

E - Agri Kit: Agricultural Aid using Deep Learning

C.SIREESHA¹(19691F00B4)

Dr. K.HEMALATHA^{2s} (Assistance professor)

Madanapalle Institute of Technology And Science

Abstract: A agricultural aid application, created and planned, to help ranchers by using Image Processing, Machine Learning and Deep Learning ideas. Our application gives elements like early recognition of plant illness, carried out utilizing different methodologies. After assessment, results showed that Convolutional Neural Network was performing better for plant illness location with a precision of 97.94% at 20 ages. It further assists the rancher with determining the climate to conclude the ideal opportunity for horticultural exercises like gathering and culling. To stay away from reoccurrence of illness because of misfortune in soil minerals, a harvest explicit manure mini-computer is consolidated which can work out how much urea, diammonium phosphate and muriate of potash expected for a given region. Since India is a multilingual country, the application has been intended to consolidate language interpretation in four dialects: Marathi, Hindi, Punjabi and English.

Key words: e-Agriculture, Deep Learning, Image Processing, Plant Disease Detection

Date of Submission: 20-05-2022

Date of acceptance: 03-06-2022

I. INTRODUCTION

According to a study by the Associated Chambers of Commerce and Industry of India, annual crops losses due to pests and diseases amount to Rs.50,000 crore (\$500 billion), which is tantamount to a country where at least 200 million go to bed hungry every night [1]. Agriculture being a vital sector has a majority of the rural population in developing countries relying on it. The sector is faced by major challenges like unprecedented pest attack and unforeseen weather conditions affecting their produce leading to major loss of food and effort. Technology plays a vital role in uplifting the livelihoods of the rural populace which can be done by using a simple agro-android application system.

Plant diseases can affect vast produce of crops posing a major menace to food security as well as leading to major losses to farmers. An extensive review of existing research was conducted by us on this domain [5] and in an effort to help farmers overcome this problem, we have designed an android application, Agricultural Aid which utilizes machine learning to provide plant disease detection. This detection is combined with an android application which provides features like weather forecast of up to 7 days, fertilizer calculator and language translation in up to 4 languages which has been implemented and integrated using Android Studio and its APIs. For disease classification, we followed two approaches: Image Processing with Machine Learning and Deep Learning models.

The first approach i.e. Image Processing approach usually includes multi-step preprocessing techniques such as: Filtering, color space conversion, thresholding and finally, contouring to mark out the infected region. These methods can be used with Machine Learning concepts to provide classification of infected regions. However, the accuracy for such methods isn't very high. As an alternative to these steps, "GrabCut" Algorithm can also be used which is an optimized method of foreground extraction to eliminate background noises using minimal user interaction [4]. It has better accuracy in terms of background elimination and can be used for better classification however for the time being this method wasn't used in the application but can be incorporated in the future to improve accuracy. For the second approach

i.e. Deep Learning approach, a deep neural architecture is used to train and test on leaf image databases to classify the disease. The paper provides a comparison of results obtained after applying Deep Learning Models such as CNN, ResNet-152 and Inception v3. In our agriculture aid, CNN Model is used to train and form an automated plant disease system based on images of leaves of both healthy and diseased plants.

In this work, several approaches were tried and tested to form an automated plant disease detection system which would later be integrated with an agricultural aid application. Section 2 describes the application and its overall working and architecture using UML diagrams. Section 3 describes the dataset used for training and testing the models and dataset parameters along with a snapshot of the images in the dataset. After which experimental methods are discussed including Image Processing Methods and Deep Learning Models which are compared and the resulting model with highest accuracy is integrated with an android application designed to

help farmers. The paper concludes with some overall discussion and future direction of research which can be done for the evolution of the application designed.

II. DATASET

A. Plant Village Dataset

Plant Village Dataset [3] is a publicly available dataset of 54,309 images of healthy and diseased leaf images for 14 crops such as tomato, potato, corn, orange, grape etc. The images are taken such that the leaf is placed on top of a neutral background before clicking their picture. Each leaf image has 38 labels assigned to them which is a crop-disease pair as shown in Figure 1.

1. Due to GPU specifications and hardware limitations, for our application we have used 6 crops namely: Tomato, Potato, Orange, Corn and Cotton. The detailed parameters of the dataset can be described by Table 1.



