Deep Learning Models for Classification of Cotton Crop Disease Detection

M. N. Tibdewal[1], Yash M. Kulthe[2], Ankush Bharambe[3], Atharva Farkade[4], Anup Dongre[5]
[1, 2, 3, 4, 5]*Dept. of Electronics and Telecom. Engineering, Shri Sant Gajanan Maharaj College of Engineering Shegaon, Maharashtra, India, 444203*

Abstract: *In India, plants play a vital role in the survival of all organisms. While India solely depends on its agriculture. The crop yield should be increased. Cotton being the most important cash crop for India's industry. Whereas it provides cotton textile industry with the raw material. As a result, a significant effort should be made to combat plant diseases, which could result in a significant loss of yield. The paper discusses the deep learning models that can be used for the cotton plant disease detection majorly Convolution Neural Networks, XceptionNet, LeNet, DenseNet and MobileNet using the images of the diseased and healthy cotton leaves. The training on the model is done using the dataset consisting of total 1711 images classified into three diseased and one healthy leaves class. The training was done keeping the input and output size as constant for all the models. The DenseNet has achieved the best training and validation losses, optimum file size and accuracy of 98.25%. And it required a smaller number of parameters compared to the other models. The results of the experiments are shown to assess the performance of all the models.*

Keywords: **Plant Disease- Bacterial Blight, Curl Virus, Fusarium Wilt, Deep Learning (DL), Convolution Neural Network (CNN), XceptionNet, LeNet, DenseNet, MobileNet.**

1. INTRODUCTION

Agriculture has become a key source of economic development in India, accounting for 18% of its GDP, making India dependent on crop productivity. The appropriate crop is chosen by the farmer depending on the kind of soil, local climatic conditions, and economic worth. Because of rising population, changing weather, and political instability, agriculture companies began looking for innovative ways to enhance crop yield. The new advancement in genetic technology and fertilizer industry, the crop yields Over the decades, there has been seen a remarkable increase.

However, still in some underdeveloped regions due to less knowledge of the farmers, crop diseases has been the most troublesome to the agricultural output and the food security is at risk. These leaf diseases can appear extremely similar in their early stages, making them not easy to classify with naked eye. Visual inspection of leaf diseases calls for a expert in the field and monitoring the crop continuously. As a result, it is exceedingly expensive, time-consuming, and unreliable. Deep learning algorithms can be used to detect and classify leaf diseases automatically, quickly, and precisely.

This task requires a huge number of sample photos of diseased corps to construct a reliable plant disease detection system. In the past, collecting such a vast number of samples was difficult. But nowadays, practically everyone now owns a mobile phone with a good camera, which may be used to gather the images locally by the farmers himself.

Cotton is the most significant fiber crop in the world, not only in India providing the cotton textile industry with the raw material. So, the data used, consists of 1711 images of cotton leaf classified into 3 diseased and 1 non-disease class: Bacterial Blight, Curl Virus, Fusarium Wilt and Healthy Leaf. Also, as the farmers nowadays can afford a smartphone, but due to internet connectivity issues in rural areas, a lightweight detection model should be implemented on the smartphones or Internet of Things (IoT) devices which have limited computing capabilities. The main objective of this work is:

• To know various deep learning models.
• Compare the different models based on different parameters.
• Propose the best deep learning model and know it's architecture.
• To project the future scope of our research in agriculture.

The paper is organized as, Section 2 narrates the performance of existing deep learning models in agricultural applications. The Section 3 describes the proposed methodology.

Section 4 presents the experiment settings and result. The end part of paper is Section 5 discussed about conclusion.

2. EXISTING DEEP LEARNING MODELS

Ferentinos and Konstantinos P. [7] trained many model architectures on an open-source database including 87,848 photos, with 25 different plants in a set of 58 distinct classes of plant disease combinations, including healthy plants, with the best performance reaching a 99.53 percent success rate. The models were like AlexNet, GoogLeNet, VGG.

G. Sambasivam et al. [9] worked on the Cassava disease detection with 10000 identified images in a limited image dataset of 5 fine-grained cassava leaf disease groups, where in they used a Convolution Neural Network to solve the problem from scratch using an imbalanced dataset achieving an accuracy score over 93%.

