Tomato Plant Disease Detection

Aanish. P

Student
B-Tech Computer Science
Bennett University
Greater Noida, Uttar Pradesh-201310
ap1634@bennett.edu.in

Anirudh Barthwal

Student
B-Tech Computer Science
Bennett University
Greater Noida, Uttar Pradesh-201310
ab1634@bennett.edu.in

Maneesh Athi

Student
B-Tech Computer Science
Bennett University
Greater Noida, Uttar Pradesh-201310
am3759@bennett.edu.in

Monish Reddy

Student
B-Tech Computer Science
Bennett University
Greater Noida, Uttar Pradesh-201310
ar4289@bennett.edu.in

***Abstract*— Plant diseases are one of the major causes of crop failure and low yield [1]. As they can spread for a single plant to an entire crop, hence it is key to detect and stop the spread of such diseases. We propose an approach of detecting plant diseases from RGB leaf images. As most of these diseases are visible on the surface of the leaf of the plant as patches, molds etc. the experiment is conducted on tomato plant as due to its large magnitude production and similarity with other plant leaves. The system consists of 2 modules, one is the deep learning system, which is used to classify leaf images based on the disease and the other is an image processing model that is used to extract leaf images from the background. The model would classify 9 disease classes and 1 healthy class. A comparative approach is taken to find the best accuracy. Mainly CNN and transfer learning models are tested in comparison. The transfer learning models, with pretrained weights have performed the best, and specifically EfficientNetB0 have resulted in the best testing accuracy of 97.49%. To segregate the leaf images different methods of watershed, graph cut, grub cut and thresholding is used in a comparative study to infer with the best technique to segregate plant leaves from a RGB images which contain a wide range of background from plants and trees to aerial images. As per the study a module with a combination of grub cut and thresholding give the best result in identifying leaf images from non-uniform background and blue channel thresholding gives the best results in masking of leaf images with a uniform background.**

Keywords—CNN, Plant Disease Detection , Image Processing ,Transfer learning

I. INTRODUCTION

In developing countries like India, agriculture has a major impact on the economy [2] and the loss in agriculture due to disease is the major concern, which could impact our economy. Modern technology leading to the green revolution has done phenomenal change to produce enough food to meet the demand of a huge population in the world that grow's at exponential rates. However, plant diseases are still one of the major cause of food scarcity and the degradation of food quality [1] . Crop diseases are important to detect at an early stage so that they don't spread to the entire farm so that we can increase the quantity and quality of food we eat. In the current scenario farmers are manually analysing each plant frequently to detect diseases which is challenging and inefficient way of doing it and in some cases even the most experienced fail to where as a computer can differentiate. Hence it was important to come up with a robust automated system to do this task for the farmer. We worked on an automated plant disease detection system. Our experiments that are discussed in this article are based on the tomato plant. As tomato is a widely cultivated vegetable crop, keeping in mind the magnitude of the crop and the similarity of the leaf structure when compared to other plants [5], we planned to implement on it. We are achieving automation with the help of Artificial intelligence. Our system consists of a deep learning model that could identify diseases with the help of RGB leaf images. We are using the plant village dataset [3]. The study contains a comparative approach of transfer learning models(with and without pretrained weights) and CNN models from scratch. These models are trained under various hyperparameters and parameters. To ease computation, transfer learning algorithms(with and without pre-trained weights) were used in comparison with CNN models from scratch to conclude with the best results. There are a couple of other methods to solve the same problem without deep learning but with image processing [9]. The technique is based on colour image processing, edge detection and some other image processing methods. The methods applied are theoretical and are restricted to experiments in a controlled domain and are highly susceptible to minor changes in the background, lighting and shear. Hence it is not a robust model. In addition to the deep learning part we also developed a robust image processing module that is used to extract leaf images from a diverse background. In a whole, it is a complete system that is deployable on a wide range of platforms from agro robots to UAV's. Our goal in the project is to come up with a computationally light system with ease in deployment which will help farmers and increase the productivity of crops hence the quantity and quality effortlessly.

