Tomato Plant Diseases Detection and Classification

Sameh Mohammed Abdo Khaled

MSc Information Technology, Bhaarathi I M.Sc.(S/W Engg), B.Ed. (CS), MBA(ISM), MA (Ed Psy)
Student, Rathinam College of Arts and Science,

Assistant Professor Department of Computer Science, Rathinam College of Arts and Science

Abstract: ***This project aims to develop a machine learning model to detect and classify diseases in tomato plants. The increasing demand for food and the need to maintain food security has led to the development of advanced technologies in the agriculture sector. One of the most challenging tasks in agriculture is to identify and diagnose diseases in crops, which can significantly reduce their productivity. In this project, we will use computer vision and machine learning techniques to automate the process of disease detection and classification in tomato plants. The first step in this project will be to collect a large dataset of tomato plant images and annotate them with disease labels. This dataset will be used to train a convolutional neural network (CNN) model. The CNN model will learn to recognize patterns in the images that are associated with specific diseases. We will then use this trained model to classify new images of tomato plants and detect the presence of any diseases. The performance of the model will be evaluated based on accuracy, precision, recall, and F1 score. The results of this project will be compared to existing methods in the literature to demonstrate the effectiveness of the proposed approach. In conclusion, this project has the potential to make a significant contribution to the field of plant disease detection and classification. By automating the process of disease detection, we can help farmers to quickly and accurately identify diseases in their crops, reducing the impact of diseases on productivity and improving food security. The results of this project will also provide a foundation for further research in the area of computer vision and machine learning for agriculture.***

Keywords: Convolutional Neural Network, bacterial infections, tomato disease identification, Late blight, Spider mites, Bacterial Spot on tomato leaves.

1. Introduction

1.1 Overview of the Project

Agriculture is amongst the most seasoned societies created by individuals. It remains the mother of all societies till date and has assumed a noteworthy job in the headway advancement of mankind. Farming instruments and practices like water system, strip editing, compost, manures, pesticides and so on have been utilized since quite a while, however have made huge enhancements over the most recent hundred years. Indeed, even around the nineteenth century, farming practices were sufficiently improved to yield commonly the collect per unit land, contrasted with the earlier years. Agribusiness is the communication of seed, soil and agrarian synthetic concoctions. Therefore, appropriate administration and care of all perspectives is compulsory for the maintainability of the agrarian framework. The point of improving farming generation without investigating natural effects has caused ecological corruption. Therefore, the objective ought to be to upgrade agribusiness with negligible natural harm or debasement. Plants develop wherever around the globe, even where people don't live. A few plants convey a great deal of data for the improvement of human progress. Since, the event of ailment in plants is normal, recognizing infection would assume a significant job in agrarian advancement. In tomato plant (Solanum lycopersicum), the rundown of ailments happening is long. Bacterial Spot on tomato leaves is a unique and perilous infection ever to exist. It spreads at an exceptionally quick rate and requires extraordinary exertion and venture to contain. Mostly, this sickness contorts the tomato plants to the degree of extreme diminishing in their attractiveness. Bacterial Spot is brought about by a bacterium named xanthomonas campestris pv. The principal manifestation of bacterial spot in tomato plants are oily, little and unpredictable checks in the lower face of the leaves. These little spots at first appear to be dim green, however later turn purple and dark. These spots may contain dark focuses and yellow or white shading inside. Bacterial spot gradually harms leaf tissue and uncovered the tomato natural product to cruel daylight which results in dull dark colored knock like development on the organic product, which influences profitability of the harvest. In this work we will concentrate on distinguishing and curing the accompanying ailments that happen/show side effects, in the leaves of the tomato crop.