Fig. 1. Example of the leaf images in Plant Village Dataset, labelled as: (1) Apple Scab, *Venturia inaequalis* (2) Apple Black Rot, *Botryosphaeria obtusa* (3) Apple Cedar Rust, *Gymnosporangium juniperi-virginianae* (4) Apple healthy (5) Blueberry healthy (6) Cherry healthy (7) Cherry Powdery Mildew, *Podosphaera clandestine* (8) Corn Gray Leaf Spot, *Cercospora zeae-maydis* (9) Corn Common Rust, *Puccinia sorghi* (10) Corn healthy (11) Corn Northern Leaf Blight, *Exserohilum turcicum* (12) Grape Black Rot, *Guignardia bidwellii*, (13) Grape Black Measles (Esca), *Phaeomoniella aleophilum*, *Phaeomoniella chlamydospora* (14) Grape Healthy

B. Cotton Dataset

Images in Plant Village Dataset are captured in such a way that the leaf is extracted and placed on top of a neutral background before clicking the image. Training on such a dataset decreases the efficiency of the model to perform in real time scenarios where the leaf image may have noisy background. Hence a dataset was needed which included more real time images.

To include more real time images having variable orientation, background noise or other leaves present in the background, we worked on raw images of healthy and diseased cotton leaves taken from Shirpur and labelled by industry experts as seen in Figure 2. These images were processed to form a proper cotton dataset which was used to train our model so that it can accommodate real time images.

Due to insufficient images for other disease classes, only Healthy and leaf images with Magnesium Deficiency were considered for training our model. Dataset parameters can be described by Table 1.

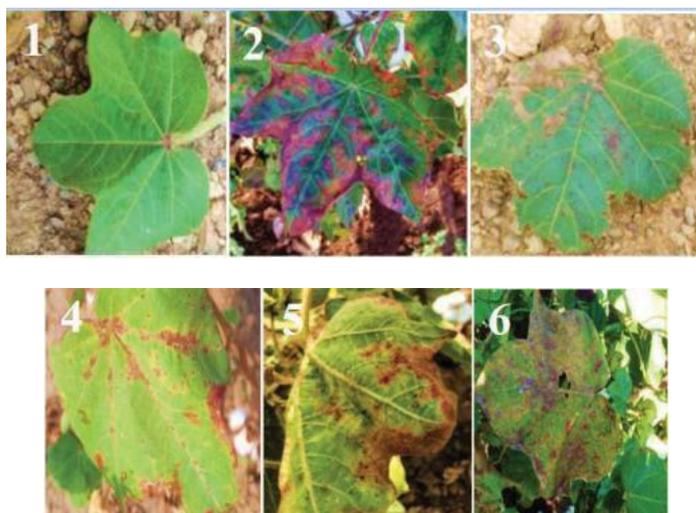


Fig. 2. Example of the leaf images in Cotton Dataset, labelled as: (1) Healthy (2) Magnesium Deficiency (3)

Alternaria Blight (4) Bacterial Blight (5) Jassid (6) Powdery Mildew

TABLE I.DATASET PARAMETERS

Parameter		Plant Village	Cotton Dataset
Size Database	of	54,309 Images	2,100 Images
Format Images	of	.jpg	.jpg
Bit Depth		24	-
Resolution		256x256 (pixels), 96x96(dpi)	Variable (mostly 506x520pixels)
Total Species	Crop	14 (Grape, Orange, Soybean, Apple, Blueberry, Cherry, Corn, Peach, Bell Pepper, Potato, Raspberry, Squash, Strawberry, Tomato)	1 (Cotton)
Types Diseases	of	Fungal (17), Bacterial (4), Mold (2), Viral (2), Mite-Affected (1)	Magnesium Deficiency, Alternaria Blight, Bacterial Blight, Jassid, Powdery Mildew
Availability		Public	On request (from Shirpur, Maharashtra, India)

III. EXPERIMENTAL METHODS

In this section, we have discussed two approaches taken to detect and classify plant diseases. Each approach is discussed in detail to include steps, type of models utilized and results.