Saleem and Muhammad Hammad [10], they proposed the comparison of state-of-the-art deep learning models with the evolution of various DL models from 2012 till date. Abdul et al. [8] proposed an optimized dense CNN architecture for corn leaf disease with the accuracy of 98.06% with even less parameters compared to the models like EfficientNet, VGGNet, XceptionNet. Bhatt, Prakruti [17], they distinguished between diseases that looked to be identical, researchers used many CNN architectures added with a combination of adaptive boosting and a decision tree-based classifier to construct a system for identifying maize leaf diseases. Healthy leaves, common rust, leaf blight, and leaf spot were among the image data categories. The Plant Village dataset provided images for each class. The model provided with an accuracy of 98%.

Shruthi U [6], studied five types of machine learning classifications for plant disease detection. Where it was found that CNN classifier detects more efficiently compared to SVM classifier. Mercelin Francis and C. Deisy [5] created a CNN model for apple and tomato leaf disease detection. Where in ach convolutional layer is followed by a pooling layer in the model. To determine whether or not disease is present, two fully connected dense layers and a sigmoid function are used. The model was trained using a 3663-picture dataset of apple and tomato leaves, resulting in an accuracy of 87%.

The basics of deep learning come from the Convolution Neural Networks (CNNs) which work on the principle of a kernel and the input image convolution. CNNs are the most important part of deep learning or any image classification algorithm. Where in normal Machine Learning, it is important to visualize the pattern, extract the needed features, do perfect pre-processing, and much more. But, for the CNNs, it directly recognizes the pattern from the image without much of the tedious tasks to be done. There are many models proposed by the researchers in the field of deep learning. Some of the popular models studied are discussed below:

A. Convolution Neural Network (CNN)

The Convolutional Neural Network (CNN) is a deep learning technology that has been a revolution in the field of image processing. Face recognition, object identification, image categorization, and other applications of CNN are common. Convolutional layers, pooling layers, fully linked layers, activation functions, and other components of a CNN model are some components that are used in different ways.
We have proposed a model architecture which can work well with a smaller number of layer and neurons in each layer. Then the major challenge was to choose the input which should not be too big which makes the model to have huge number of parameters, while not too less which will affect the performance. We decided to take an input image of size (150, 150, 3). Suppose the input has size W, padding P, filter size F and strides S, then using the convolution on filter on the input, the next layer will be of size given in (1).

$$size = (W - F + 2P)/S \tag{1}$$

We propose a 15-layer model having an input layer for the image data, four Conv2D layers for extracting the features, two Dense layers using ReLU activation for adding up the training parameters so as to extract more features, and a last dense layer using Soft max activation for final output with total parameters of 2,028,228 as seen in Table 1.

Table 1. Results Of Training Various Deep Learning

Layer	*Output Shape*	*Total Parameters*
Input	150^2 x 3	0
Conv2D	148^2 x 32	896
MaxPooling2D	74^2 x 32	0
Conv2D	72^2 x 64	18496
MaxPooling2D	36^2 x 64	0
Conv2D	34^2 x 128	73856
MaxPooling2D	17^2 x 128	0
Conv2D	15^2 x 256	295168
MaxPooling2D	7^2 x 256	0
Dropout	7^2 x 256	0
Flatten	12544	0
Dense	128	1605760
Dropout	128	0
Dense	256	33024
Dropout	256	0
Dense	4	1028

A. Convolution Neural Network (CNN)

The Convolutional Neural Network (CNN) is a deep learning technology that has been a revolution in the field of image processing. Face recognition, object identification, image categorization, and other applications of CNN are common. Convolutional layers, pooling layers, fully linked layers, activation functions, and other components of a CNN model are some components that are used in different ways.

B. XceptionNet

With the introduction of a model named as inception network by Szegedy [18]. The inception network's key feature was the use of computational resources within the network. It does away with all fully connected layers in favor of average pooling, resulting in a system with only 5 million parameters. With the big boom of the inception model, which in the stack of feature maps captures parallel pathways with different receptive field size and operations, there were many advanced versions made of it and a network was then proposed called as XceptionNet [16], it outperformed many algorithms on a large dataset.