1. Septoria leaf spot 2. Late blight 3. Spider mites/Two spotted spider mite 4. Early blight 5. Target Spot 6. Bacterial spot Thus, our dataset consists 1000 images of each of the above diseases in tomato plant leaves along with images of healthy tomato leaves. Our work involves collecting tomato leaf images, both healthy and infected, and analyzing them for diseases/ symptoms using image processing. The algorithm used in this work is a Convolutional Neural Network (CNN) which are a part of the deep neural networks and are mostly used in analysing visual images. Some widely used CNN architectures are mentioned-ResNet, GoogLeNet, AlexNet, VGGNet, LeNet. Convolutional Neural Networks can be thought of as an artificial brain at work, solving a number of problems that are happening around us every day. It uses a huge variety of multilayer perceptron that do not require much pre-processing and collectively try to mimic a biological neural network. The building blocks of a CNN are five different layers, namely, 1. Convolutional layer 2. Rectified Linear Unit layer 3. Fully connected layer 4. Pooling layer 5. Loss Layer

1.2 Organization Profile

Zeboto is a Digital Marketing Company in Coimbatore We provide optimized solution to your business needs in a cost effective manner. We are sure that the client will experience the world class quality in our product and services. Our web solutions are handled professionally to save on your precious investment and time as per the global standards. Zeboto as a company believes in hard work to become key service provider with quality. We strive to carry this forward and forge strong long- term relationships with our clients. We are focusing on providing world-wide online solutions. We create successful digital experiences that address our client's specific business goals and solve user challenges. We have the team and skills to create a site that you will be proud to show off, but our relationship and service are what make us a company that you will love having as a partner. Our vision is to be trusted & respected as a world-class web development company in delivering and developing state-of-the-art, innovative IT solutions for our clients to improve profits as well as build efficiency.

2. Existing System

Diseases such as rust, bacterial infections, late blight etc can plague the leaves of common crop plants. It is a common occurrence in the agricultural sector. Detecting the diseases and identifying their possible remedies is a cumbersome and tedious task. It is also often inaccurate and requires expert help. Farmers often lose out on their yields due to this. Getting an expert to come down and manually check out the leaves is time-consuming and often unreliable.

2.1 Disadvantages of the existing system

1) Not secure
2) It is not user friendly
3) Need lot of paperwork
4) It requires an internet connection.
5) It requires a large database.

2.2 Proposed System

In this project, we focus on creating a solution that would be an easy fix for the leaf disease detection problems. We have collected a large dataset consisting of images of healthy and diseased leaves. We built a web app for the detection of the disease a crop leaf is afflicted with, based on a picture of the leaf. The app will make use of ML algorithms to analyze and predict the disease a leaf has. It will use a model that has been trained on pre-identified diseased leaf images. Based on it, any newly encountered diseased leaf will be identified by its disease. The input image can be either an uploaded one or clicked through the phone camera. The output will be the disease name it has been identified with. Thus, a classification and identification of the disease is enabled. An option to look up remedies for the disease will also be provided. An additional option for a front-end is also made- a Rest Api service that hosts a website to allow entering of a picture name and displaying its output class.

Advantages of the proposed system

1) Provides a convenient, easy-to-use approach to identify leaf diseases.
2) Reduces the need for human dependency in detecting and remedying the disease.
3) Farmers can detect and remedy their crop leaves in an efficient manner.
4) Increases crop production yield and quality.
5) It saves time and effort in the detection process.
6) Provides an accurate and reliable model for identification.
7) Gives options for the user-interface. 8. Can be used to detect uploaded images even in the absence of an internet connection.

2.2.1 Problem Definition

The problem to be addressed in the tomato disease identification project is the accurate and timely detection of diseases that affect tomato plants. Tomato plants are vulnerable to a range of diseases that can significantly reduce crop yield and quality, including fungal, bacterial, and viral infections. Early detection and timely treatment are crucial to preventing the spread of these diseases and minimizing their impact on tomato production. The objective of this project is to develop an automated system that can identify tomato diseases accurately and quickly. The system should be able to detect multiple diseases and provide recommendations for appropriate treatments based on the identified disease. The system should be user-friendly and accessible to tomato growers with minimal technical expertise. To achieve this objective, the project will require collecting a large dataset of tomato plant images that includes both healthy plants and plants affected by various diseases. This dataset will be used to train machine learning models to accurately identify and classify different tomato diseases. The trained model will be integrated into a user-friendly application that can be used by farmers to identify tomato diseases and receive recommendations for appropriate treatment