II. METHODOLOGY

A. Data Set

The data set contains RGB images of tomato plant leaves taken from Plant Village [3]. The complete data set contains images from various plants such as potatoes, grapes, apples, corn, blueberry, raspberry, soybeans, squash and strawberry but we have used the tomato plant only. The reason to choose tomato is because the magnitude of tomato cultivation throughout the globe is very high [4] and the tomato plant leaf is visually and structurally identical to most other plants [5]. Hence the weighted advantage of requiring less computing power to train the model has led us to choose tomato plant instead of other plants. There are 10 classes out of which 9 classes are various disease that occur in tomato plants and one class is the healthy tomato leaf. The diseases identified by the classifier are,

- Bacterial spot
- Early blight
- Late blight
- Mold
- Septoria spots
- Two spotter spider mite
- Target spot
- Mosaic virus
- Yellow leaf curl virus

Visualising the data set in fig.1

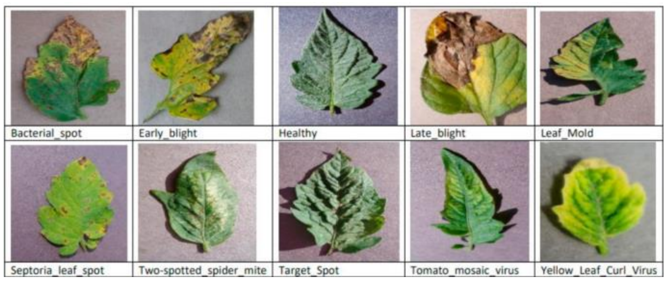


Fig. 1

The dataset contains 18159 images, the distribution varies as show in fig.2.

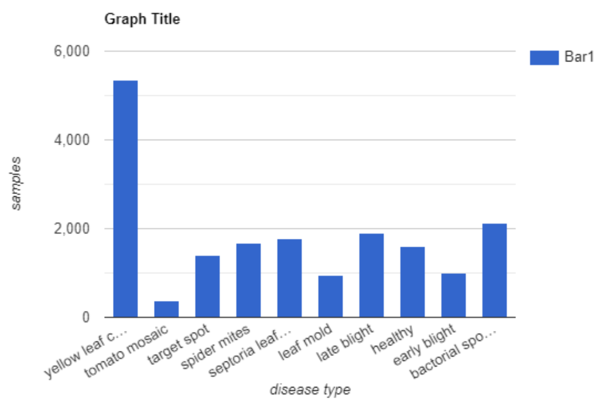


Fig. 2

As the dataset is fairly distributed evenly except for a few kinks in some disease classes and the total number of images are relatively less to train a transfer learning model, hence we would increase the data set by 20 percent with the help of image augmentation.

The data set contains a constant image size of 256 X 256 pixels and jpeg format for all classes, if there were any discrepancies in the size of the image then they were dropped, if they belonged to a populous class or else resized or shrunk to 256 X 256 dimension. However our overall motive was to increases the total trainable images and equalize the distribution.

When it comes to training CNN and transfer learning models [10], we had a random split of 30 percent testing data and 70 percent training data. The validation set was the same as testing data in our case. The total number of training samples are 14,527 and testing are 3,632. Since we are dealing with a classification problem, the ratio of individual class samples were maintained the same during the random split. Since we are following a comparative study on various algorithms, to keep it fair all the results of the trained models are trained on the same test and train data.

B. *Pre-Processing*

As we can see from fig.1 the image dataset is captured in a uniform back ground, but the back ground varies from one image to another. The background however uniform still contains the shadow of the leaf projected on the background leaving a black region. Now when we are training a deep learning model the computer assumes these inconsistences mentioned above as features as it cannot differentiate from a leaf and a background. This could decrease the overall accuracy and increases the number of miss predictions, as the some diseases are very identical to others or the healthy plant. Hence we had to mask [11] the background we all black pixels, by doing so we would nullify the effect of an uneven background while training the deep learning model, so that the model doesn’t learn anything from the mask and since all the images would have the same mask pixel value the background region would be treated as null and void by the deep learning model.

