International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 09 Issue: 02 | Feb 2022 www.irjet.net p-ISSN: 2395-0072

Plant disease detection system using image processing

Eric Mervin Anandraj[1], Ahnaf Rehan Shah[2], Shobhit Singh[3], Tanishq Kohli[4]

[1234]*U.G. Student, Dept. of Computer Science Engineering and Technology*
Bennett University, Greater Noida, Uttar Pradesh, India

Abstract - *In India, one of the main sources of income for a large chunk of the population is agriculture. According to a survey, one of the main problems that farmers face is crops getting infected by pests, ruining months of hard work. To prevent this, we planned to implement an image processing algorithm that would detect and classify the infected part of the plant leaf, with the help of Python and Convolutional Neural Networks (CNN), in order to identify the disease and suggest a treatment. According to our results, the accuracy of the method we chose comes out to be 95.3%, which is better than the conventional approach. The CNN algorithm proposed can play a vital role in the field of smart agriculture. According to studies, human society needs to increase food production by an estimated 70% by 2050 to feed an expected population size that is predicted to be over 9 billion people. Currently, infectious diseases reduce the potential yield by an average of 40% with many farmers in the developing world experiencing yield losses as high as 100%.*

Key Words: **image processing, plant disease detection, densenet, tensorflow, keras, convolutional neural networks**

1. INTRODUCTION

The Indian economy is also highly dependent on agricultural productivity which contributes around 20% of India's GDP [1]. One of the main problems that farmers face is the crops getting infected by pests, fungi ruining months of hard work causing huge economic losses every year and threatening food security. According to a survey, annual crop losses due to plant diseases are around 220 billion dollars worldwide [2]. The conventional tools are not very useful as it is very difficult for farmers to identify the diseases by just naked eye observation, and it takes a lot of time and manual work as it requires continuous monitoring of the plant which will cost you a lot when you do it on a large scale. In countries like India, farmers do not have proper facilities, due to which this process may cost you even more. Visually detecting plant diseases is a risky task as it is less accurate and can only be done by some consulting experts. About 85% of plant diseases are caused by fungi and if a farmer can detect these diseases in the early stages and apply appropriate treatment then it can prevent a lot of waste and economic loss. We have come up with the idea of developing an application implemented with an image processing algorithm that will detect and classify the infected part of the plant leaf, with the help of Python and Convolutional Neural Networks (CNN) in order to identify the disease and suggest a treatment. We can give our application to the farmers, and they just have to take a picture of the plant and our application will tell them whether the plant is healthy or has any diseases. With the help of our application, we can analyse how agriculture in India can be made more efficient and sustainable.

2. RELATED WORK

Convolutional Neural Network (CNN) performance for object recognition and image detection has improved dramatically in recent years (Szegedy, 2015; He, 2015; Zeiler and Fergus, 2014; Simonyan and Zisserman, 2014; Krizhevsky, 2012) [3]. In the past, it was based on hand engineered features namely SURF (Bay, 2008), HoG(Dalal and Triggs, 2005), SIFT (Lowe, 2004) [4] etc. This was later followed by a learning algorithm.

It was extremely tedious and complicated since it had to be revisited every single time the dataset changed considerably. Almost all traditional plant disease detection attempts had this problem and thus relied heavily on very labour-intensive methods. Traditionally, the focus was on a small number of classes, typically within the same crop. E.g. "a feature extraction and classification pipeline using thermal and stereo images in order to classify tomato powdery mildew against healthy tomato leaves [5]; the detection of powdery mildew in uncontrolled environments using RGBD images [6]; comparison of two aerial imaging platforms for identification of Huanglongbing infected citrus trees [7]; the detection of tomato yellow leaf curl virus by using a set of classic feature extraction steps, followed by classification using a support vector machines pipeline etc. Recently, the use of machine learning on plant phenotyping [8] ("for the classification and detection of Phalaenopsis seedling diseases like bacterial soft rot, bacterial brown spot, and Phytophthora black rot") substantially talked about the work in this area of study. Even though neural networks were used in the identification of diseases, it required images to be carefully selected before the classification could begin. Not long ago, a new study by Ilya Sutskever and his colleagues [9] showed for the very first time that it was practically possible to do end to end supervised training using a deep CNN architecture despite the image classification problems using a large number of classes. This left traditional methods that used hand-engineered methods in the dust by a huge margin. As the labor-intensive aspect of feature engineering

was not present, it made this solution very appealing and promising because of its huge scalability.