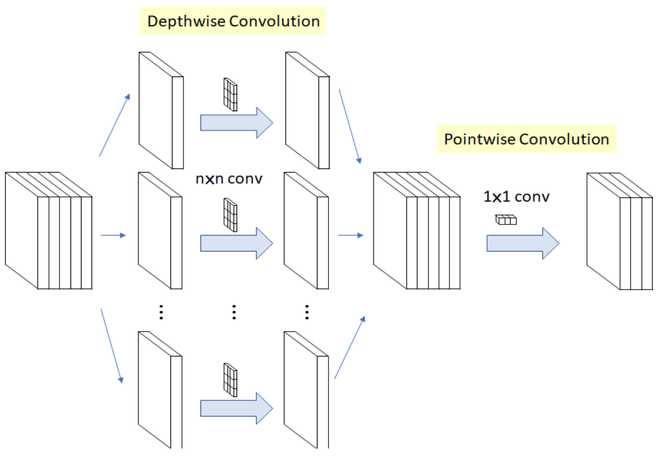


Figure 1. Depth wise Separable Convolution [22]

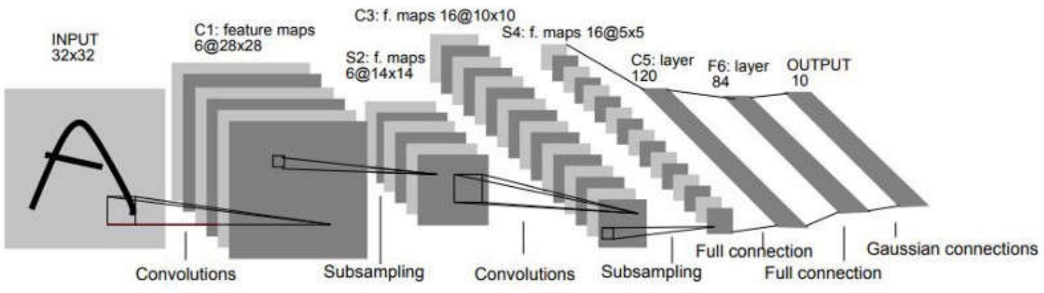


Figure 2. Architecture of LeNet-5 [19]

C. LeNet

The LeNet-5 was introduced by Yann LeCun and Yoshua Bengio [19]. It was one of the first CNN architectures. LeNet is considered to be the father of all the CNNs as it was introduced at a early time when there was not a lot of research in the field of Neural Networks. LeNet-5 has a straightforward architecture. It has three convolution layers, two pooling layers, and three dense layers. It takes a 32 x 32 image as input and outputs a 10-item list that contains the probability for each class. The architecture of the LeNet-5 is shown in Figure. 2.

D. DenseNet

In a feed-forward fashion, DenseNet [14] connects each layer to the other layers. The feature maps of the first layer are utilized as inputs in each of the following layers, and their feature maps are used as inputs in all future layers. DenseNet solves the vanishing gradient problem [20], improves feature propagation, allows for feature reuse, and greatly reduces the number of parameters. Each nth layer contains n inputs, which are made up of the feature-maps from the convolutional blocks before it. Its feature-maps are passed on to all N - n following layers. In an N-layer network, this introduced N (N + 1)/2 connections. This architecture is known as a Dense Convolutional Neural Network because it features a dense connectivity pattern.

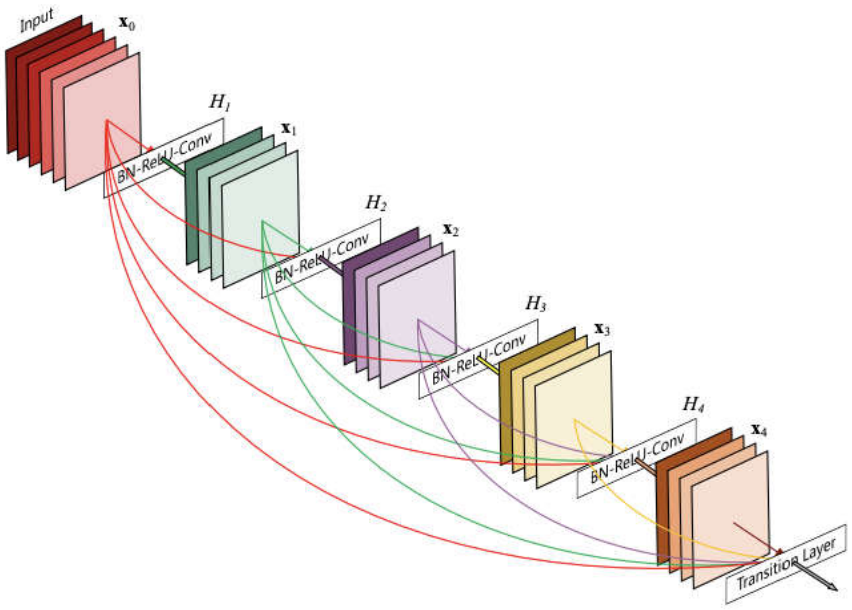


Figure 3. DenseNet 5-layered Block [14]

E. MobileNet

Mobile Net [21] is a convolutional neural network for mobile vision applications that is efficient, simple and uses less computational power. Object detection, fine-grained classifications, facial traits, and localization are just a few of the real-world applications that use MobileNet. MobileNets are made up of depth wise separable convolutions, which were first employed in Inception models to reduce processing in the first few layers. Flattened networks produced a network that was entirely made up of fully factorized convolutions, showcasing the capability of highly factorized networks. The Xception network revealed how to outperform Inception V3 networks by scaling up depth wise separable filters.

