 1.2853534e-03, 7.8940293e-06, ..., 1.7027174e-08,
                 4.4990799e-15, 1.0563260e-12],
                [9.9984777e-01, 8.7998596e-05, 2.0946256e-05, ..., 9.1979618e-06,
                 6.6090261e-10, 9.5537231e-09],
                ...,
                [4.2085323e-04, 3.0433375e-05, 3.8465066e-04, ..., 5.0753115e-06,
                 5.4985878e-07, 9.9796343e-01],
                [3.4840484e-04, 2.6550245e-07, 1.5368896e-06, ..., 7.6961122e-09,
                 1.5528991e-09, 9.9959749e-01],
                [1.7293416e-08, 2.7507371e-03, 7.5074161e-07, ..., 5.9858024e-08,
                 1.1517774e-05, 9.7133934e-01]], dtype=float32)
```

```
In [24]: y_pred = np.argmax(y_pred, axis=1)
         y_pred
```

```
Out[24]: array([0, 0, 0, ..., 9, 9, 9], dtype=int64)
```

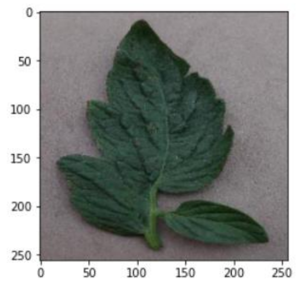
Tomato Bacterial Spot

```
In [25]: test_img = plt.imread("H:/3rd Year/6th SEM/TAA1- Tarp/Tomato/New Plant Diseases Dataset(Augmented)/train/Tomato___Bacterial_spot/
```

Visualize

```
In [26]: plt.imshow(test_img)

Out[26]: <matplotlib.image.AxesImage at 0x22644b6f4f0>
```



Save model

```
In [27]: modelvg.save("vgg_19tl.model")

         INFO:tensorflow:Assets written to: vgg_19tl.model\assets
```

load a image and detection

```
In [28]: import cv2
         import tensorflow as tf
         def prepare(filepath):
             img_array = cv2.imread(filepath, cv2.IMREAD_COLOR)
             img_array = img_array / 255
             new_array = cv2.resize(img_array, (128, 128))
             return new_array.reshape(-1, 128, 128, 3)

         model = tf.keras.models.load_model("vgg_19tl.model")
```

Prediction

Class Dictionary

```
In [29]: class_dict = trainning_set_vg19.class_indices
         class_dict

Out[29]: {'Tomato___Bacterial_spot': 0,
          'Tomato___Early_blight': 1,
          'Tomato___Late_blight': 2,
          'Tomato___Leaf_Mold': 3,
          'Tomato___Septoria_leaf_spot': 4,
          'Tomato___Spider_mites Two-spotted_spider_mite': 5,
          'Tomato___Target_Spot': 6,
          'Tomato___Tomato_Yellow_Leaf_Curl_Virus': 7,
          'Tomato___Tomato_mosaic_virus': 8,
          'Tomato___healthy': 9}
```

label

```
In [30]: def prediction_cls(prediction):
             for key, clss in class_dict.items():
                 if np.argmax(prediction) == clss:
                     return key
```

```
In [31]: prediction = model.predict([prepare("H:/3rd Year/6th SEM/TAA1- Tarp/Tomato/New Plant Diseases Dataset(Augmented)/valid/Tomato___h
         prediction_cls(prediction)

Out[31]: 'Tomato___healthy'
```

```
In [32]: prediction = model.predict([prepare("H:/3rd Year/6th SEM/TAA1- Tarp/Tomato/New Plant Diseases Dataset(Augmented)/valid/Tomato___T

         prediction_cls(prediction)

Out[32]: 'Tomato___Tomato_Yellow_Leaf_Curl_Virus'
```

```
In [34]: epare("H:/3rd Year/6th SEM/TAA1- Tarp/Tomato/New Plant Diseases Dataset(Augmented)/valid/Tomato___Early_blight/0e2abcfb-e62b-4c61

Out[34]: 'Tomato___Early_blight'
```

```
In [35]: prediction = model.predict([prepare("H:/3rd Year/6th SEM/TAA1- Tarp/Tomato/New Plant Diseases Dataset(Augmented)/valid/Tomato___T

         prediction_cls(prediction)

Out[35]: 'Tomato___Tomato_mosaic_virus'
```