3. PROPOSED METHODOLOGY

3.1 Dataset Description

For this experiment, we used the Plant Village Dataset, which comprises over 55 thousand photos of different plants and crops. The plants considered for this dataset include oranges, tomatoes, apples, among others.

David P. Hughes and Marcel Salate [10] compiled the original data set under Cornell University in the year 2015, which was later revised in 2016. Their primary motive behind gathering this dataset was to start a crowdsourcing effort to curb the increase in yield losses. On average, 40% of crops are of no use (worldwide, as of 2015) because of being unable to detect and treat the diseases affecting the crops. Hughes and Marcel believe we can solve this problem using computer vision.

This dataset consists not only of plants affected by bacterial and/or viral diseases, but there is a subset of healthy plant leaves for each plant taken into consideration. It includes photos of 17 basic diseases, 4 bacterial diseases, 2 moulds (oomycete) diseases, 2 viral infections, and 1 mite disease. There are additional 12 crop species.

All the photos in the dataset are in the JPEG format in RGB colour mode, with a resolution ranging from 0.05 to 0.1 Megapixels. The total number of class labels assigned to the plant leaves is 38, where each label is a crop-disease pair. We aim to predict this same pair, with only the photos of the plant leaf being given to the model.
We passed the dataset through an augmentation algorithm where the photos were rotated and colour shifted, to train the model better and generate better results. We saved the augmented photos along with the originals. This offline augmentation led to an increase in the dataset's size from 54k to around 87k images. Apart from this, we added a separate directory comprising around 30 images for the prediction that is to be carried out later.

3.2 Data Preprocessing

We split our dataset into an 85/15 ratio of training and validation, for the preprocessing of the dataset.

Besides the offline augmentation carried out on the dataset, we run the dataset through a preprocessing phase using the Keras ImageDataGenerator. Keras allows us to augment our images in real-time before using them for the training and testing of the model. The different augmentations we used include scaling, sheering, shifting the image either along its width or its height. [11]

The data fed into neural networks will be normalised to make the network's processing easier. In our case, we will normalise the pixel values so that they're all in the [0, 1] range, which is where they were previously.

Before performing any tests, the Convolutional Neural Network will give the validation phase, during which it will boost efficiency. The following phase will be to test the model, and then the best model will be deployed.