A. Image Processing Approach

Image Processing is a popular first step in the process of plant disease detection and often includes multi-step processes to achieve processed, ROI centric input images for further classification [6][7][8][9][10]. In this approach, various Image Processing techniques were used on the input image to get a final output image which would mark the infected area and also calculate the percentage of area infected in the leaf. The major advantage observed in this approach was that this approach eliminated the need to extract the leaf and place it on a black background during image capture for the algorithm to work. In real time scenario, the infected leaf would be present amongst a cluster of mixed crops and this approach was able to segregate the infected leaf from the healthy ones in an input image.

Step 1: For the first step of the algorithm the user can click an image of the leaf he wishes to classify and input it to the algorithm through a GUI.

Step 2: Then we applied Mean Shift Filtering, a data clustering filtering algorithm where for each original pixel is replaced by a new spatial and color mean that creates a smoothing effect on the input Image which removes any background noise

Step 3: Next, contouring is performed to identify which leaf is infected and of interest by marking the closed curves for further steps in the algorithm. This step also serves as the identification of the region of interest, the diseased spots.

Step 4: Further, thresholding is implemented to show the infected parts of the leaf by converting images to binary images such that any pixel value below threshold value and above threshold value are segmented into two parts and converted into either 0- or 255-pixel value.

Step 5: After thresholding, the infected part is masked out to get the infected part with original RGB values.

Step 6: This second contouring is done to identify and mark out the infected part in the leaf which is contoured and selected in the first contouring. After executing all the above steps, the result is observed as illustrated by Table 2.

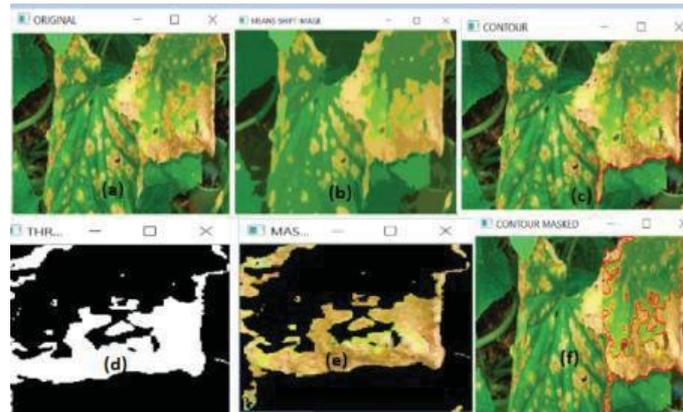


Fig. 3. a) Input Image b) Mean shift Filter c) Contouring d) Thresholding e)Masking f) Final Contouring

TABLE II. IMAGE PROCESSING RESULT:

	
Input	Output
Perimeter	589.99
Total Area	15707.00
Infected Area	4200.50
Percentage Infected Area (%)	26.74

1) GrabCut Algorithm

GrabCut is an image processing algorithm designed by Carsten Rother, Vladimir Kolmogorov & Andrew Blake from Microsoft Research Cambridge, UK [4]. It provides a more optimized algorithm for foreground extraction (leaf image) while discarding noisy background in the process. The reason for choosing grabcut algorithm is that it is more optimized and thus gives better background elimination results than the previous multistep approach. The working of the algorithm can be explained as following:

