A Deep Learning Approach to Classify Plant and Detect Disease

[1]Bhavesh Lohana, [2]Nikhil Gangaramani, [3]Tanish Sahijwani,, [4]Aakash Chauhan, [5]Vidya Zope
[1, 2, 3, 4] Student, [5] Assistant Professor
Department of Computer Engineering,
Vivekanand Education Society's Institute of Technology, Chembur, Mumbai-400074, India

***Abstract*:** In the plant production industry plant growth monitoring and plant protection are some of the most important elements. The productivity and quality of plants are influenced by these factors. One of the major factors that deteriorate plant health is diseases. The use of computer vision techniques, machine learning algorithms, and deep learning models have gained value due to their capability of dealing with complex data with accuracy. These techniques are very widely used for pattern recognition and classification problems. Therefore in this work, a multilayer convolutional neural network is proposed for the classification of the plant species and diseased plant leaf images. The main objective of our project is to build an online application that identifies the plant species from leaf image and gives a description of that plant. And it detects that the plant has any disease, simultaneously.

Index Terms: **Computational Neural Network, Deep Learning, Machine Learning, Image Processing, Plant Classification, Disease Detection.**

I. INTRODUCTION

As India's agriculture advocates an important role in economic growth. And as there are a large number of cultivation fields. There are more than 3,91,000 plant species in the world. With such little variation, it has become harder to identify the correct plant species for the human eye. Plant pathogens consist of fungi, organisms, bacteria, viruses, etc. Therefore, plant disease detection and classification is a vital and urgent task. While humans have traditionally relied on expert knowledge to achieve this, the presence of trained experts in fields is hard to achieve and at the same time, those experts cannot give consistent diagnosis due to variabilities in knowledge, plant conditions etc. In this paper, we present an application that works on a Deep Learning model for better speed and accuracy. Accuracy and speed are the two main factors that will decide the eminence for our model. The model will help farmers and botanists to correctly classify a plant and detect the disease and alert the user before it starts spreading.

With the advances in the field of Deep Learning, there have been numerous developments of sophisticated frameworks which are capable of predicting accurately even from the noisiest of data. Our model was trained on a database with over 70,000 images with 14 plant species and 26 diseases.

II. LACUNA IN THE EXISTING SYSTEM

A. Standalone Products

Existing systems in the market do not provide a product which can classify plant diseases and detect plant diseases together. Our product reduces the user's hassle to keep track of individual applications for classification and detection.

B. Cross-Platform competency

Existing systems are not available for multiple operating systems and are platform dependent.

C. User Data Integrity

Our product securely maintains the user's data and the predictions made using our model over a period of time.

D. Authentication issues

Our system supports authentication using phone number which results into easy login and logout.

III. LITERATURE SURVEY

The various studies related to this topic have been analysed and summarized in this section.

Gilbert Gutabaga, Hungilo Gahizi Emmanuel, Andi W. R. Emanuel in [1] talked about classification in plant disease where they used VGG 16, Inception V4, ResNet, and DenseNets, etc for plant disease classification. The study was extensively focused on detecting plant disease. They found that even with high accuracy of the model the complex architecture made the process slower. It also dropped the performance of the model. Parul Sharma, Yash Paul Singh Berwal, Wiqas Ghai in [2] study using 2-layered Convolutional Neural Network(CNN) solely focusing on the disease detection section with 94% validation accuracy.

Hongga Li, Yarong Zou, Xiaoxia Huang, Renrong Jiang, Xia Li, Xin Du, Yilan Liu in [3] where they compare Maximum Likelihood Classification(MLC) and Support Vector Machine(SVM) which focuses on plant species classification only and it was found that both of the algorithms have complex mathematics and exploits both the multi-spectral and multi-temporal information associated with the pixels.

Siddharth Singh Chouhan, Uday Pratap Singh, Ajay Kaul in [4] compare several models such as Multi-Column Convolutional Neural Network(MCNN), Particle Swarm Optimization(PSO), Radial Basis Function Neural Network(RBFNN), Support Vector Machine(SVM) concentrating on one of the sections i.e. plant disease detection. The accuracy ranges from 87% to 98%. They used the three-fold strategy to validate the performance of the model. One of the demerits that were noticed is the complex architecture of RBFNN and MCNN.