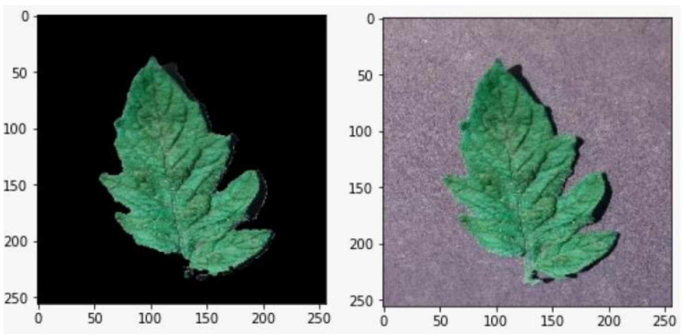


Fig. 3

For background identification, the experiments performed are discussed. First we found the edge map of the image using Cammy edge detection [6]. Once we found the edge in it the leaf was clearly differentiable from the background but we did not find a way to mask the background from the image hence the idea was dropped. Later we tried to convert the entire image into a grayscale image and by using global auto thresholding [12], trying to segregate the leaf from the background. One problem that occurred in this process is that, the back ground is relatively light whereas the leaf is relatively dark but in some of the samples there is a shadow of the leaf on the back ground which is again dark so when we tried to threshold it, the shadow that belongs to the background does not get masked. The results for the same are shown in fig.3 Finally tweaking the before process a little what we did was, we split the RGB image into separate channels [13] and visualized it. We noticed that in the blue channel the shadow was differentiated with a large margin, this was because blue has a greater tendency to present green [14] more clearly than black. Hence in the blue channel of the image the process of image thresholding was feasible. So after the image was split into the blue channel, the blue channel image was converted into a grayscale image and after applying thresholding on the later, a mask with clear boundaries was reviled but however inside the image there were a few patches of the mark, this was the patches and mold on the leaf surface and since this is import for our classification it was necessary to for us include them. This was corrected by using the function fill all holes [7]. The final image from the processes is shown in the fig4.

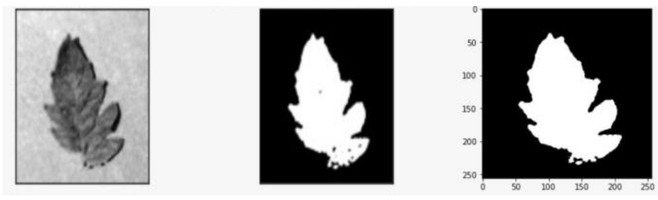


fig. 4

The sample of images are captured in a controlled environment. The background is constant, all the images have a constant scale and all images are perfectly kept on a flat surface. The data from the real world would not be the same the images could have irregular zoom, curls on edges or different sheers. Hence it is important for us replicate some of these factors and introduce some discrepancies like zoom, rotation and sheer into the sample dataset, so that the model would be robust in handling new type of samples altogether. By doing so we would resolve the bias problem and also introduce moderate amount of variance as well, thus making the model capable of handling new data. For introducing the rotation, shear and zoom factor we have used the imagedatagenerator function [8]. The constrains of the function are,

- Rotation +25 degrees to -25 degrees
- Shear range of +0.2 to -0.2
- Zoom range of +0.2 to -0.2

The imagedatagenerator function is kept constant for all the models to keep the results fair. For additional verification, the current dataset samples could be increased by sampling new samples by introducing the factors of zoom, rotation and shear. This would help increase the dataset and also provide an additional cross validation dataset.

C. *Image processing*

The image processing module is used to extract the leaf with a masked background so that we can pass it to the deep learning model. This module is different from the one described in the Preprocessing part. This module is used to extract masked leaf images from images with uneven or diverse backgrounds. The importance or use of the module is to identify leaves from images of plants or a farm like environment. This module plays a key role in the deployment of the system in a real time.

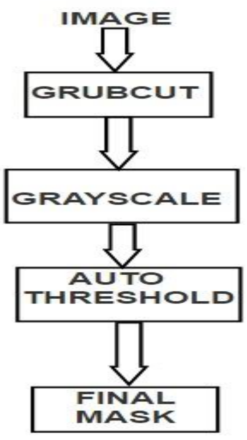


Fig. 15

The module consist of image processing techniques and the image processing work flow is as shown in fig.15. the method proposed consists of two steps one grub cut which is an inbuilt algorithm of OpenCV [15] library and a blue channel auto thresholding algorithm [12]. The input image is passed to grub cut algorithm [16] which in turn returns a mask. The mask is used to cut the original input image. Since the mask contains black pixels and while taking the sum it would affect the auto thresholding in the subsequent step so we initialized the mask value to the mean pixel to avoid this anomaly. After masking from Grub Cut the image is passed for auto thresholding[12], in auto thresholding the blue channel of the image is auto thresholded to generate the final masked image. We are having an additional thresholding function because grub cut is a running graph like algorithm [16] and the inclusion of similar objects is certain but however exclusion is not certain hence we add an additional thresholding function to remove and additional exclusions in the mask. The reason we use blue channel for thresholding is because blue channel has a grater perspective to represent the green color [14] and the patches and shadows casted due to the leaf.

D. *Training*

As we are following a comparative study, we have trained 7 different deep learning models of both CNN and transfer learning models. In CNN we trained one model from scratch without any pretrained weights and to keep the computation lite the model was trained for 30 epoch only. In transfer learning we used AlexNet [17], ResNet [18], VGG16 [19] and EfficientNet [20]. However there are many more transfer learning models but due the time constraint it was more efficient to test algorithm with better performance in the ImageNet [21] challenge. This is because in the ImageNet challenge [21] it contains a wide range of classes and it also is a classification problem. All of the models were trained on pretrained weights this is because if we were to train CNN and transfer learning models from scratch then we would need a huge training data and a lot of computation to train the model for over 500 epochs. Hence due to limited computation and data we have used pretrained weights from ImageNet [21] challenge. The following points are the analysis of each learning model that has been trained during our study,

1. CNN from Scratch

We trained a CNN model from scratch with the layers and number of neurons as shown in the fig.5 The CNN model was trained without any pre trained weights and was trained for 50 epochs. The model accepts images of dimension 256 X 256 X 3 and the summary of the model was it had 58,097,546 total parameters and 58,094,666 trainable parameters. The resultant accuracy was 93.97percent which was fairly decent but given the number of iterations and computational complexity, this model was not efficient.