2.2.2 Module Description

The tomato disease identification project can be divided into several modules, each of which performs a specific task or set of tasks. These modules include: 1. Image Acquisition: The first module involves collecting high-quality images of tomato plants that include both healthy plants and those affected by various diseases. These images will be used to train the machine learning models that will be used for disease identification. 2. Image Pre-processing: The acquired images will undergo several pre-processing steps to enhance their quality, including resizing, normalization, and color correction. This module is critical to ensure that the images are of uniform quality, which will improve the accuracy of the machine learning models. 3. Disease Identification: This module uses machine learning models to identify tomato diseases in the pre-processed images. The models will be trained using a dataset of labeled images and will use various computer vision techniques to detect disease symptoms. 4. Disease Classification: Once a disease has been identified in an image, this module classifies the disease into a specific category. The classification process involves comparing the identified disease symptoms with a database of known tomato diseases and selecting the most likely match. 5. User Interface: The user interface module is responsible for providing an easy-to-use interface for tomato growers to input images and receive disease identification. This module should be intuitive and accessible to users with minimal technical expertise. 6. Database Management: This module is responsible for managing the dataset of labeled images and the database of known tomato diseases and their treatments. It includes data storage, retrieval, and management functions. Overall, the modules work together to provide a complete and automated system for accurate and timely identification of tomato diseases and recommendations for appropriate treatments.

3. System Design

3.1 Input Design

CNN

A Convolutional neural network (CNN or ConvNet) is utilized to deal with the issue articulation. It is a unique sort of multi-layer NN that is intended to perceive visual examples straightforwardly from pixel pictures alongside least pre-handling. It is along these lines compelling in picture arrangement situations.

Activation Function

CNNs likewise utilize Activation works that are utilized to start up a reaction like the functions in the human mind neurons. They trigger a response once aspecific edge limit has been come to. We can look over various them for our utilization, as Relu, Softmax, Sigmoid and so on. RELU - It stands for rectified linear unit and is the most commonly used activation function. Its use is very common in CNNs and deep learning. It is a half-rectified function. f(z) denotes a value that becomes zero when z is negative and becomes z when the value of z is greater or equal to zero. Thus, this function ranges from zero to infinity. The function is monotonic in nature. The function should be used only in the hidden layers. SIGMOID - This function has a curve that looks like an S. It ranges from zero to one. Its most applicable in situations that demand a prediction of probabilistic output. The function is differentiable and monotonic both. It shows low convergence. SOFTMAX - This function is also called as the normalised exponential function. It is most often used in the final layer of the neural network. The output generated by it is large if the input aka logit is large. It produces a small output for a small input. It ranges from zero to one and is very useful in situations that need an output range of probabilities.

Optimizers

An optimizer is used to improve the model to its best capability by tampering with the weights in a neural network. It takes into account the loss function. Often, they face the issue of being stuck at the local minima. Basically, it is the path we use to reduce the loss in a neural network. It calculates and updates the parameters of the model in such a way to cause minimum loss to the training. They fall in 2 types of categories-First Order Optimization - The loss function is reduced or increased based on its gradient values and parameters. It tells if the function is growing or reducing at the point. Gradient Descent is the most common of all. They are also famous for being quite fast. Second Order Optimization - They use the 2nd order derivative to increase or decrease the loss function. This approach is usually unused as it is quite costly. It is useful to tell us about the curvature. They are also better in case the performances are compared. Few of the optimizers seen in practice are- 1. Gradient descent 2. Adagrad 3. Adam 4. Stochastic gradient descent

ADAM - It denotes adaptive moment estimation. It makes use of the past to find out the current. This optimizer basically adds parts of the previous gradients to the current one. It has been very popular for its use in training NNs in recent times. ADAGRAD - This optimizer is known for its ability to have different learning rates in tune for individual needs. That is, different weights may have varying learning rates depending on need. However, the rates tend to reduce over time. Even then, they are very popular for sparse datasets. GRADIENTDESCENT - It is an iterative method and is very popular. It adjusts the weights of the network based on the gradients calculated. The gradient at a point can be got by the differentiation with respect to theta. STOCHASTIC GRADIENT DESCENT - This optimizer uses a unit batch size for every iteration. Thus, it may be good for multiple iterations but has a drawback of being very noisy. It is also an iterative method and quite useful for sparse Ml problems.