Saradhambal. G, Dhivya. R, Latha S, R.Rajesh in [5] use K-means clustering algorithm to predict the infected area of the leaves. It was found that the algorithm was easy to adapt to new algorithms and generalised to different clusters. The demerits found were that it was hard to predict the K-value and it did not work well with clusters of different size and density.

Pushkara Sharma, Pankaj Hans, Subhash Chand Gupta in [6] work on algorithms such as Logistic Regression, Support Vector Machine(SVM) and K-Nearest Neighbours on detecting plant diseases. K.B.Shobana, P.Perumal in [7] used Support Vector Machine(SVM) and Artificial Neural Network(ANN) and the data was processed under RGB to HSV shading space. The accuracy of SVM and ANN was 98% and 92% respectively. The demerit found was that the classification of plant species was done using fewer aspects only.

IV. METHODOLOGY

Fig. 1 proposes the basic flow of our Machine Learning model. All the modules of the process are elaborated in detail below.

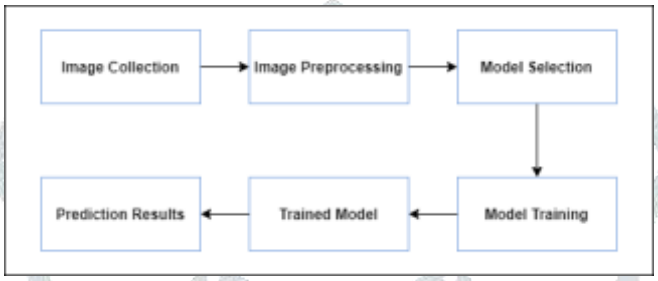

Figure 1. Flowchart of the proposed work

A. Dataset collection/Image Collection:

Firstly, the dataset was collected from Kaggle named New Plant Village Dataset by Samir Bhattarai. The dataset consists of 70,000 images with over 38 different sections inside, including both healthy and disease datasets. The dataset covers diseases like black rot, scab, common rust, bacterial spot, late blight, mould, etc. The plant species covered by the dataset are Apple, Corn, Potato, Squash, Tomato, Pepper, Grape, etc.

B. Image Preprocessing:

In this section, the dataset is divided into two sections with the training set consisting of 80% of the images and the validation set has 20% of the images of the dataset. We use Tensorflow as our primary library.

C. Model Selection:

This is the classification problem as we have to classify the plant species and the type of disease on the leaf, if any, of the plant. So, we have plenty of machine learning as well as deep learning algorithms that we can apply to this dataset.

We primarily focused on Deep Learning models such as CNN, AlexNet and ResNet.

1. AlexNet:

The AlexNet architecture consists of five convolutional blocks, three max-pooling layers, two normalization layers, two fully connected layers, and one softmax layer. The convolutional layers have convolutional filters and ReLU as a non-linear activation function. Input size of 64x64 is fixed due to the presence of fully connected convolutional layers. AlexNet has 60 million parameters.