3. PROPOSED METHODOLOGY FOR DEEP LEARNING

A model in a conventional Convolutional Neural Network contains an input image, which is then processed through the hidden layers then to the output layer to generate an output of predicted label in a relatively simple forward pass. Except for the first convolutional layer, each convolutional layer uses the preceding convolutional layer's output to create an feature map, which is then given as input to the next layer. There are N direct connections between each layer and its subsequent layer for N layers.

In DenseNet each layer in the design is connected to every other layer. N(N+1)/2 direct connections exist for N layers. The information from the previous layers are used as input for each layer, and the output of the current layer will be used as input for the upcoming layers. The skip connections solve the problem of vanishing gradient, hence which allows us to train the network deeper. As a result, they can be trained deeper than traditional networks and easily optimized with this new residual usage. This innovative CNN architecture has delivered state-of-the-art results on many standard datasets with extremely few parameters. The huge advantage of DenseNet is the usage of bottleneck layer. To limit the amount of input feature-maps and hence enhance computational efficiency, a bottleneck layer of 1 x 1 convolution is added to reduce the number of feature-maps.

The first layers of the DenseNet model contains a 7x7 convolution Layer with stride-2 followed by a 3 x 3 MaxPool layer stride-2. Which is followed by a set of Dense Blocks and Transition Layers, and the final block is the Classification Layer that takes the input from these previous blocks and performs the classification.

A. Dense Connectivity

Consider a network of L layers, each performing a J_l non-linear transformation. The output of the network's Lth layer is indicated as xl, whereas the input image is designated as x0. In the forward propagation, the output of the Lth layer is given as an input to the $(L+1)^{th}$ layer, as we all know. The skip connection, on the other hand, can be expressed in (2).

$$x_1 = J_l(x_{l-1}) + x_{l-1} \tag{2}$$

The dense connections in the DenseNet model are given as in (3).

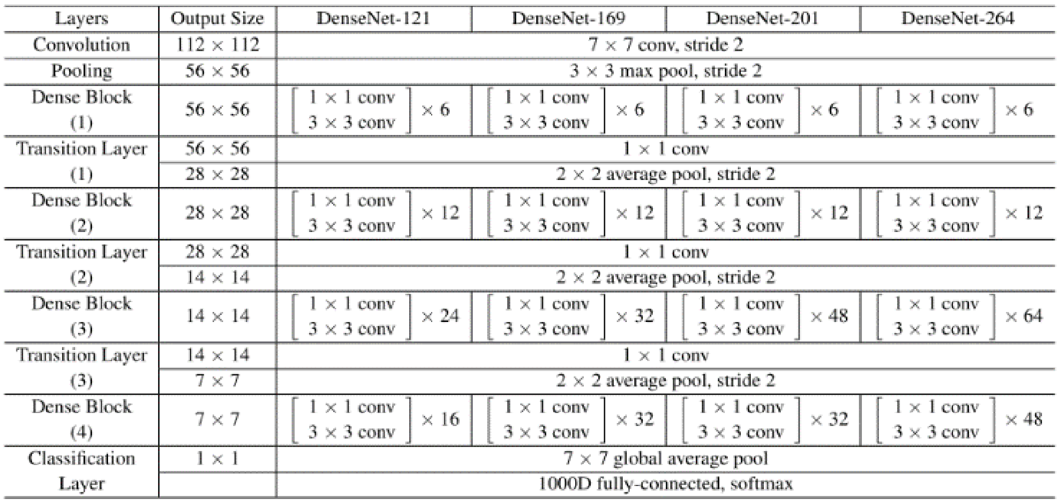
$$x_1 = J_l([x_0, x_1, \ldots, x_{l-1}]) \tag{3}$$

B. Transition Layers

Transition layers are the layers that perform convolution and pooling between dense blocks. A batch-norm layer, 1x1 convolution, and a 2x2 average pooling layer make up the DenseNet architecture's transition layers. The 1x1 convolution performs the down sampling. There are many different architectures of DenseNet being, the DenseNet-121, 169, 201 and 264. The model having the least number of parameters is the DenseNet-121 and also, it is good for low to moderate number of training data hence, we are going to work on this algorithm a bit and modify it for our images. The conventional DenseNet architectures are shown in Figure. 4.