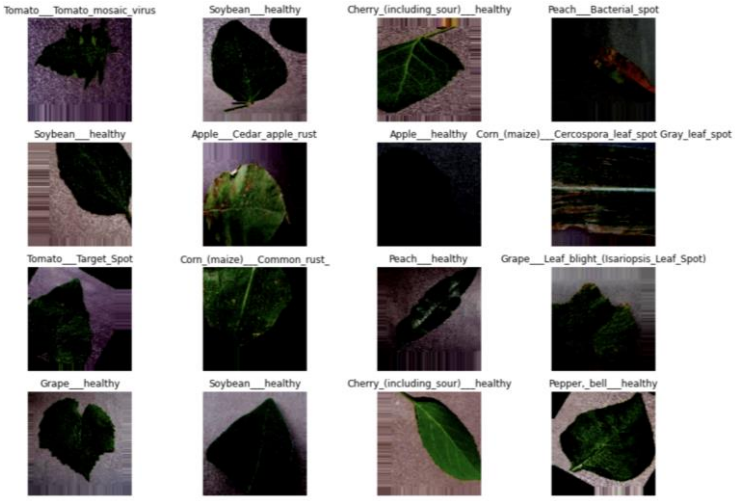

Fig. 1. Preprocessed Images

3.3 Convolutional Neural Network

In our project, we are using the Convolutional Neural Network (CNN) in which a single neuron is present in a layer that will only be connected to a small region of the layer, rather than all the neurons being fully connected. We are using CNN because of its widely demonstrated efficacy in performing image classification, automatic feature learning capability, and ease of training. Generally, CNN is composed of 4 types of layers - convolutional layer, pooling layer, ReLu layer, and fully connected layers. The convolutional layer is the core building block. Its parameters consist of a set of learnable filters that get activated when it detects some specific type of feature at some occupied position in the input. A pooling layer is used to reduce the space occupied by an image in the network. ReLu is an abbreviation for Rectified Linear Units, it enhances the nonlinear properties of the overall network without affecting the receptive fields of the convolutional layer. Finally, after several convolutional layers, the highest level of reasoning is done through the fully connected layers [12].

To train our CNN model, we need an activation function and a pooling layer between each convolution layer. Activation functions are responsible for determining the output of each convolution computation and reducing the complexity of our project. In our case, the activation function is the ReLu function which keeps any negative results at 0 and keeps the positive values for the same and having all these zeroes will make the network more efficient to train in computational time.

For our project, we've chosen DenseNet201 instead of VGG because the deeper the CNN gets, the harder it becomes to train as the gradients start to disappear. DenseNet improves gradient propagation by connecting all layers directly to each other. This vanishing gradient problem occurs in networks with large layers. When we backpropagate the unit, this unit is reduced at every step and eventually becomes zero because we're partially deriving each unit. It has some advantages over VGG because it uses parameter efficiency which means that each layer only adds a limited number of parameters thereby improving the flow of gradients through the network. As compared to VGG, it has reduced the number of parameters by five times with the same number of layers.

3.4 Flow Diagram

The project is divided into three stages:

- TensorFlow and Keras will be used to build and create a machine learning model.
- TFLite will be used to deploy the model to an Android application.
- The development method will be documented and made open source.

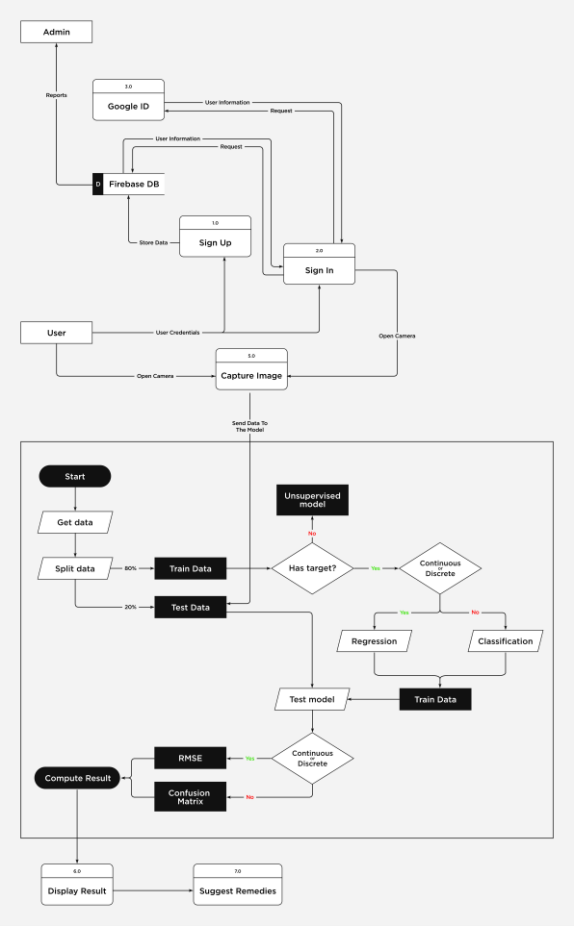

Fig. 2. Flow Diagram

3.5 Densenet Algorithm

In our project, we are focusing on arguably one of the best and most promising CNN architecture - DenseNet (Densely Connected CNN). It has multiple versions depending on the depth, varying from 121 layers with 800,000 parameters to 264 layers with 15,300,000 parameters. In a typical network, n layers will have n connections, whereas here, it'll have n(n+1)/2 connections.