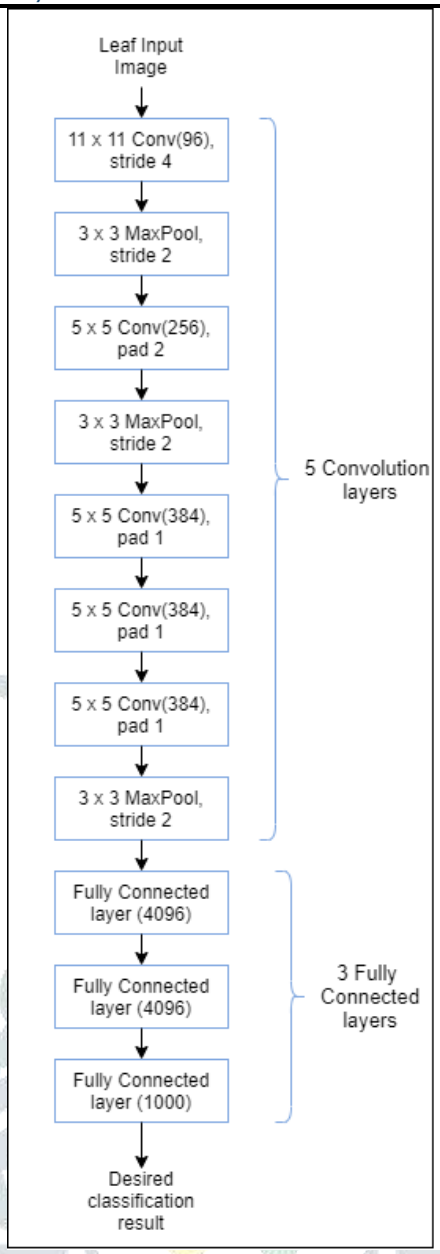

Figure 2. Alexnet's proposed architecture

As our dataset was large, AlexNet could not scale it. This was due to the gradients of the non-linear activation function ReLU are expensive to compute during the backpropagation of the error adding the non-scalability of AlexNet. The performance of the model dropped and the training time was more than the rest of the models experimented with.

The validation accuracy of 96% was seen constant throughout the training. The test accuracy was 88% for AlexNet.

2. CNN:

CNN is a multi-layered deep learning model with convolutional filters. The model consists of 3 convolutional blocks followed by 3 max-pooling layers and a flatten after which the model is dense to get the desired output. This is an important feature of the deep learning model.

However, the deeper the structure, the more complex it is to handle it. This also increases the computational overhead of the network. Deep models are also known for their requirement of a high amount of data to give superior results.

Adam has been used as an optimiser. It is a replacement optimization algorithm for stochastic gradient descent for training deep learning models.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 128)	3584
max_pooling2d (MaxPooling2D)	(None, 31, 31, 128)	0
batch_normalization (BatchNo	(None, 31, 31, 128)	512
conv2d_1 (Conv2D)	(None, 29, 29, 256)	295168
max_pooling2d_1 (MaxPooling2	(None, 14, 14, 256)	0
batch_normalization_1(Batch)	(None, 14, 14, 256)	1024
conv2d_2 (Conv2D)	(None, 12, 12, 512)	1180160
max_pooling2d_2 (MaxPooling2	(None, 6, 6, 512)	0
batch_normalization_2(Batch)	(None, 6, 6, 512)	2048
flatten (Flatten)	(None, 18432)	0
dense (Dense)	(None, 512)	9437696
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 256)	131328
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 38)	9766

Figure 3. Description of the CNN model used.

CNN had the most prominent result regarding speed as the model was trained quickly compared to AlexNet. Fig, 4 shows that validation and test accuracy of the model was found as 96% and 93% respectively.

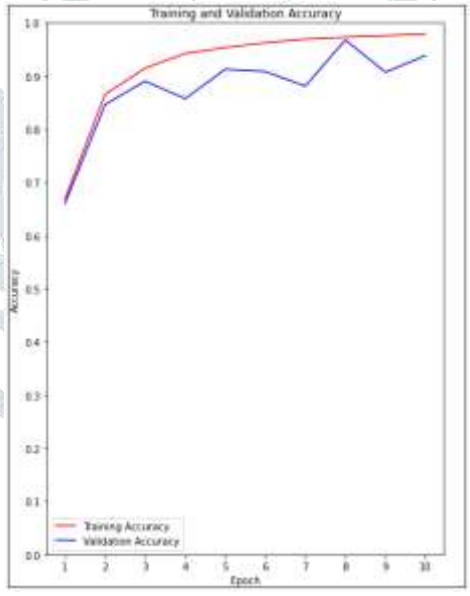

Figure 4. Training and Validation accuracy for CNN.

3. ResNet

Residual Network, ResNet uses 30-layer network architecture in which then the shortcut connection is inserted between layers which makes it different. These shortcut connections are what makes the residual network.