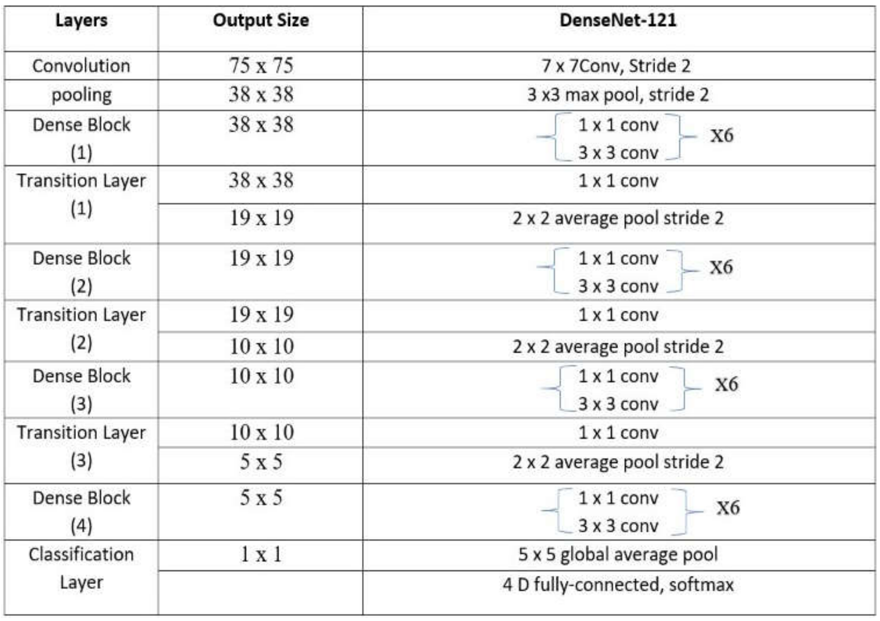
As it is training all the models on an image input of (150, 150, 3) with a batch size of 32, it was mandatory to keep the input and output same for all the models, to keep the parameters less and also to compare the models by the hidden layers.

In the original DenseNet-121, the input parameters are of shape 224 x 224 as shown in Fig. 4. The proposed modified DenseNet contains the same hidden network but with different input and output tensors. The first convolution 7 x 7 when given an input of 150 x 150, stride-2, gives the output of 75 x 75 then given a pooling layer to reduce the parameters by reducing the feature map whose output is 38 x 38. Which is then followed by a Dense Block containing 3 x 3 and 1 x 1 convolutions followed by a Transition layer having a bottleneck layer and an avg. pooling with stride-2 giving an output of 19 x 19, then a series of 2 more pair of these Dense Block and Transition Layers are used, which is at last followed by one more Dense Block having output of 5 x 5. Lastly, a Classification Layer is used having global avg pool layer and a softmax fully-connected layer having 4 outputs for our cotton image classification. The proposed model can be seen in Figure. 5.



Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Figure 4. DenseNet Structure [14]



Layers	Output Size	DenseNet-121
Convolution	75 x 75	7 x 7Conv, Stride 2
pooling	38 x 38	3 x3 max pool, stride 2
Dense Block (1)	38 x 38	{1 x 1 conv, 3 x 3 conv} X6
Transition Layer (1)	38 x 38	1 x 1 conv
	19 x 19	2 x 2 average pool stride 2
Dense Block (2)	19 x 19	{1 x 1 conv, 3 x 3 conv} X6
Transition Layer (2)	19 x 19	1 x 1 conv
	10 x 10	2 x 2 average pool stride 2
Dense Block (3)	10 x 10	{1 x 1 conv, 3 x 3 conv} X6
Transition Layer (3)	10 x 10	1 x 1 conv
	5 x 5	2 x 2 average pool stride 2
Dense Block (4)	5 x 5	{1 x 1 conv, 3 x 3 conv} X6
Classification Layer	1 x 1	5 x 5 global average pool
		4 D fully-connected, softmax

Figure 5. Proposed DenseNet-121 Structure

4. EXPERIMENTAL RESULTS

The experiment setting and results for the models addressed in Section II with input size 150x150x3 and Soft-Max output of 1x4 are presented in this section. The hyperparameters and the settings are as follows:

- The most popular optimization function, Adam optimizer, with a learning rate of 0.0001 and a sparse categorical cross-entropy loss function because our model is working on multiple class classification.
- The epochs to 200 and set a callback list to save the best model.
- The data was shuffled and manually divided it into two sets: 20 percent for validation and 80 percent for training from 1711 images
- Data Augmentation is done before feeding the data to our model.