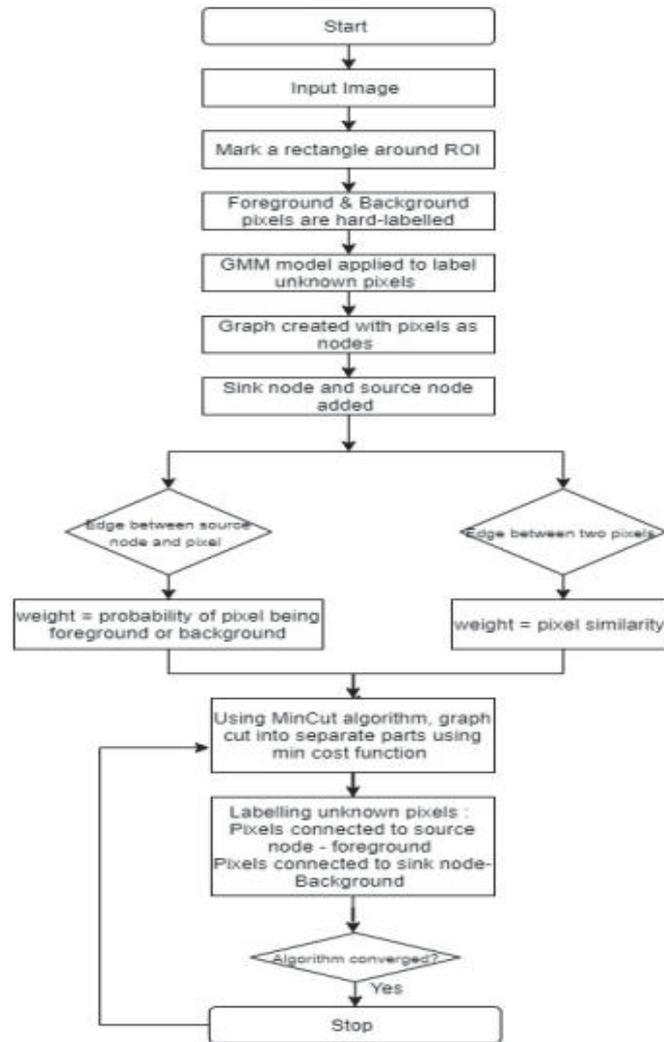


Fig. 4. Flowchart for GrabCut algorithm

- 1) After selecting the image for extraction, the user is required to mark out a rectangle selecting the region/object of interest (ROI) where everything outside the rectangle is taken as background.
- 2) The computer labels the pixels as “foreground” and “background” pixels which is also known as hard-labelling.
- 3) After labelling, a Gaussian Mixture Model (GMM) is applied on the foreground and background. GMM learns and tries to label unknown pixels inside the rectangle as either probably foreground or background according to hard-labelled pixels and color statistic which is similar to clustering.
- 4) Using this pixel distribution, a graph is created where pixels are denoted as nodes. Additionally, 2 nodes are added, namely: Source node and Sink node such that every foreground pixel is connected to source node and every background pixel is connected to sink node.
- 5) The weight of the edges connecting pixels to source node is calculated as the probability of the pixel being a part of the foreground or the background.
- 6) The weight of the edges connecting two pixels is defined by pixel similarity such that if there is a small difference in pixel color the weight will be high.
- 7) Using a min-cut algorithm on the graph, the graph is cut into separate source node and sink node using minimum cost function where the cost function is sum of all the edges that are cut.
- 8) After the cut, nodes (pixels) connected to source node are labelled as foreground and nodes (pixels) connected to sink node becomes background.
- 9) This algorithm is applied till the classification converges and we get the final leaf image as shown in Figure 5.

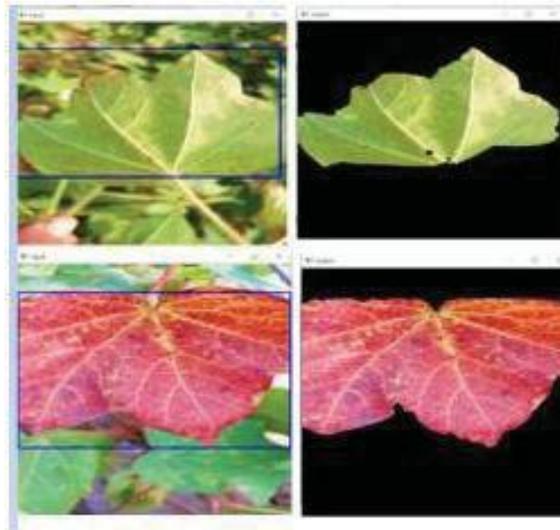


Fig. 5. a) Grab Cut Algorithm on healthy Cotton leaf b) Grab cut algorithm on Magnesium deficiency leaf

GrabCut algorithm provides an excellent method for background elimination as a preliminary preprocessing step for classification of disease. Automating the selection of the ROI using the rectangle instead of a user input can increase the efficiency and will allow background elimination on datasets with large number of leaf images

B. Deep Learning Approach

1) CNN Model

In the Deep Learning [13][14] Approach, we decided to take a subset of the PlantVillage dataset along with the cotton dataset to train and test the CNN model. The input image is fed into this model, which initially took a portion of the PlantVillage and Cotton Dataset, with a training - validation split of 70-30, therefore getting 4200 images for training and 1800 images for validation.