Layer Type	Output shape	param
conv2d_10 (Conv2D)	(None, 256, 256, 32)	896
activation_14 (Activation)	(None, 256, 256,32)	0
batch_normalization	(None, 256, 256,32)	128
max_pooling2d_6	(None, 85, 85, 32)	0
dropout_8 (Dropout)	(None, 85, 85, 32)	0
conv2d_11 (Conv2D)	(None, 85, 85, 64)	18496
activation_15 (Activation)	(None, 85, 85, 64)	0
batch_normalization_14	(None, 85, 85, 64)	256
max_pooling2d_7(MaxPooling2	(None, 42, 42, 64)	0
dropout_9 (Dropout)	(None, 42, 42, 64)	0
conv2d_13 (Conv2D)	(None, 42, 42, 128)	73856
activation_17 (Activation)	(None, 42, 42, 128)	0
batch_normalization_15	(None, 42, 42, 128)	512
conv2d_14 (Conv2D)	(None, 42, 42, 128)	147584
activation_18 (Activation)	(None, 42, 42, 128)	0
batch_normalization_16	(None, 42, 42, 128)	512
max_pooling2d_8 (MaxPooling2	(None, 21, 21, 128)	0
dropout_10 (Dropout)	(None, 21, 21, 128)	0
flatten_2 (Flatten)	(None, 56448)	0
dense_4 (Dense)	(None, 1024)	57803776
activation_19 (Activation)	(None, 1024)	0
batch_normalization_17	(None, 1024)	4096
dropout_11 (Dropout)	(None, 1024)	0
dense_5 (Dense)	(None, 10)	10250
activation_20 (Activation)	(None, 10)	0
Total params: 58,097,546		
Trainable params: 58,094,666		
Non-trainable params: 2,880		

Fig. 5

2. ResNet

We trained 2 models with ResNet [18] convolution. One we used ResNet [18] with pretrained weights for ImageNet [21] challenge. In the first step for the convolution we used ResNet [18] and the resultant passed through a flattening layer and a dropout to avoid overfitting and the resultant of that is passed through a fully connected layer which in turn returns the output from a softmax layer of 10 nodes. The summary of the model is represented in fig.7 . This model took input of images of dimension 256 x 256 x 3 and ran for 30 epochs. The resultant accuracy was 97.02 percent. Second we used a pretrained weights for ImageNet challenge [21]. In the first step for the convolution we used ResNet [18] and the resultant passed through a flattening layer and a dropout to avoid overfitting and the resultant is directly passed to the softmax layer with 10 nodes to output the results of the classification. This was done to see the learning provided by the fully connected layer. This resulted in accuracy of 95.73. From this we could say that convolution alone is not sufficient to conclude with the result and hence its required to have an additional layer to infer with better results. On an overall we could say that ResNet [18] is one of the best performing models.

Layer (type)	Output Shape	Param
resnet50 (Functional)	(None, 8, 8, 2048)	23587712
dropout_9 (Dropout)	(None, 8, 8, 2048)	0
flatten_3 (Flatten)	(None, 131072)	0
dense_8 (Dense)	(None, 1024)	134218752
activation_1 (Activation)	(None, 1024)	0
batch_normalization_1 (Batch	(None, 1024)	4096
dropout_10 (Dropout)	(None, 1024)	0
dense_9 (Dense)	(None, 10)	10250
activation_2 (Activation)	(None, 10)	0
Total params:	157,820,810	
Trainable params:	135,285,770	
Non-trainable params:	22,535,040	

Fig. 6

3. AlexNet

We trained AlexNet [17] with pretrained weights from ImageNet challenge [21]. The convolution layer was contained by AlexNet [17] the output of the convolution was passed to a fully connected neural network after flattening and dropout to avoid overfitting. The summary of the models fully connected layers and convolution is stated in detailed in fig.6. The input dimensions of the model are 256 X 256 X 3 and model contains 31,235,490 total parameters and 31,214,354 trainable parameters and is run for 15 epochs. The resultant is passed through a softmax layer with 10 neurons which outputs the result of the classification. The resultant accuracy was 91.28 percent. The accuracy of the model is efficient even if its less than the CNN from scratch as the AlexNet [17] model has less number of parameters and ran on just 15 epochs.