4.1. PC Hardware Setting

Nowadays, the deep learning models have become more intricate with the architecture, which need GPUs or TPUs to train the models but, our model is far smaller than the orthodox architectures for deep learning and have capabilities to run and train on systems with low processing power. However, we decided to go with the free version of google colab which provides us a GPU unit computer for a short period of time. Colab is a free to use online application like Jupyter notebook. Most importantly, it doesn't require any setup, and you can share the file with your team wherein they can also modify the same file on cloud, much like Google Docs projects. The plus point of working on Google colab is that all the libraries be it general python or advanced machine learning libraries are pre-installed in it, all we need to do is import them.

4.2. Data Preprocessing

Pre-processing data is an important part of any machine learning methodology. The models used, have large number of parameters, implying that the model will require a huge number of training data. Furthermore, gathering and producing well-labeled data takes a lot of time and work. This problem is solved mainly by the augmentation of the training data. In general, augmentation of data is nothing but to generate some duplicates of an image with the addition of some parameters such as cropping, shearing, noise and transform the dataset. This can be used on practical applications where there is a shortage of data and can generate huge dataset by the processing of the data by these methods.

There are huge advantages of data augmentation, like the addition of noise previously discussed can be a saving element for the model, as a model trained on dataset having noise, it will know how to deal with the noises and essential elements of the data, making the data more robust in dealing with unknown data. It also aids in overcoming the issue of class imbalance in categorization issues. The main plus point being that it reduces the cost of collection of extra data and labelling it. The main advantage being that the models learn limited data over time, it can prevent them from over-fitting and by learning the data from various angles.

The process of data augmentation has been extremely efficient and easy-to-use by the introduction of Image Data Generator by Keras. It lets us to augment the images in real-time while the model is still in training phase. Which saves up a lot of memory and make the training process robust. We have used the Image Data Generator with the following settings:

- Normalize the data by dividing it by 255
- Rotation Range – 40^o
- Width Shift Range
- Height Shift Range - 0.2
- Shear Range - 0.2
- Zoom Range - 0.2
- Horizontal Flip – TRUE
- Resized the images to 150 x 150 with 32 batch size.

4.3. Results

The trained 6 models being our own designed are, Convolution Neural Network, XceptionNet model, LeNet, DenseNet-121 and MobileNet. All of them were designed such that the input will be of 150x150x3 which outputs a data of 1x4. The other settings were discussed before in part A of this Section.

Table 2 shows the results of our training over some of the models with the same dataset and other settings. The CNN model has shown a really good potential being its training and validation accuracy is 96.19% and 95.63% respectively and also it has a smaller number of parameters, and the model size was also low i.e., 23.3 MB. XceptionNet was the most brilliant and a powerful architecture which comes with the major disadvantage of the file size being over 250 MB which is not feasible for a mobile application or IoT systems, otherwise all of its aspects were brilliant just the number of parameters over 22.9 million which requires high end computing device.

The LeNet model, one of the oldest models out of these has also performed really well in the classification algorithm. But in Table 2 as we can see the major disadvantage here was the underfitting of data which can be seen as the validation accuracy is greater than the training. For this reason, we do not see it fit for our classification. The MobileNet which was developed for the sole application to work on low computing power systems like embedded or IoT system. It has performed exceptionally compared to the others previously discussed as it had approx. 3.2 million parameters and with that, it achieved an accuracy of 96.21% on the validation. Although, it is the same case here, the training accuracy is less than the validation which can be resolved by feeding more data or using more intricate network.