A CNN (Convolutional Neural Network) is a deep learning model that takes inputs which are assigned weights depending on various features [11][15]. CNN is a widely used neural network for image-based datasets.

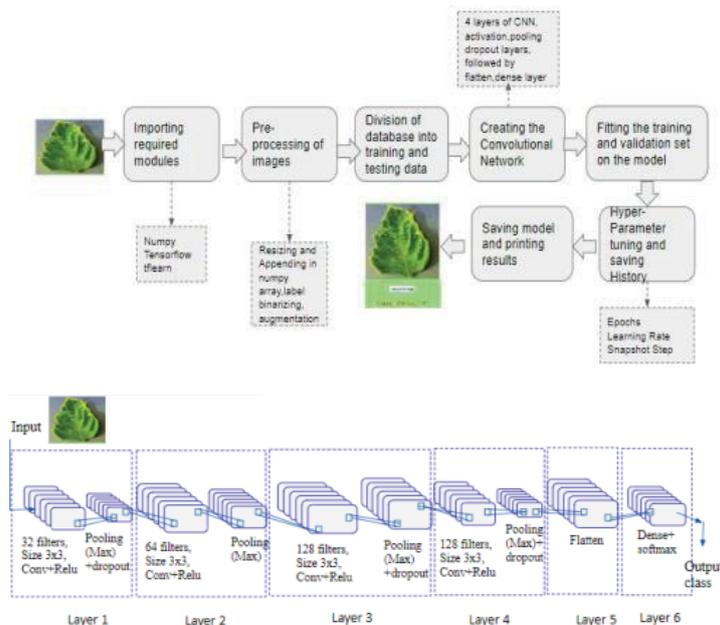


Fig. 6. Proposed CNN Architecture

Our CNN model consists of 4 main convolutional layers with 32, 64, 128, 128 filters consecutively, each followed by a ReLU activation function, max pooling and dropout layer as seen in Figure 6. This set of convolutional layers is followed by a flatten and then a dense layer which is finally followed by the softmax activation function that tells us which class has the maximum probability.

ReLU [12] is defined by the equation:

$$y = \max(0, x) \quad (1)$$

And is the most commonly used activation function. It is a piecewise linear function which makes it a real valued function composed of straight-lined sections.

Softmax Activation function returns a probability distribution over the target classes in a multiclass classification problem (like in our case). Softmax activation function forces the values of output neurons to take values between zero and one, so they can represent probability scores.

The parameters used in our model are as follows:

- Dropout Rate: Dropout rate is a regularization technique that is used to reduce overfitting data. Our model has a dropout rate of 0.25 which means that 25% of the neurons is dropped.
- Adam Optimizer: Adam Optimizer is used to minimize the loss due to stochastic gradient descent. Adam optimizer has these parameters: Learning Rate, Beta 1, Beta 2 and amsgrad.
- Learning Rate: The amount by which the weights are updated during training is called learning rate. The learning rate used by the regression function is 0.001.

Loss Function: Loss function is used to calculate loss Binary Cross entropy after binarizing the labels. The function is given as:

$$CE = -(y \log(p) + (1 - y) \log(1 - p)) \quad (2)$$

where,

CE = Cross Entropy,

y = binary indicator (0 or 1) of class label.

2) Transfer Learning – ResNet

ResNet follows VGG’s full 3×3 convolutional layer design. The residual block has two 3×3 convolutional layers with the same number of output channels. Each convolutional layer is followed by a batch normalization layer and a ReLU activation function. With the help of residual blocks, we can increase the number of hidden layers as much as required without worrying about the vanishing exploding gradients problem.

In ResNet, the module creates a direct path between the input and output to the module implying an identity mapping and the added layer just needs to learn the features on top of already available input. Since the added layer is learning only the residual, the whole module is called the residual module.

3) Transfer Learning – Inception v3

Inception-v3 is a convolutional neural network that is 48 layers deep. We can load a pre-trained version of the network trained on more than a million images from the ImageNet database and can also classify new images. Instead of choosing between tasks like convolution, max pooling and sub tasks like convolution with 1x1, 3x3 or 5x5 filter size, Inception model allows it perform these tasks together and stacks the output as shown. Inception v3 consists of two parts - The first part consists of feature extraction with a convolutional neural network while the second part consists of classification with fully-connected and softmax layers.