Layer (type)	Output Shape	Param
conv2d (Conv2D)	(None, 62, 62, 96)	34944
max_pooling2d (MaxPooling2D)	(None, 31, 31, 96)	0
batch_normalization (BatchNo	(None, 31, 31, 96)	384
conv2d_1 (Conv2D)	(None, 21, 21, 256)	2973952
max_pooling2d_1 (MaxPooling2	(None, 10, 10, 256)	0
batch_normalization_1	(None, 10, 10, 256)	1024
conv2d_2 (Conv2D)	(None, 8, 8, 384)	885120
batch_normalization_2 (Batch	(None, 8, 8, 384)	1536
conv2d_3 (Conv2D)	(None, 6, 6, 384)	1327488
batch_normalization_3	(None, 6, 6, 384)	1536
conv2d_4 (Conv2D)	(None, 4, 4, 256)	884992
max_pooling2d_2 (MaxPooling2	(None, 2, 2, 256)	0
batch_normalization_4	(None, 2, 2, 256)	1024
flatten (Flatten)	(None, 1024)	0
dense (Dense)	(None, 4096)	4198400
dropout (Dropout)	(None, 4096)	0
batch_normalization_5	(Batch (None, 4096)	16384
dense_1 (Dense)	(None, 4096)	16781312
dropout_1 (Dropout)	(None, 4096)	0
batch_normalization_6 (Batch	(None, 4096)	16384
dense_2 (Dense)	(None, 1000)	4097000
dropout_2 (Dropout)	(None, 1000)	0
batch_normalization_7 (Batch	(None, 1000)	4000
dense_3 (Dense)	(None, 10)	10010
Total params:	31,235,490	
Trainable params:	31,214,354	
Non-trainable params:	21,136	

Fig. 7

4. ResNet with Logistic regression

From the previous experiment we saw that after a convolution it is necessary to pass the resultant flatten tensor into a neural network to infer with better resultants. Since neural networks are computational more expensive than other machine learning algorithms like logistic regression or linear regression. We tried and got a tensor after passing images through a ResNet [18] convolutional layer and flattening the resultant feature maps of the convolution. The resultant tensor contained 1,32,152 features. We used this tensor with 1,32,152 features and pass it into a logistic regression model for training. The time taken to train the model was less when compared to other algorithms and it resulted in an accuracy of 96.28 percent. This model is a perfect trade-off in between computational complexity and accuracy. The summary of the model is present in fig.8.

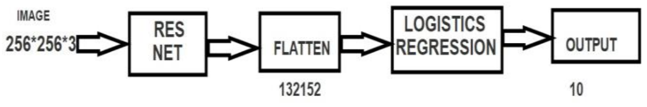


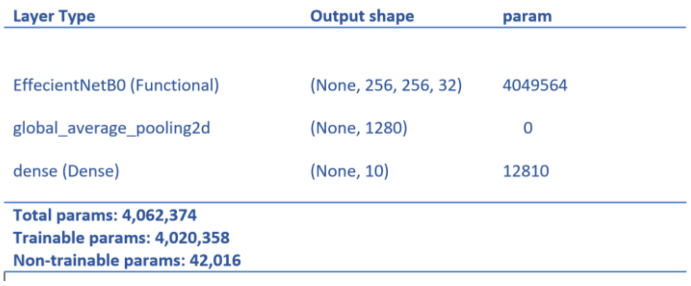
Fig. 8

5. VGG16

VGG16 [19] model was used to train the convolutional layer of the model. VGG16 [19] model used pretrained weights from ImageNet [21] challenge. It took an input image of dimensions 256 x 256 x 3 and ran for 15 epochs. The convolution contains VGG16[19] architecture and the resultant passed through a flattening layer and dropout to prevent overfitting. The resultant is passed through a fully connected neural network and the prediction is output from the softmax layer with 10 nodes. The resultant accuracy of the model is 96.17.

6. EfficientNetB0

EfficientNet [20] model was used to train the convolutional layer of the model. EfficientNet [20] model used pretrained weights from ImageNet challenge[21]. The model took an input image of dimensions 256 x 256 x 3 and ran for 15 epochs. From the EfficientNet [20] convolution the results are passed into a fully connected layer with number of neurons 100 and 50 consecutively and the end result is output from a softmax layer with 10 neurons. The summary of the model is represented in the fig.9. The resultant accuracy of the model is 97.49 percent.