Let's assume we have x number of input layers, and an RGB image will have 3 channels. In the first layer (R), it'll make 4 feature maps (a=4). But as we go deeper into the network to layer 2 (G), it'll not only take 4 layers from the previous layer (denoted by the black arrow) but also layers from the input layer (denoted by the yellow arrow). Further, the 3rd layer (B) takes all the preceding layers as input.

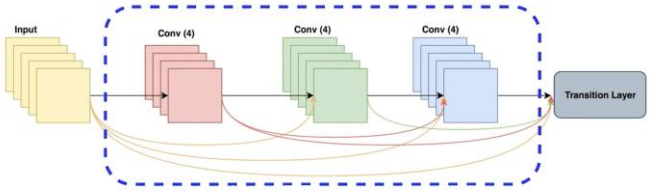

Fig. 3. Diagram showing Image Mapping in a DenseNet for an RGB Image

However, this becomes unsustainable as we go deeper and have more layers. This may lead to a feature map explosion. To overcome this issue, the number of output maps are fixed for each layer and also, a dense block is created which has a predetermined number of layers inside it. The feature maps are shared among those layers. The output from a particular dense block is passed on to a transition layer which uses a 1×1 convolution followed by max-pooling to reduce the size of the feature maps. [13]

Without a dense block, max-pooling wouldn't have been possible because the size of feature maps across max-pooling would be less and would be extremely difficult to concatenate the feature maps. To reduce the model complexity and size, BN-ReLU-1×1 Conv is done before BN-ReLU-3×3 Conv [14].

Here's a diagram to explain how DenseNet will be used to detect a diseased leaf:

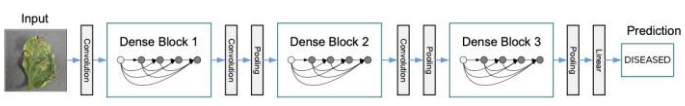

Fig. 4. Detecting a diseased leaf using DenseNet

4. RESULT AND ANALYSIS

4.1 Optimizers, Losses, and Metrics

In our project, we chose 'Adam' as the optimizer because of its computational efficiency. We found it to be the most suitable for our dataset, which is large and contains over 55,000 images. It is a combination of both Adagrad algorithm and RMSProp algorithm and thus, yields the most optimized results. [15]

The loss function we chose to minimize was 'Categorical Cross-Entropy'. It is suitable for multiple classes and our project has 38. As the result can only belong to one category out of many, we needed the model to decide which one and thus, making it the pick for our project.

For the metric, we chose 'Accuracy' as it is a built-in function, commonly used in Keras.

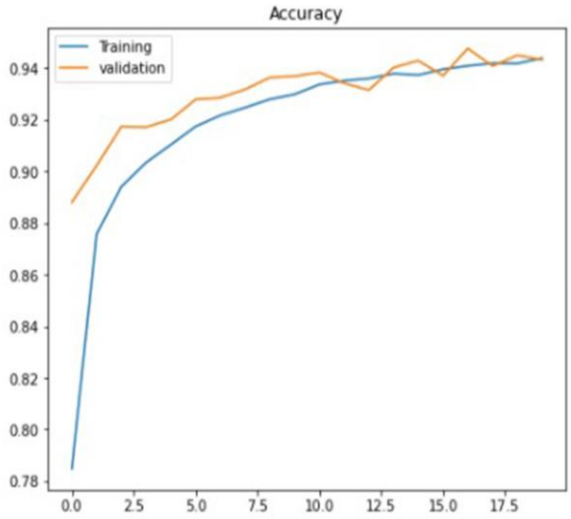

Fig. 5. Plotting Validation Accuracy and Training Accuracy of our model against epochs