4) Reasons for choosing CNN

TABLE III. COMPARISON BETWEEN CNN, RESNET AND INCEPTIONV3

Feature	CNN	ResNet	Inception v3
Number of Layers	4 (Conv layers)	50	48
Number of Epochs	20	20	5
Accuracy	97.94%	87%	81.28%
Loss	9.6% (Binary Cross Entropy)	48.2% (Categorical Cross Entropy)	70.4% (Categorical Cross Entropy)

Table 3 shows the comparison between CNN, ResNet and Inceptionv3 based on the number of layers, epochs, accuracy and loss obtained. The models were implemented on Google Colab and due to resource limitations, only a certain number of epochs could be executed. Inceptionv3 usually gives a good accuracy but is computationally demanding, hence only 5 epochs were executed. As we can see from the table, our CNN model gives the best accuracy and least loss out of the 3 models with 20 epochs, hence we decided to integrate it with the application.

C. Android Integration

Figure 7 provides an overview of our application. The model of our application has three parts: user, server and the application code that connects the user and the server.

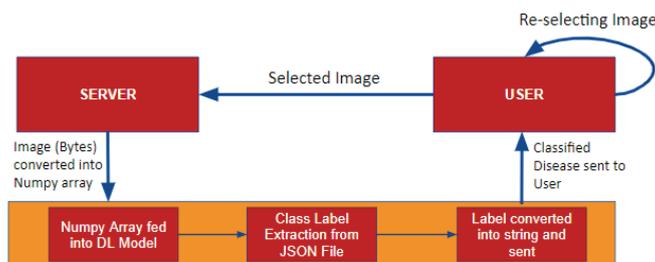


Fig. 7. Application architecture

- 1) The user takes images or selects previously existing images to upload to the image through our application. The application encodes the selected image in a string of base64 characters and sends it to the server.
- 2) The server decodes the image, reshapes it, converts it into a numpy array, loads our model and feeds the array into the model. The class label is then extracted from a JSON file that holds all class names uploaded to the server by matching the class numbers from the model. The class label is packed into a string variable and sent back to the application to be displayed to the user. The client server model here is deployed using Retrofit and Flask modules.

The Retrofit and Flask combination help in the easy integration of the application that is written in Java, and the deep learning model, which is coded in Python. The client and server APIs use retrofit instances to call the IP addresses inside the application.

We have used two Retrofit instances: one to reference our local server and the other to call the OpenWeatherMap API. The server API collects server responses and displays them to the user.

IV. RESULT AND DISCUSSION

A. Image Processing Approach

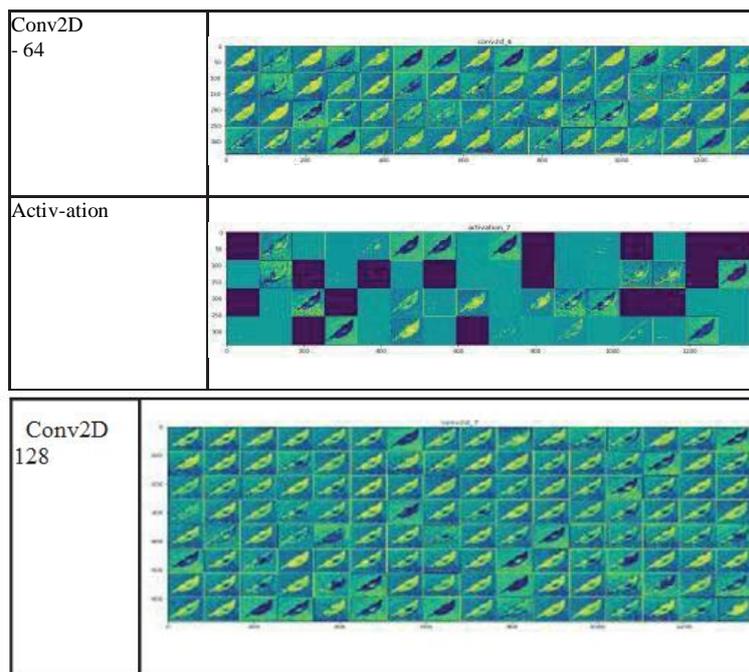
This approach works best as a preprocessing technique and is able to identify only the visible disease spots. For earlier diagnosis, using more sophisticated methods like deep learning approach would be beneficial where image processing techniques can help with the preprocessing like removing background noise and segmentation.