Layer Type	Output shape	param
EffecientNetB0 (Functional)	(None, 256, 256, 32)	4049564
global_average_pooling2d	(None, 1280)	0
dense (Dense)	(None, 10)	12810

Total params: 4,062,374
Trainable params: 4,020,358
Non-trainable params: 42,016

Fig. 9

III. Results

In fig.10 is the comparative results of accuracy of the different algorithms that were trained and the number of parameters that it was trained on. Now analysing the number of parameters its trained on is important because it is a measure of computational complexity. Hence when we choose a model it show have the perfect trade of between computational complexity and accuracy.

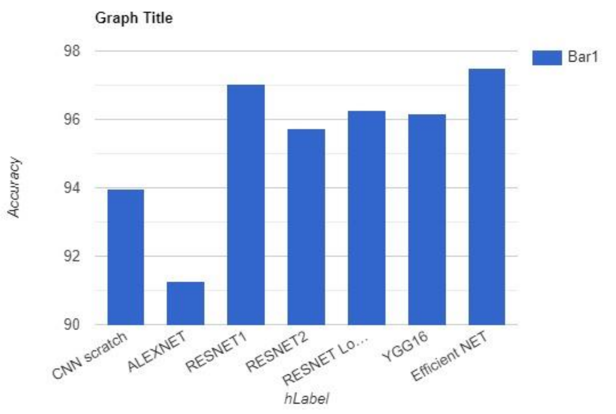


Fig. 10

As we can see from fig.10 that in our experiment the EfficientNetB0 [20] model has performed the best. The EfficientNetB0 [20] with 4,062,374 total parameters and 4,020,358 trainable parameters and has been run for 15 epochs. It has an accuracy of 97.49 percent the highest among others. However there are algorithms with lesser parameters but lack accuracy, accuracy has a higher importance with respect to number of parameters and the best model is concluded with both the factors into consideration. The hyperparameters of the final EfficientNetB0 [20] model could be visualized from fig.11.

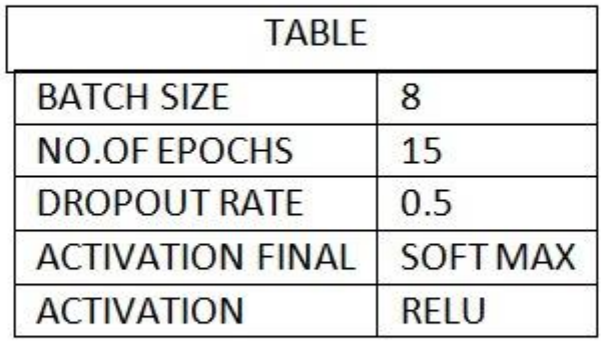
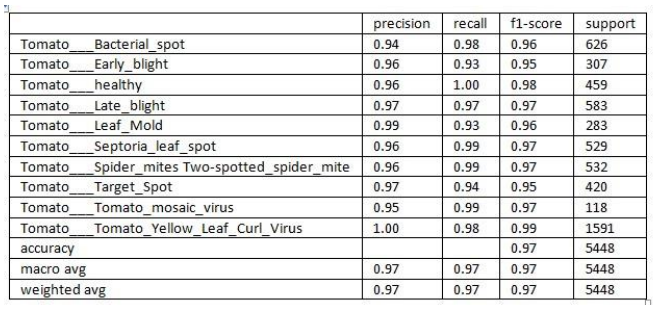