Loss Functions

It is used to find out how good the dataset is getting modelled by the training process. It is aspired to keep a minimum value for it for best results. This is also known as a cost function. Some of them are - 5. Cross-Entropy 6. Huber 7. Mean absolute error 8. Mean squared error CROSS-ENTROPY - Used to measure for a classification model with probability value lying between the range of zero and one. As the prediction diverges from the label, the loss is found to be increasing. An ideal model should have zero log loss. HUBER - Used in regression cases. HINGE - Found suitable for classification scenarios. MEAN ABSOLUTE ERROR - It has been popularly used in regression scenarios. It is defined as the sum of absolute differences between the actual and predicted values. Hence, the avg magnitude of errors is checked, without looking at the direction. MEAN SQUARED ERROR - It denotes the avg of the squared error of loss functions that are used for least squares regression. Its the most commonly used for regression scenarios. It denotes the sum of squared distances of predicted and target values.

3.2 Output Design

OVERVIEW We make use of the Core Ml framework that is found in Web, in conjunction with a ML model that has been trained on any prior algorithm. This can be used to classify the inputs. Another important framework used here is the Vision framework that acts with the earlier one to enable easier classification by Core Ml by subjecting the image data set to some image processing. This app uses our custom created model, to identify an image using three layers of classification.

GETTING STARTED This code project runs on ubuntu, on window 10 and above.

PREVIEW THE WEB APP The app can be seen in action by building and running the project on Ubuntu, then making use of the buttons in the app's first screen to successfully click a picture or to choose a picture from the photo library.

The app then uses the Vision framework to apply the Core ML model to the inserted image, which is our input parameter, and shows the result in the form of classification labels. On the developer side we can also see the numbers indicating the confidence level of each input parameter (leaf's picture) and its classification. It displays the classification which is the culmination of all the classifications based on confidence score.

Set Up Vision VIA Core ML

A file is generated by the CoreMc by default, which enables easy access to the custom model.

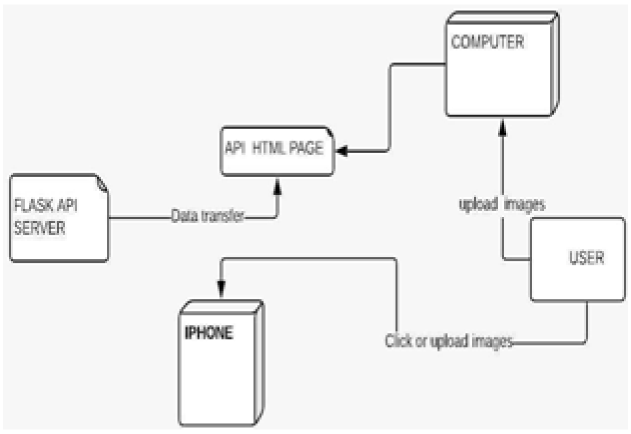
Creation of an instance of a class and subsequent use of its model property to create a VNCoreMLRequest request is the way to go about in the process of setting up a Vision request. After the request is run, the request object's completion handler is used to mention the method to get the results from the model. Vision is tasked with scaling or cropping the images to fit the Ml model, as each model needs the input pictures to be in a fixed ratio, when in reality, the pictures may be of varied aspect ratios. To ensure best performance, the request's imageCropAndScale- Option property is set in accordance with the layout used in the training process of the model.

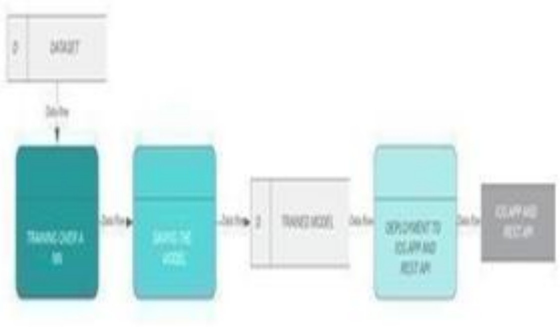
Image Orientation

The images are converted to a proper orientation, so all the images are in the same aspect ratio and thus pixel to pixel image processing is applied equally on all the pixels of all the images. We convert UIImages to CGImages in linux because values of this type define the pixel coordinate origin point as 0, 0 and the direction of the coordinate axes relative to the indented display orientation of the image. These values are taken from the image metadata.