The one model who has the standpoint here is the proposed DenseNet-121 model which was seen the handiest in all the parameters achieving a training accuracy of 98.34% and validation accuracy of 98.25%. With optimum file size and parameters of 46.2 MB and approx. 3.9 million respectively. The major difference seen in all the architectures is in the loss function and the epoch at which maximum validation accuracy was obtained. The training and validation losses must be theoretically same or the difference between them must be minimum to minimum. When comparing all the architectures, it is found out that the least loss is in XceptionNet and DenseNet, but the minimum difference is found out in the proposed DenseNet model, making it the superior one. Also, as we can see that the DenseNet achieved the optimum results at the 78th epoch making it the model which can trained fastest. While in the others, the optimum results were found when they crossed the 150-epoch mark. Although, the performance of the DenseNet model can be increased with more data added to training or with the use of transfer learning.

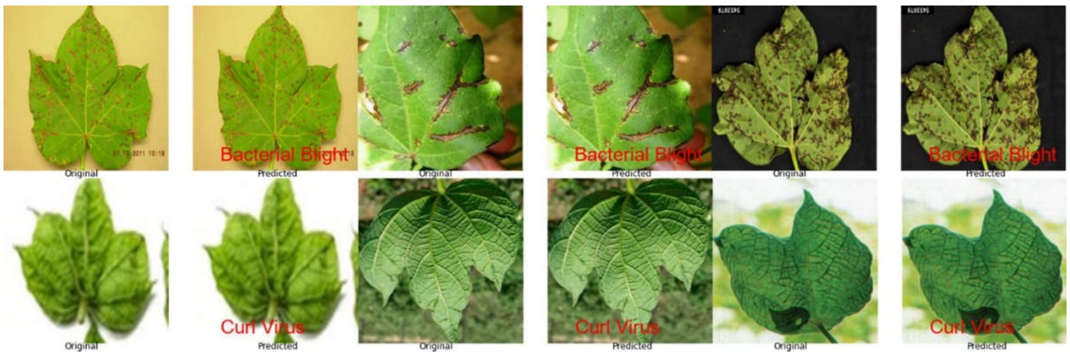


Figure 6. Results of testing the model on unseen data - Bacterial Blight and Curl Virus

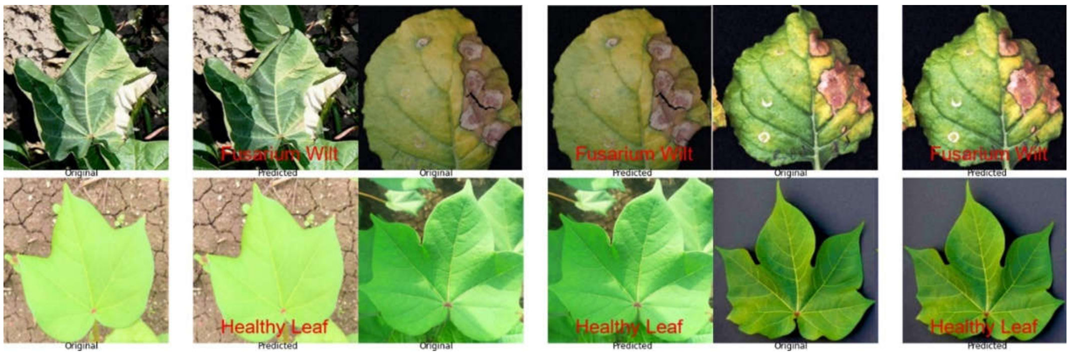


Figure 7. Results of testing the model on unseen data - Fusarium Wilt and Healthy Leaf

These all the parameters make the proposed DenseNet-121 model the best pick out of these 5 architectures and is quite adaptable and feasible for both high and low-end computing systems, as it achieves really good results in low parameters and the final model is formed of 46.3 MB which is normal for today's era.

For testing purposes, 12 unseen images of diseased and healthy cotton leaves are tested on the DenseNet model and the results can be seen in Figure. 6. for Bacterial Blight and Curl Virus, whereas in Figure. 7. for Fusarium Wilt and Healthy Leaf. As discussed previously, we have seen that data augmentation helps a lot in images that might be distorted or noisy. In Figure. 6., the first prediction for curl virus is done on an image with low resolution, making its noise high. A correct prediction is done on it. Showing that data augmentation is a key part.

