On Using Transfer Learning For Plant Disease Detection

Abhinav Sagar, Dheeba J

Vellore Institute of Technology, Vellore, Tamil Nadu, India

ABSTRACT

Deep neural networks have been highly successful in image classification problems. In this paper, we show how deep neural networks can be used for plant disease recognition in the context of image classification. We have used a publicly available Plant Village dataset which has 38 classes of diseases. Hence the problem that we have addressed is a multi class classification problem. We have compared five different architectures including VGG16, ResNet50, InceptionV3, InceptionResNet and DenseNet169 as the backbones for our work. We found that ResNet50 which uses skip connections using a residual layer archives the best result on the test set. For evaluating the results, we have used metrics like accuracy, precision, recall, F1 score and class wise confusion metric. Our model achieves the best of results using ResNet50 with accuracy of 0.982, precision of 0.94, recall of 0.94 and F1 score of 0.94.

Index Terms—Convolutional Neural Network (CNN), transfer learning, image classification, Residual neural network (RELU), Adam Optimizer, data augmentation

1. INTRODUCTION

It is very important to get an accurate diagnosis of plant diseases for global health and well being. In this ever changing environment, identifying the disease including early prevention is important to avoid problems that we might face otherwise. Some of these problems could have devastating impacts on humanity including global shortage of food. It is crucial to prevent unnecessary waste of financial resources achieving a healthier lifestyle, by addressing climate change from an ecological perspective. It is difficult for the naked eye of a human being to catch all sorts of problems with plant diseases. Also doing this time and time again is also laborious and unproductive work. In order to achieve accurate plant disease detection, a plant pathologist should possess good observation skills so that one can identify characteristic symptoms. An automated system designed to help identify plant diseases by the plant's appearance and visual symptoms could be of great help. This can be deployed in agricultural fields so that the whole pipeline can be automated. This would not only lead to better efficiency as machines could perform better than humans in these redundant tasks but also improve the productivity of the farm. Our work builds on the above mentioned problem of automating plant disease classification using deep learning and computer vision techniques.

2. EXISTING WORK

A lot of research has been done in the last decade on plant disease detection using deep learning and computer vision. Machine Learning approaches include traditional computer vision algorithms like haar, hog sift, surf, image segmentation, Support Vector Machines (SVM), using K-Nearest Neighbours (KNN), K-means and Artificial Neural Networks (ANN) have been successfully applied to a lot of different datasets. Deep Learning based plant disease classification models includes the use of a variety of CNN models such as AlexNet, GoogleNet, VGGNet etc. It is seen oftentimes as the dataset size is not enough, multi class classification with a lot of classes requires careful hyperparameter tuning to avoid overfitting as the model could easily get stuck in a local minimum. [18] This method uses RGB feature extraction techniques to identify the diseases in which the captured images are processed for enhancement first. After this color image segmentation is carried out to get target regions around the original image. Next homogenization techniques including Sobel and Canny filters are applied to identify the edges. These extracted edge features are used in classification to identify the disease spots present in the image. [19] uses images of the infected rice plants by digital camera. Further it also uses the concept of image growing. Image segmentation techniques including K means algorithms to detect infected parts of the plants are used. Finally the infected part of the leaf has been used for the classification purpose using a deep neural network using a softmax layer. [21] starts by identifying the mostly green colored pixels. Next, these pixels are masked based on specific threshold values that are computed using Otsu's method. Then those green pixels which are more than some threshold value are masked. An additional step is that the pixels with zeros red, green and blue values and the pixels on the boundaries of the infected cluster were completely removed. [22] starts by capturing images that are processed for enhancement. Then image segmentation using neural networks is carried out to get target regions (disease spots) on the leaves and fruits. Concluding, if the diseased spot on leaf is bordered by yellow margin then it is said that leaf is infected by bacterial blight otherwise not. [23] uses image preprocessing including RGB to differentiate color space conversion, image enhancement, segment the region of interest using K-mean clustering for statistical usage to determine the defect and severity areas of plant leaves, feature extraction and classification. Finally classification is achieved using SVM. [26] uses a pre-trained convolutional neural network using 1.8 million images and used a fine-tuning strategy to transfer learned recognition capabilities from general domains to the specific challenge of Plant Identification task. This work uses various transfer learning architectures and shows how hyperparameter tuning can be done to achieve the best results. Our work builds on top of this and we demonstrate state of the art results for this particular problem.

2.1 Image Classification

Image classification is one of the core problems in computer vision that is used for classifying an image according to its visual content. For example, an image classification algorithm may be designed to tell if an image contains a human body or not. The general steps involved in image classification is depicted in Fig 1.

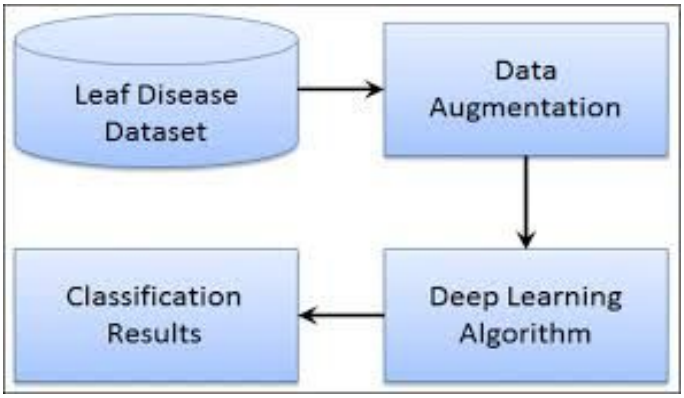

Fig 1. General steps for image classification application

2.2 Deep Learning

Deep learning is a set of machine learning methods that was inspired by information processing and distributed communication in networks of biological neurons present in our brain. Some of the applications of deep learning are self driving cars to recognize a stop sign, or to distinguish a pedestrian from a lamppost, classifying medical images whether tumour is benign or malignant. In deep learning, the algorithm learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, which is better than any other algorithm and even exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers of neurons.