TABLE	
BATCH SIZE	8
NO.OF EPOCHS	15
DROPOUT RATE	0.5
ACTIVATION FINAL	SOFT MAX
ACTIVATION	RELU

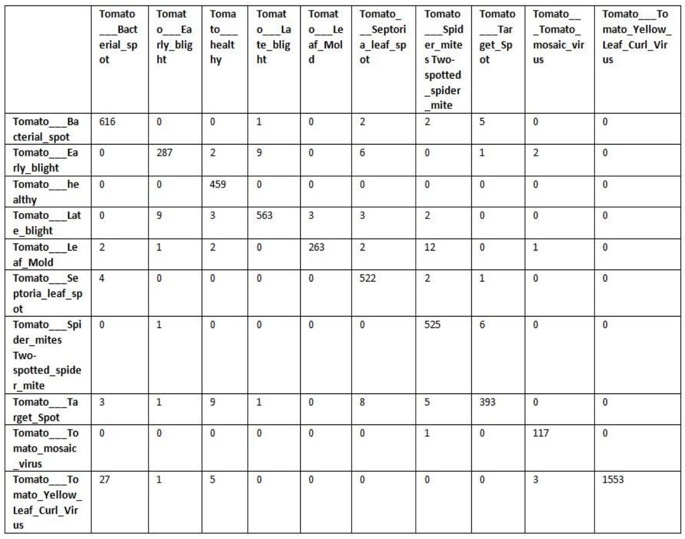
Fig. 11



	precision	recall	f1-score	support
Tomato___Bacterial_spot	0.94	0.98	0.96	626
Tomato___Early_blight	0.96	0.93	0.95	307
Tomato___healthy	0.96	1.00	0.98	459
Tomato___Late_blight	0.97	0.97	0.97	583
Tomato___Leaf_Mold	0.99	0.93	0.96	283
Tomato___Septoria_leaf_spot	0.96	0.99	0.97	529
Tomato___Spider_mites Two-spotted_spider_mite	0.96	0.99	0.97	532
Tomato___Target_Spot	0.97	0.94	0.95	420
Tomato___Tomato_mosaic_virus	0.95	0.99	0.97	118
Tomato___Tomato_Yellow_Leaf_Curl_Virus	1.00	0.98	0.99	1591
accuracy			0.97	5448
macro avg	0.97	0.97	0.97	5448
weighted avg	0.97	0.97	0.97	5448

Fig. 12

In fig.12. Is the score of the model with respect to each class. As we can see that Tomato_Yellow_Leaf_Curl_Virus is the most accurate classification with 100 percent accuracy and Tomato Bacterial spot is least accurate one with 94 percent accuracy. One very import observation we can make from the fig.13. which is the confusion matrix is that that less non healthy leaf’s is classified as healthy. This is very important in prediction as care has to be given to infected leaves and the healthy ones can be ignored, as a result if a leaf is classified as healthy even if it’s effected than there is a bigger chance of damage to the crop and hence this scenario must be avoided.



	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	616	0	0	1	0	2	2	5	0	0
Tomato___Early_blight	0	287	2	9	0	6	0	1	2	0
Tomato___healthy	0	0	459	0	0	0	0	0	0	0
Tomato___Late_blight	0	9	3	563	3	3	2	0	0	0
Tomato___Leaf_Mold	2	1	2	0	263	2	12	0	1	0
Tomato___Septoria_leaf_spot	4	0	0	0	0	522	2	1	0	0
Tomato___Spider_mites Two-spotted_spider_mite	0	1	0	0	0	0	525	6	0	0
Tomato___Target_Spot	3	1	9	1	0	8	5	393	0	0
Tomato___Tomato_mosaic_virus	0	0	0	0	0	0	1	0	117	0
Tomato___Tomato_Yellow_Leaf_Curl_Virus	27	1	5	0	0	0	0	0	3	1553

Fig. 13

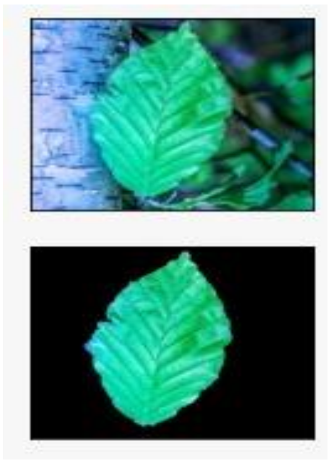


Fig. 14

Talking about the image processing component, as we can see in fig.14. that Grub cut with thresholding works best. Since this is a no quantitative analysis we have no measure of how good the algorithm is but by using human logic. We can identify the efficiency or accuracy of the model by looking at the results output images after masking. To make the conclusion we tested all the algorithms on a single image (fig.14) which is diverse and contains all the possible anomalies a model can commit then the output images of all the models were compared to see if the area of the leaf masked and the additional leaf part that was covered and left by the model. The model that masks the image with least area error is considered the best model. In this case we do not take into consideration the computational complexity because all the algorithms are alike and unlike transfer learning or CNN models it doesn’t have many features to work with, hence all the models are assumed to have the same computation time and complexity.