Fig 5. shows the detailed model with 8 convolutional layers, 4 max-pooling layers, a flatten and linear to convert the pooled features to a single column. An activation function Rectified Linear Unit(ReLu) was used to have the advantage of nonlinearity which is advantageous over linear functions such as sigmoid.

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 256, 256]	1,792
BatchNorm2d-2	[-1, 64, 256, 256]	128
ReLU-3	[-1, 64, 256, 256]	0
Conv2d-4	[-1, 128, 256, 256]	73,856
BatchNorm2d-5	[-1, 128, 256, 256]	256
ReLU-6	[-1, 128, 256, 256]	0
MaxPool2d-7	[-1, 128, 64, 64]	0
Conv2d-8	[-1, 128, 64, 64]	147,584
BatchNorm2d-9	[-1, 128, 64, 64]	256
ReLU-10	[-1, 128, 64, 64]	0
Conv2d-11	[-1, 128, 64, 64]	147,584
BatchNorm2d-12	[-1, 128, 64, 64]	256
ReLU-13	[-1, 128, 64, 64]	0
Conv2d-14	[-1, 256, 64, 64]	295,168
BatchNorm2d-15	[-1, 256, 64, 64]	512
ReLU-16	[-1, 256, 64, 64]	0
MaxPool2d-17	[-1, 256, 16, 16]	0
Conv2d-18	[-1, 512, 16, 16]	1,180,160
BatchNorm2d-19	[-1, 512, 16, 16]	1,024
ReLU-20	[-1, 512, 16, 16]	0
MaxPool2d-21	[-1, 512, 4, 4]	0
Conv2d-22	[-1, 512, 4, 4]	2,359,808
BatchNorm2d-23	[-1, 512, 4, 4]	1,024
ReLU-24	[-1, 512, 4, 4]	0
Conv2d-25	[-1, 512, 4, 4]	2,359,808
BatchNorm2d-26	[-1, 512, 4, 4]	1,024
ReLU-27	[-1, 512, 4, 4]	0
MaxPool2d-28	[-1, 512, 1, 1]	0
Flatten-29	[-1, 512]	0
Linear-30	[-1, 38]	19,494

Figure 5. ResNet model architecture

The validation accuracy was found to be 99% and test accuracy was 97%. Fig 6. shows the accuracy of the model over the iterations of the epochs.

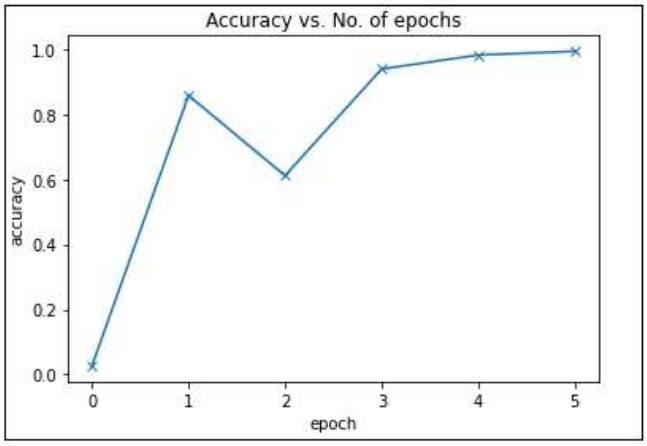

Figure 6. Accuracy vs. No. of Epochs

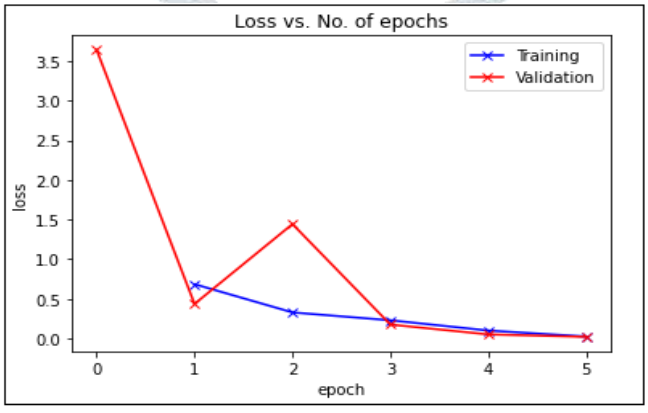

Figure 7. Training & Validation loss with respect to epochs.