Table 2. Results Of Training Various Deep Learning Models

Architecture	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss	File Size	No. of Parameters	Max. Validation Acc. at epoch no.
CNN	96.19%	95.63%	0.1066	0.1779	23.3 MB	2,028,228	194
XceptionNet	99.03%	99.12%	0.0328	0.0732	262.6 MB	22,914,484	169
LeNet	97.71%	96.21%	0.0721	0.1319	173 MB	15,115,864	147
DenseNet-121	98.34%	98.25%	0.0425	0.0635	46.2 MB	3,921,476	78
MobileNet	95.91%	96.21%	0.1257	0.1704	37.5 MB	3,243,908	180

5. CONCLUSION AND DISCUSSION

Convolution Neural Networks are very perspective research field. To take an image, detect, and classify the leaf if it is diseased or healthy, a modified DenseNet-121 was used. When compared to other current models, it achieved a training accuracy of 98.34% and validation accuracy of 98.25% with 3,921,476 number of parameters. As the model was trained perfectly on 78th epoch and started to overfit the data on the training data, suggesting us to use the early stopping method to pick the model which is perfect in variance and bias. Using the orthodox deep leaning architectures to train from scratch, to get optimum accuracy is a time-consuming operation. As a result, different models can be employed or retrained depending on the previously trained models referred to as transfer learning. As a result, in future work, it is proposed to employ a transfer learning model which being more efficient, less in size and easy to apply than traditional training, which might be one of the best solutions to be utilized in smartphones or other embedded devices. We will also work on plant disease detection android application to help with deep learning-assisted smart agriculture for common man.

Acknowledgments

We would like to thank Serosh Karim Noon for providing us the dataset on Kaggle and making it available for us.

REFERENCES

[1] Yang, Xin, and Tingwei Guo. "Machine learning in plant disease research." March 31 (2017): 1.

[2] Arsenovic, Marko, et al. "Solving current limitations of deep learning-based approaches for plant disease detection." Symmetry 11.7 (2019): 939.

[3] Sujatha, R., et al. "Performance of deep learning vs machine learning in plant leaf disease detection." Microprocessors and Microsystems 80 (2021): 103615.

[4] Goncharov, P., et al. "Disease detection on the plant leaves by deep learning." International Conference on Neuroinformatics. Springer, Cham, 2018.

[5] Francis, Mercelin, and C. Deisy. "Disease detection and classification in agricultural plants using convolutional neural networks—a visual understanding." 2019 6th International Conference on Signal Processing and Integrated Networks (SPIN). IEEE, 2019.

[6] Shruthi, U., V. Nagaveni, and B. K. Raghavendra. "A review on machine learning classification techniques for plant disease detection." 2019 5th International conference on advanced computing & communication systems (ICACCS). IEEE, 2019.

[7] Ferentinos, Konstantinos P. "Deep learning models for plant disease detection and diagnosis." Computers and electronics in agriculture 145 (2018): 311-318.

[8] Waheed, Abdul, et al. "An optimized dense convolutional neural network model for disease recognition and classification in corn leaf." Computers and Electronics in Agriculture 175 (2020): 105456.

[9] Sambasivam, G., and Geoffrey Duncan Opiyo. "A predictive machine learning application in agriculture: Cassava disease detection and classification with imbalanced dataset using convolutional neural networks." Egyptian Informatics Journal 22.1 (2021): 27-34.

[10] Saleem, Muhammad Hammad, Johan Potgieter, and Khalid Mahmood Arif. "Plant disease detection and classification by deep learning." Plants 8.11 (2019): 468.

[11] Venkataramanan, Aravindhan, Deepak Kumar P. Honakeri, and Pooja Agarwal. "Plant disease detection and classification using deep neural networks." Int. J. Comput. Sci. Eng 11.9 (2019): 40-46.

[12] Novaković, Jasmina Dj, et al. "Evaluation of classification models in machine learning." Theory and Applications of Mathematics & Computer Science 7.1 (2017): 39-46.

[13] Liu, Jun, and Xuewei Wang. "Plant diseases and pests detection based on deep learning: a review." Plant Methods 17.1 (2021): 1-18.

[14] Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[15] Noon, Serosh Karim et al. 'Computationally Light Deep Learning Framework to Recognize Cotton Leaf Diseases'. 1 Jan. 2021 : 1 – 16.

[16] Chollet, François. "Xception: Deep learning with depthwise separable convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[17] Bhatt, Prakruti, et al. "Identification of Diseases in Corn Leaves using Convolutional Neural Networks and Boosting." ICPRAM. 2019.

[18] Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[19] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.

[20] Bengio, Yoshua, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult." IEEE transactions on neural networks 5.2 (1994): 157-166.

[21] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).

[22] Tsang, Sik-Ho. "Review: Xception - with Depthwise Separable Convolution, Better than Inception-V3 (Image..." Medium, Towards Data Science, 20 Mar. 2019, https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution-better-than-inception-v3-image-dc967dd42568.