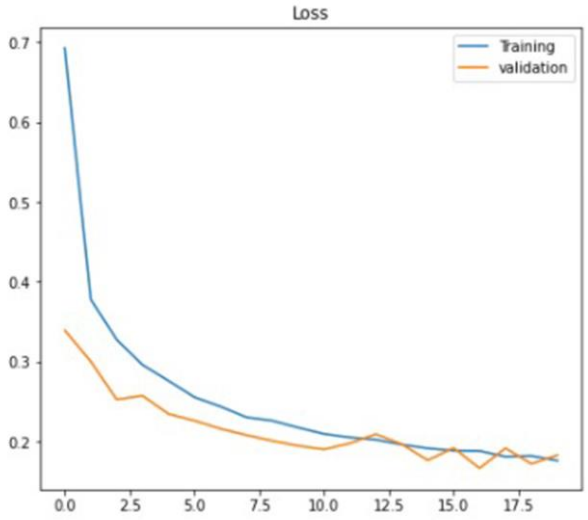

Fig. 6. Plotting Validation Loss and Training Loss of our model against epochs

4.2 Comparison with similar algorithms

Since the release of the initial version of the Plant Village Dataset by Hughes and Salate, a lot of experiments on disease detection using plant leaves have been conducted by various scholars and researchers. Hughes and Salate, in their initial experiments, could achieve an accuracy of 31.8% but with time and developments in CNN, the accuracy has now increased drastically.

	DenseNet201	**VGG16**
Size (MB)	80	528
Layers	201	16
Top-1 Accuracy	0.773	0.713
Top-5 Accuracy	0.936	0.901
Time (ms) per inference step (CPU)	127.24	69.5

Table -1: Keras Models DenseNet201 vs VGG16

One such project is that of Dhiman Thakuria of Kaziranga University, titled Leaf Disease, which uses the Plant Village Dataset to detect and predict diseases. We have used the same dataset but to increase the accuracy, added more images by augmenting and saving a portion of the images again in the dataset. [16]

Moreover, we have used the DenseNet201 of the Keras Application compared to VGG16 that has been used in their project. The advantages of using DenseNet201 over VGG16 are as follows:

- DenseNet201 has a depth of 201 layers which is 185 layers more compared to the number of layers in VGG16 (16 layers)
- VGG16, which was released in the year 2014 is also very heavy in terms of boot disk required, 530MB in total. DenseNet201 on the other hand is only 80MB in size.
- DenseNet201 yields better results in both Top-1 Accuracy and Top-5 Accuracy compared to VGG16.
- When run on the CPU, VGG16 takes less time per each inference step.

Let's compare the accuracy of the two projects now. As it can be seen from Fig. 7 and 8, our model has a better validation accuracy and loss than that of Dhiman's.

3. CONCLUSIONS

The plant leaf disease recognition model that we have proposed in this paper, is supervised, has high accuracy, and has high training efficiency. The model, primarily based on deep learning still has some problems. To solve the

remaining problems in accuracy and practicability of plant disease detection and enhance the detection process, this paper proposes a CNN model which could efficiently save us the trouble of plant disease recognition in a tedious way. The proposed model increases the accuracy of disease detection and adapts to complex environments. In contrast to the conventional approach, the one proposed in this ensures the robustness of the CNN and additionally reduces manual effort drastically. Therefore, the version may want to assist agricultural manufacturing employees to save you and therapy the plant ailment quickly.