In Fig. 7, the graphic representation shows the drastic decrease in the loss for validation data. In the end, both validation and training have an almost equal loss.

Although ResNet and AlexNet have good accuracy, the performance drops due to their vast architecture. Hence, we decided to use CNN as our model for the application.

TABLE I: TRAINING, VALIDATION AND TESTING ACCURACY OF THE MODELS TESTED.

	Training	Validation	Testing
CNN	93%	96%	93%
AlexNet	96%	96%	88%
ResNet	98%	99%	96%

V. PROPOSED APPLICATION ARCHITECTURE

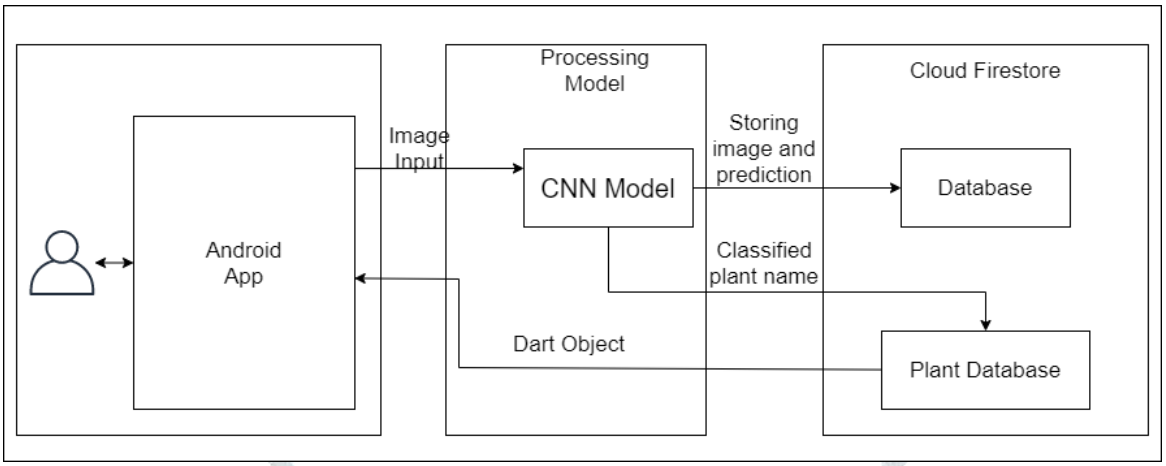

Figure 8. Architecture of the application.

The Fig.8 shows the detailed architecture of the application. The application consists of two main divisions:

- Server (Firebase)
- Client (Smart Phone)

A. Server (Firebase):

The server is hosted on Firebase. The server is the main backbone of our application. It enables the whole system to be portable and hence even supports cross platform usage.

- The data of the application is stored in Firebase. With the help of Firebase, authentication is done. Authentication is done with the help of Mobile number of users.
- All the images the user clicks or selects from the gallery or storage of the smartphone are also stored in Firebase.

B. Client (Smart Phone):

The smartphone is used to provide an interface to use our prediction model. The application is made on Flutter[8] as it supports iOS and Android both. The model is used as a .tflite file because Flutter extensively supports TensorflowLite. Flutter applications are lightweight and scalable. The client requires their phone to access all the features of the app. An internet connection is required to predict the model.

It is also needed for the plant information database. The application fetches comprehensive information about the plant from the database for the user. The software requirements are that the device needs to be an android device running android Jelly Bean, v16, 4.1.x or newer or an iOS device running iOS 8 or newer.

VI. SYSTEM DESIGN AND WORKING

The user has full access to the application. To use the application, the user has to authenticate themself using their mobile number. Phone number is a prerequisite for registering. After registration, the phone number would act as a unique identifier to store the data later on.