IV. Conclusion

As mentioned in introduction there are a couple of other ways to solve the plant disease detection and most of them are via image processing. These solutions are highly restricted to theoretical use and cannot be considered as a deployable solution because they are highly susceptible to background changes. Hence the deep learning approach is justified and is the right direction to be headed. In deep learning itself there are many approaches one could take, based on the results we can conclude that CNN works best. This is because the plant diseases have spot, molds and coloured wrinkles on its images and by using convolutions and pooling layers is the best way to identify them as the mask travels and identifies the spikes and spots as anomalies. In CNN there are again a couple of more approaches, we have streamlined them to CNN from scratch and transfer learning CNN models. We can note from the results that CNN from scratch without pretrained weights lacks in accuracy and is computationally expensive. This is because the model when not trained from pretrained weights has to start updating the values from the beginning and this is a long process. To also keep the CNN from overfitting we would also need a huge number of samples of train data to keep the model generalized. Since we lack both the computation and data samples its more efficient to use pretrained models or transfer learning to ease the computation and attain best results. Hence we could conclude that transfer learning works best when compared to CNN from scratch. Now once we get the output from the feature map and which is flattened to a tensor. This tenor is our new input features we could direct it to a softmax layer and get the output or pass it into a neural network or pass it into a machine learning model. As we can see from the results that if the tensor is directly passed into the softmax layer then the accuracy decreases, this is because additional logic or learning is required to infer results form the tensor, since the tensor contains results from max pooling layer. Over all we can summaries that the transfer learning models and CNN models are useful to extract more information or features form the image. After the feature tensor extraction we have to decide with the model we would train the tensor on. We can conclude that machine learning algorithms such as logistic regression are computationally light but not all machine learning algorithms are light for example it would take a very long time to train SVM and performance wise logistic regression is a inferior to the neural networks by a marginal amount. If we take a look at the VGG16 [19] model it has a few trainable parameters but however it has many non-trainable parameters, which means that the time taken and computational complexity would still be high. However since we strive to get the best accuracy we are going with neural networks and a final softmax layer which makes the prediction. Since we are undertaking the task of disease prediction it is very important for the system to accurately identify if a plant has a disease irrespective of the type of the disease its key to identify it because it’s critical to bring it the notice of humans to take action. We can notice from the confusion matrix that the false prediction of healthy leaves is very low and this factor is very important in the system.

In the image processing module as disused in results grub cut algorithm with blue channel thresholding works best. The combination of both takes into account of the texture and the colour component. Is leaf masking the best result is achieved when both of the components are considered. Hence the combination of both algorithms gives the best result.

References

[1] Pico, B., D ´ ´ıez, M.J., Nuez, F., 1996. Viral diseases causing the greatest economic losses to the tomato crop. ii. the tomato yellow leaf curl

[2] Mundlak, Y. (2000). Agriculture and economic growth: theory and measurement. Harvard University Press.I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[3] Hughes, D., & Salathé, M. (2015). An open access repository of images on plant health to enable the development of mobile disease diagnostics. arXiv preprint arXiv:1511.08060.R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[4] Jones Jr, J. B. (2007). Tomato plant culture: in the field, greenhouse, and home garden. CRC press..

[5] Bogdanove, A. J., Kim, J. F., Wei, Z., Kolchinsky, P., Charkowski, A. O., Conlin, A. K., ... & Beer, S. V. (1998). Homology and functional similarity of an hrp-linked pathogenicity locus, dspEF, of Erwinia amylovora and the avirulence locus avrE of Pseudomonas syringae pathovar tomato. Proceedings of the National Academy of Sciences, 95(3), 1325-1330.

[6] Wang, Z., Chi, Z., & Feng, D. (2003). Shape based leaf image retrieval. IEE Proceedings-Vision, Image and Signal Processing, 150(1), 34-43.

[7] https://answers.opencv.org/question/9863/fill-holes-of-a-binary-image/

[8] https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

[9] S. D. Khirade and A. B. Patil, "Plant Disease Detection Using Image Processing," 2015 International Conference on Computing Communication Control and Automation, Pune, 2015, pp. 768-771, doi: 10.1109/ICCUBEA.2015.153.

[10] Torrey, L., & Shavlik, J. (2010). Transfer learning. In Handbook of research on machine learning applications and trends: algorithms, methods, and techniques (pp. 242-264). IGI global.

[11] Polesel, A., Ramponi, G., & Mathews, V. J. (2000). Image enhancement via adaptive unsharp masking. IEEE transactions on image processing, 9(3), 505-510.

[12] https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html

[13] https://en.wikipedia.org/wiki/Channel_(digital_image)

[14] McCarthy, J. K., Cohen, J. G., Butcher, B., Cromer, J., Croner, E., Douglas Jr, W. R., ... & Weng, T. (1998, July). Blue channel of the Keck low-resolution imaging spectrometer. In Optical Astronomical Instrumentation (Vol. 3355, pp. 81-92). International Society for Optics and Photonics.

[15] https://opencv.org/

[16] https://docs.opencv.org/3.4/d8/d83/tutorial_py_grabcut.html

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet classification with deep convolutional neural networks. Commun. ACM 60, 6 (June 2017), 84–90.

[18] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[19] https://arxiv.org/abs/1409.1556v6

[20] Tan, M. & Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks (cite arxiv:1905.11946Comment: Published in ICML 2019)

[21] Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. (* = equal contribution) ImageNet Large Scale Visual Recognition Challenge.