B. CNN Results

After fitting into the CNN model, the outputs of the intermediate layers for a given input leaf image, were visualized. Some of them are shown in table 4:

TABLE IV. VISUALIZATION OF CNN LAYERS

Layer	Visualization
Conv2D - 32 Activation	
Max Pooling Dropout	



Figures 8 shows the training, validation accuracy and loss respectively for the dataset, after it is applied to the CNN model. The trends shown in the figures are quite satisfactory as both training and validation loss is decreasing over the number of epochs while the accuracies of both the sets are increasing. The increase is somewhat uniform, however, there are fluctuations, which are not desirable. Despite the resource limitations, we got a good validation accuracy.



Fig. 8. Training, Validation Accuracy and Loss

C. Android Integration

The steps of the application are as follows:

- 1) *Starting up the application:* The application starts up with the display of the logo and the title.
- 2) *Choice of language:* The application provides a choice of language as we recognize the needs of farmers, the majority of who cannot read or converse in English. The Indian languages the app offers are Hindi, Punjabi and Marathi.
- 3) *Instructions:* The instructions provide a clear picture to the farmers on how to operate the application. There are 4 basic steps: taking a picture or selecting one from the gallery, getting solutions and using other agro solutions, such as the fertilizer calculator and the weather forecast.
- 4) *Login:* The Login screen requires the user to enter their phone number and password. The password cannot be less than 5 characters.
- 5) *Home Screen:* The home screen consists of the basic functionalities the application has to offer. It allows the user to either start the process of disease detection or choose among the calculator and the weather options.



Fig. 9. Application Screens

6) *Image selection:* The image to be selected for disease detection can be either taken on the spot or from the user’s camera roll or gallery. Both options are listed to the users using a dialog box.

1. If the “Take Photo” option is selected, the user can take a photo, confirm it or retake the photo as many times as they may like. Once chosen, the photo becomes the new thumbnail for the users to understand what photo they have chosen.

2. If the “Choose from Gallery” option is chosen, the user can scroll in his gallery and choose the required image. Here too, the thumbnail displays the selected image.

7) *Uploading and getting results:* Once the photo is selected, the user can click on the “Upload Image” button to upload it to the server. The server sends a “Upload Done!” toast on successful uploading to the server. The disease is then displayed and the users can choose to see symptoms, causes and other details by clicking on buttons.

8) *Weather:* The application provides a 7-day weather forecast to its users. It provides information on the wind, cloudiness, humidity, pressure and the time of sunrise and sunset. There are also graphs of the past week’s temperature, wind, rain and snow. The app initially takes the user’s location to accurately extract values for the above mentioned, the user also has the option to search for a specific location.

9) *Fertilizer Calculator:* The fertilizer calculator functionality of the application allows users to choose the type of crop they are seeking advice for and insert the number of hectares their plot is of. The calculator then provides the user with the DAP, MOP and Urea values the crop requires in kgs.

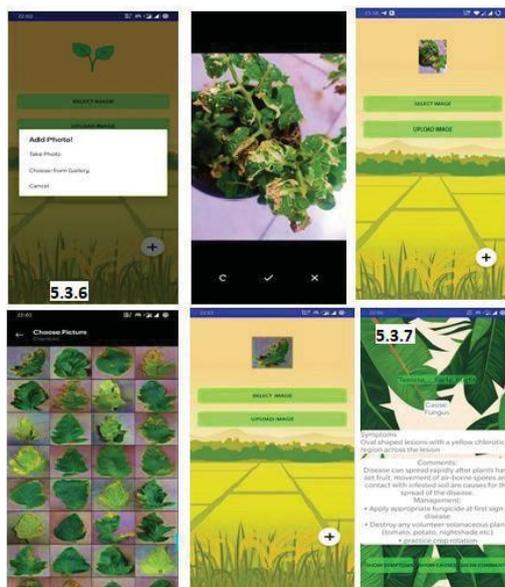


Fig. 10. Plant Disease Detection



Fig. 11. Additional features

V. CONCLUSION AND FUTURE WORK

During our analysis, we have understood the need for efficient plant disease identification & classification algorithms and prevention methods. Due to a large number of crops and diseases available, it is crucial that the detection system should be able to adapt to the changing variables and trends. Hence, Machine learning and Deep learning approaches were employed for this project which ensures that the code trains itself against as many possible numbers of different crops and diseases as possible.