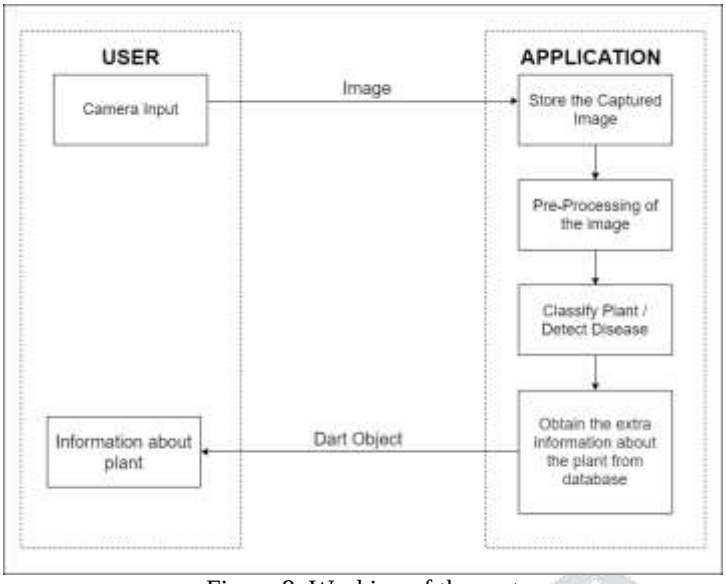

Figure 9. Working of the system.

Figure 10. User details and log out option.

After registration, on the home page there are three sections: Profile, Predictor and Prediction history. The profile section can be seen in Fig. 10. Users can watch their personal details and have the feature to log out from the app.

The predictor is backed by our Deep Learning model. There the user can have two ways shown in Fig. 11 to predict the plant:

- Using a phone camera to take the live image of the plant.
- Uploading an already saved image from Files or Gallery or Camera Roll.

Once the photo has been taken, the predictor predicts the plant species and whether it has any disease as shown in Fig. 12. If it contains any, then the plant shows both species name and the disease name that it detected. It also shows extra details about the plant that has been fetched from the plant information database stored in the Firebase.

Figure 11. Different options to use the model.

Figure 12. Plant leaf photo with its prediction.

Figure 13. Prediction history with information of the plants.

The history shows the entire history of user predictions that has been fetched from the database i.e. Firebase. It can be seen in Fig. 13. The history shows the image, basic information and whether the predictor detected any disease.

VII. CONCLUSION

The project was developed with the aim to overcome and tackle two problems i.e. Plant species classification and disease detection both at once. Our proposed Flutter application can do both and it works on all types of operating systems such as Android, iOS, etc. Our feature of authentication helps users to access their past predictions on the go. The plant information database made for this application gives extensive information about the plant that has been detected. As Firebase has been used as the application server, fetching data is quite fast. The application is open-source, light and easy to install. The application UI is minimalistic and easy to navigate.

REFERENCES

[1] Gilbert Gutabaga, Hungilo Gahizi Emmanuel, Andi W. R. Emanuel, “Image Processing Techniques for Detecting and Classification of Plant Disease – A Review”, 2019.

[2] Parul Sharma, Yash Paul Singh Berwal, Wiqas Ghai, “KrishiMitr (Farmer’s Friend): Using Machine Learning to Identify Diseases in Plants”, 2018.

[3] Hongga Li, Yarong Zou, Xiaoxia Huang, Renrong Jiang, Xia Li, Xin Du, Yilan Liu, “Projection Pursuit Learning Network Algorithm for Plant Classification”, 2017.

[4] Siddharth Singh Chouhan, Uday Pratap Singh, Ajay Kaul, “A deep learning approach for the classification of diseased plant leaf images ”, 2019.

[5] Saradhambal. G, Dhivya. R, Latha S, R.Rajesh, “PLANT DISEASE DETECTION AND ITS SOLUTION USING IMAGE CLASSIFICATION”, 2018.

[6] Pushkara Sharma, Pankaj Hans, Subhash Chand Gupta, “CLASSIFICATION OF PLANT LEAF DISEASES USING MACHINE LEARNING AND IMAGE PREPROCESSING TECHNIQUES”, 2020.

[7] K.B.Shobana, P.Perumal, “Plant Classification Using Machine Learning”, 2020.

[8] Flutter documentation:<https://flutter.dev/docs>