Using Storyboards

Storyboards are used to enable prototypes and to facilitate the creation of many view controllers, in the same file, which is usually called Main.Storyboard. Before Storyboards were launched, one had to create a UI using XIB files and you could only create one file per view (UICollectionViewCell, UIHeaderView or any other UIView types). Their many advantages can be seen below: - 1. Connection of all the views using segues after laying them out. It helps us generate a visual representation of the user interface and the interaction between each element and screen. 2. Description of the transition from one screen to another is possible. Due to segues, we need to write less code for our UI. 3. We have used tableview in our app to represent the remedies for the diseases. Making tableview is very easy using storyboard and static cells now, we can also create dynamic cells for our custom use. 4. Auto Layout is another feature that has been used in the app which enables the use of mathematical functions to define relationships among elements and also to give them dynamic, run-time size and positing. This helps in making a dynamic user interface for different screen sizes for different devices. 5. Storyboards are written in XML which is a markup language. So, one can either use





3.3 Testing Categories

Testing is a series of investigations or rather "tests" conducted on unfinished products to reveal all quality and vulnerability issues. It is used to confirm that the expected and the predicted results are following suit as per the set expectations. One or more properties is checked for in the process of running a software part or the software in its entirety. This process can be carried out either in person or by the use of some tools. Both of them are very important in combing out the bugs that are certain to cause loss to the system. Losses caused due to testing are evident in many cases like- Nissan cars, Starbucks, Bloomberg terminal in London etc.

The testing process can be divided in various types:
Functional: -
1) Unit
2) Beta
3) System
4) Integration, etc.

Non-functional: -
1) Ability to use
2) How secure it is
3) Extent of its performance
4) Its installation process, etc.

Major types of testing taken into consideration are: -

Features:-
1) The model is tested for different images to check accuracy of detection.
2) User entered image names are used as input.
3) The image is read using PILand resized to 256x256 before passing to the model.

4) We have a precision recall for all the images being tested on the trained dataset.

1) **Black box testing:** It consists of a method which involves the ignorance of the tester about any of the system's internal workings. He has no idea how either the structure or the working of the project is being carried out. As it is based on an oblivious tester, focus is entirely the kind of inputs fed to system. It's totally dependent on the software needs demands. Some common tools put into action for this testing are- Selenium, Jmeter etc. The strategies used for these are- Decision table, Boundary value, Equivalence class.
2) **White box testing:** It consists of a method which involves the tester being abreast of all the system's internal workings. Here, the entire code can be clearly seen by the user. The test cases are framed after taking into consideration the system's structure and the program code in its entirety. The focus is entirely based on the flow of the system in its working process. The loopholes looked for here are- bad paths, predicted output, security weak spots, inputs flow pattern, etc.

Other levels of testing are seen as: -
1) **Unit testing:** Here, the smallest parts of the system are checked for any issues in its individuality.
2) **Integration testing:** Here, the various system parts are consolidated and checked to see if they work fine even after merging and getting connected to each other.
3) **System testing:** Done after the earlier mentioned testing, here, the entire system is checked as a whole to see if it is functioning well.
4) **Acceptance testing:** Here, the entire system is checked to see it has its compliance with the mentioned needs for the software and is working as per user demands satisfactorily and does not miss out on any of them.

In our project: -
1) The trained model is tested with over 14 images in each of the 7 classes. The 98 images are then checked for accuracy using the existing labeled classes. It was found that there was an accuracy of 91 per cent.
2) Apart from the testing of the model after the training, testing was done on the web API.
3) Images were downloaded and tested on both the interfaces. We found favorable results on both of the front-end mechanisms.
4) The web app was also tested by clicking images using the file uploader. Results were satisfactory in that case too. The project was tested on various test-cases and the result generated in the form of a table.

Dataset Loading
1) The dataset is imported into the Python Jupyter notebook using Pytorchs's modules like- torchvision.datasets. ImageFolder and torch.utils.data. Dataloader.
2) Data is split into training and testing parts.
3) Images are transformed using Pytorch's transforms module. They are converted to tensors and normalized.
4) Few images are displayed.

Neural Network Creation
1) A CNN is created consisting of 2 convolution sets and 3 FC layers
2) Filter size used is 5. Input image is of size 256x256 and has 3 channels.
3) The activation functions used are Relu and Softmax.
4) The Optimizer used is Adam.
5) Loss function is CrossEntropyLoss().