The paper consists of an android application covering plant disease detection and other functionalities such as language translation, weather forecasting and fertilizer calculator. With this application we aim to provide aid in the unprecedented agricultural activities and ensure a healthy plant. Through our thorough literature review and robust implementation, we have tried several approaches as discussed above and chosen the best model- CNN with accuracy of 97.94 using 20 epochs. We have also tested our application on cotton dataset and performed realtime analysis on a diseased tomato crop to ensure our model does not overfit and

performs well in a live environment.

In future, we aim to expand our dataset to include more varied types of crops and disease so that the algorithm can adapt better to real time conditions and provide wide coverage.

ACKNOWLEDGMENT

We acknowledge the assistance of Mr. Prashant Udawant (NMIMS University) for providing the cotton dataset images used in this work.

REFERENCES

- [1]. CropLife International (May 2015). India's farmers fighting pests.
- [2]. Retrieved from: <https://croplife.org/news/keeping-indias-pests-in-line/>
- [3]. Economic Times (Sept 2018). India sets record farm output target for 2018-19. Retrieved from: <https://economictimes.indiatimes.com/news/economy/agriculture/india-sets-record-farm-output-target-for-2018-19/articleshow/65858058.cms>
- [4]. Sharada P. Mohanty David P. Hughes and Marcel Salathé. "Using Deep Learning for Image-Based Plant Disease Detection." *Front. Plant Sci.*, 22 September 2016
- [5]. Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. 2004. "GrabCut": interactive foreground extraction using iterated graph cuts. In *ACM SIGGRAPH 2004 Papers (SIGGRAPH '04)*. Association for Computing Machinery, New York, NY, USA, 309–314.
- [6]. R. Chapaneri, M. Desai, A. Goyal, S. Ghose and S. Das, "Plant Disease Detection: A Comprehensive Survey," 2020 3rd International Conference on Communication System, Computing and IT Applications (CSCITA), Mumbai, India, 2020, pp. 220-225, doi: 10.1109/CSCITA47329.2020.9137779.
- [7]. Raghavendra, B. K. (2019, March). Diseases Detection of Various Plant Leaf Using Image Processing Techniques: A Review. In 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), (pp. 313-316). IEEE.
- [8]. Malathi, M., Aruli, K., Nizar, S. M., & Selvaraj, A. S. (2015). A Survey on Plant Leaf Disease Detection Using Image Processing Techniques. *International Research Journal of Engineering and Technology (IRJET)*, 2(09)
- [9]. Kaur, S., Pandey, S., & Goel, S. (2018). Semi-automatic leaf disease detection and classification system for soybean culture. *IET Image Processing*, 12(6), 1038-1048.
- [10]. Patil, S., & Chandavale, A. (2015). A survey on methods of plant disease detection. *International journal of Science and Research (IJSR)*, 4(2), 1392-1396.
- [11]. Rathod, A. N., Tanawala, B. A., & Shah, V. H. (2014). Leaf disease detection using image processing and neural network. *International Journal of Advance Engineering and Research Development (IJAERD)*, 1(6).
- [12]. "Computational Vision and Bio-Inspired Computing", Springer Science and Business, Media LLC, 2020.
- [13]. Agarap, A.F. (2018), "Deep Learning using Rectified Linear Units (ReLU)," ArXiv, abs/1803.08375.
- [14]. Glorot, Xavier & Bordes, Antoine & Bengio, Y. (2010), "Deep Sparse Rectifier Neural Networks," *Journal of Machine Learning Research*. 15.
- [15]. 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.
- [16]. Szegedy, Christian et al. "Going deeper with convolutions." 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015): 1-9.