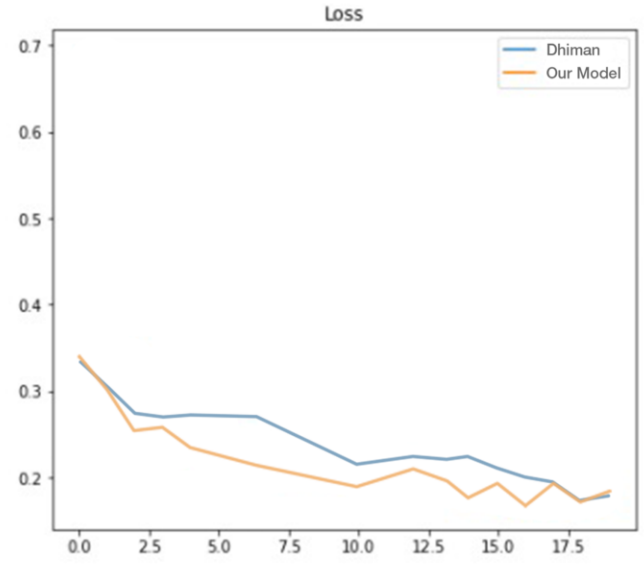

Fig. 7. Validation Loss Comparison

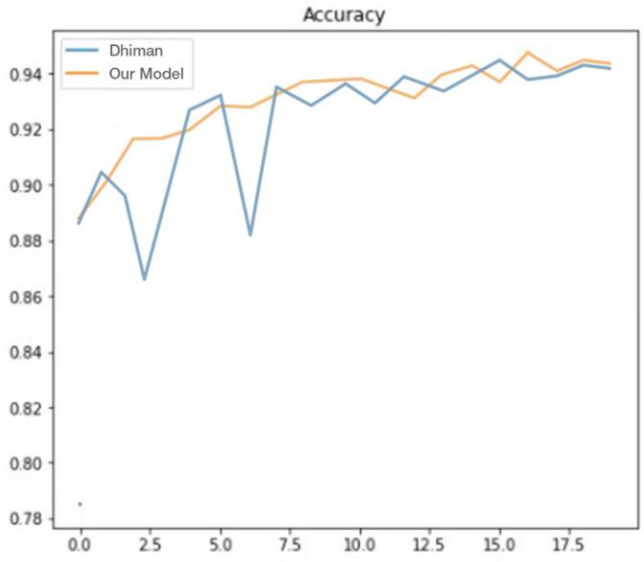

Fig. 8. Accuracy Loss Comparison

The proposed algorithm enhances the conventional approach and helps to increase the accuracy. In the field of smart agriculture this approach will be of substantial importance and facilitates scientists and researchers alike, to take note of the critical position of problems faced in plant disease recognition.

Hence, the model uses convolutional networks and statistics to solve a predominant problem in agricultural manufacturing and is complementary to the sustainable advancement of smart agriculture.

REFERENCES

[1] Manglesh R. Yadav and Shashank Gore. Strengthening the Indian Agriculture Ecosystem.
[2] Karen Simonyan and Andrew Zisserman. Global spread of plant pests and diseases.
[3] The Food and Agriculture Organization of the United Nations. New standards to curb the global spread of plant pests and diseases.
[4] Navneet Dalal and Bill Triggs. Histograms of Oriented Gradients for Human Detection.
[5] John Clarkson Shan e Ahmed Raza, Gillian Prince and Nasir Rajpoot. Automatic Detection of Diseased Tomato Plants Using Thermal and Stereo Visible Light Images.
[6] David P. Hughes Sharada P. Mohanty and Marcel Salathe. Using Deep ´ Learning for Image-Based Plant Disease Detection.
[7] Joe Mari Maja W. S. Lee Francisco Garcia-Ruiz, Sindhuja Sankaran. Comparison of two aerial imaging platforms for identification of Huanglongbing-infected citrus trees.
[8] D Ng JJ Johnston, KL Lewis and LN Singh. Individualized iterative phenotyping for genome-wide analysis of loss-of-function mutations.
[9] Ilya Sutskever Alex Krizhevsky and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks.
[10] David Hughes, Marcel Salathe, et al. An open access repository of ´ images on plant health to enable the development of mobile disease diagnostics. arXiv preprint arXiv:1511.08060, 2015.
[11] Manpreet Singh Minhas. Image Data Generators in Keras.
[12] Saad Al-Zawi Saad Albawi, Tareq Abed Mohammed. Understanding of a convolutional neural network.
[13] Jason Brownlee. Model complexity. arXiv preprint arXiv:1312.4400, 2017.
[14] Sik-Ho Tsang. Review: DenseNet — Dense Convolutional Network (Image Classification).
[15] Jimmy Ba Diederik P. Kingma. Adam: A Method for Stochastic Optimization.
[16] Dhiman Thakuria. Leaf Disease Detection