Training and Model Saving
1) The epochs are run 5 times and running loss is displayed every 50 mini-batches.
2) After training, the model is saved with a.pth extension.
3) The Trained dataset accuracy is around 91 percent.
4) This model is later converted to ml model extension.

Deployment to Web API
1) Flask is used in Python to create the Web Api.
2) A HTML file is made for thesite layout and the model is invoked in a Python file.
3) User entered image name is sent fromHTML input to Flask application.
4) There, model is invoked and predicted result is sent back to Html to be displayed to user.
5) We use Crosss-Origin Resource Sharing (CORS)(app) for cross-origin sharing.
6) The @app.route () functions are used to call upon the requests.
7) Flask runs on port 5000 and site template on port 8005.
8) The entire data transfer is done in JSON format.

3.4 System Implementation

Training the dataset and creation of model:
We loaded the dataset and created a CNN. The data was trained over the NN and the result saved as a model with. pth extension. The first 2 layers of the NN used here are based on the Le Net CNN architecture-proposed by Yann Lecun in 1998. The NN has 2 Conv layers and 3 FC layers. Relu and Softmax activation functions are used. Prediction is done over 7 classes of Tomato leaf images for a dataset of 7000 images.

Testing the model:
The saved model was checked with multiple test images to check the accuracy of the predictions.

Deployment of model over a Web Api:
The saved model was also deployed over a Web Api using Flask, as an optional front- end for use. The data transfer between the interfaces is in a \ac{JSON} format.

3.5 System Maintenance

Maintaining a software system for a tomato disease identification project:
1) Regularly back up your data: Make sure to back up all data and code regularly to avoid data loss in case of system failure or errors. Store the backups in secure locations, preferably on a separate physical device.
2) Keep your software up-to-date: Make sure that your software components are updated regularly, including

your operating system, web server, and any third-party libraries that your project relies on.

3) Monitor your system: Monitor your system's performance and usage to identify potential issues before they cause problems. Use tools like monitoring software, logs, and alerts to stay informed of any problems or issues that arise.
4) Address security concerns: Be vigilant about security risks and take proactive measures to address them. Use encryption and authentication protocols to protect sensitive data and ensure that access to the system is secure.
5) Test and debug regularly: Test and debug your system regularly to identify and fix bugs and issues. Use automated testing tools to test your system's functionality, performance, and security.
6) Document your system: Keep your system documentation up-to-date to make it easier for other developers to work on your system or to troubleshoot issues that may arise.
7) Use version control: Use version control tools like Git to manage changes to your codebase and to ensure that you can roll back to a previous version if necessary.
8) By following these tips, you can help ensure that your tomato disease identification project system is reliable, secure, and effective in identifying tomato diseases accurately.

4. Conclusion & Future Enhancements

4.1 Conclusion

This project proposes a CNN model to enable the detection of the disease that has affected a given leaf image. A neural network was built and trained upon the data set. The generated model was saved and tested. The model is further deployed in 2 ways as an WEB app and as a Rest Api.

Python was used to develop the model. The dataset consisted of 7 classes of images, each of size 256x256. In total, there were 7000 images in the data set.

A user can upload the image to be checked on the web app and view the predicted disease type and the suggested remedies too. Another alternative method to use the model is through the Rest Api, where the user enters the name of the image file to check, on the site and the result is displayed on the site itself. The project model can be used to aid farmers in identifying the diseases that plague their crop/leaves and lead to a timely and convenient detection process. It is beneficial for botany students and people who take gardening as a passion. These people can get real- time disease detection and remedy solutions provided to them. It saves up a lot of time and money. It is beneficial in education people about how to take care of their plants in less time possible. Calling a specialist and waiting for him to analyze and then go ahead with a remedy can sometimes be too late and lead to major loss of crops.

4.2 Future Enhancement

There can be a lot of future enhancements in this project that can help us serve our purpose better. Some of them can be:

1) In terms of neural network, modifications can be done by adjusting with different optimizers and loss functions. An increase in the number of layers can also be done.
2) The scope of the number of diseases can be increased for a more expanded view on the subject by training the algorithm to accommodate a wide range of food and cash crops.
3) A different architecture can also be used for implementing neural network such as Artificial Neural Network.
4) More machine learning algorithms can be used to increase the accuracy of disease detection.
5) An android application can also be created, which caters to a large user base.

References

[1] Mrunalini R. Badnakhe and Prashant R. Deshmukh, “An Application of K-Means Clustering and Artificial Intelligence in Pattern Recognition for Crop Diseases”, pp. 135-137, 2011.
[2] Manisha Bhangea and H.A Hingoliwala, “Smart Farming: Pomegranate Disease Detection Using Image Processing”, 58 pp. 280 – 288, 2015.
[3] Jagadeesh D. Pujari, Rajesh Yakkundimath and Abdulmunaf S.Byadgi, “Image Processing Based detection of Fungal Diseases in Plants”, 46 pp. 1802 – 1808, 2015.
[4] Pranjali B. Padol and Anjali A. Yadav, “SVM classifier based grape leaf disease detection”, pp. 3288-3294, 2016.
[5] ArtiN. Rathod, Bhavesh Tanawal and Vatsal Shah, “Image Processing Techniques for Detection of Leaf Disease”, pp. 397-399, 2013.
[6] Sanjay B. Patil and Dr. Shrikant K. Bodhe, “Leaf Disease Severity Measurement Using Image Processing”, pp. 297-301, 2011.
[7] Smita Naikwadi and Niket Amoda, “Advances in Image Procesing For Detection of Plant Diseases”, pp. 168-175, 2013.
[8] Anand.H.Kulkarni and Ashwin Patil R. K., “A plying image rocessing techni ue to detect plant diseases”, pp. 3661-3664, 2012.
[9] Amar Kumar Dey, Manisha Sharma and M.R.Meshram, “Image Processing Based Leaf Rot Disease, Detection of Betel Vine”, pp. 748-754, 2016.
[10] activation functions and its types which is better: https://towardsdatascience.com/
[11] activation-functions-and-its-types-which-is- better a9a5310cc8f.html
[12] image classifier using cnn https://www.geeksforgeeks.org/image-classifier-using-cnn/.html
[13] cnn architectures types https://medium.com/@RaghavPrabhu/cnn-architectures- lenet- alexnet-vgg-googlenet-and-resnet-7c81c017b848.html.

[14] flask documentation https://flask-restful.readthedocs.io/en/latest/.html
[15] fetch api documentation:https://developer.mozilla.org/enUS/docs/Web/API/Fetch- API.html optimizers cheatsheet https://ml-cheatsheet.readthedocs.io/en/latest/optimizers.
[16] adam-optimization-algorithm-for-deep-learning https://machinelearningmastery.com/adam-optimization-algorithm-for-deep- learning/
[17] train-test-split-and-cross-validation-in-python, https://towardsdatascience.com/train- test-split-and-cross-validation-in-python- 80b61beca4b6/
[18] pytorch-tutorial, https://www.analyticsvidhya.com/blog/2018/02/pytorch-tutorial
[19] pytorch-tutorial-distilled, https://towardsdatascience.com/pytorch-tutorial-distilled- 95ce8781a89c.
[20] designing-a-restful-api-with-python-and-flask, https://blog.miguelgrinberg.com/post/designing-a-restful-api-with-python-and- flask.
[21] Designing a restful api in flask, http://blog.luisrei.com/articles/flaskrest.html.
[22] Conversion to model, https://developer.apple.com/documentation/ coreml/converting/trained/models/to/core/ml.
[23] Core-ML, https://www.google.com/search?q=core+mloq=core+mlaqs=chrome...69i57j0 l5.2175j0j4sourceid=chromeie=UTF-8.
[24] Vision, https://developer.apple.com/documentation/vision
[25] Pytorch to coreML conversion, https://medium.com/@alexiscreuzot/building-a-neural- style-transfer-app-on-ios-with-pytorch-and-coreml-76e00cd14b28
[26] Modeltesting in Xcode, https://www.appcoda.com/create-ml/
[27] Machine Learning in iOS, https://developer.apple.com/machine-learning/
[28] CoreML in your app, https://developer.apple.com/documentation/coreml/integrating-a- core-ml-model- into-your-app
[29] CoreML in iOS 11, https://www.raywenderlich.com/ 577-core-ml-and-vision- machine- learning-in-ios-11-tutorial
[30] Machine Learning in iOS 11, https://www.raywenderlich.com/ 577-core-ml-and-vision- machine-learning-in-ios-11-tutorial