2.3 Convolutional Neural Networks

CNN has Convolutional layers, ReLU layers, Pooling layers and a Fully connected layer. A CNN is used for convolving learned features using filters with input data and uses 2D convolutional layers. This means that this type of neural network is ideal for processing 2D images. The main advantage of using CNNs is that it uses very little image preprocessing. This means that they can learn the filters that have to be designed by ourselves in other traditional machine learning algorithms. CNNs are used in a lot of applications like image and video

recognition, image classification, and recommender systems, natural language processing and medical image analysis.

CNNs have an input layer, output layer, and hidden layers. The hidden layers usually consist of convolutional layers, ReLU layers, pooling layers, and fully connected layers. Convolutional layers apply a convolution operation to the input which is taking a dot product of the filter weights with the weights present in the input image. This passes the information on to the next layer. Pooling layer is used for combining the outputs of neurons which are present in clusters into a single neuron in the next layer. Fully connected layers are used for connecting every neuron in the previous layer to every neuron in the next layer. In a convolutional layer, neurons only receive input from a subregion of the previous layer.

CNN works by extracting features from images. This eliminates the need for manual feature extraction. The features are not trained. The weights are learned while the neural network trains on a set of images. This makes deep learning algorithms extremely accurate for computer vision applications. CNNs learn feature extraction through hundreds of thousands of hidden neurons. Each layer successively increases the complexity of the learned features and thus the algorithm goes on from learning edges and blobs to full objects present in the image. The convolutional neural network architecture is shown in Fig 2.

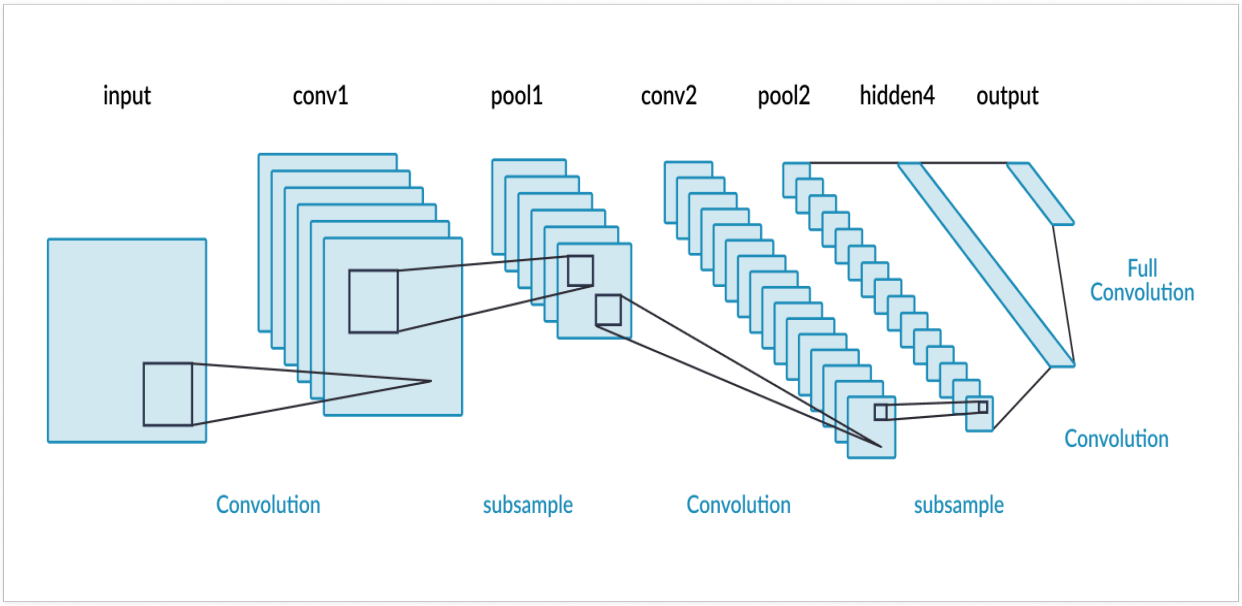

Fig 2. Convolutional Neural Networks architecture

2.4 Plant Disease Detection

Disease detection in plants plays an important role in agriculture as farmers have often to decide whether the crop they are harvesting is good enough. It is of utmost importance to take these seriously as it can lead to serious problems in plants due to which respective product quality, quantity or productivity is affected. Plant diseases cause a periodic outbreak of diseases leading to large-scale death which severely affects the economy. These problems need to be solved at the initial stage, to save the lives and money of people. Automatic classification of plant diseases is an important research topic as it is important in monitoring large fields of crops and at a very early stage if we can detect the symptoms of diseases when they appear on plant leaves. This enables computer vision algorithms to provide image-based automatic inspection. Comparatively, manual identification is labor intensive, less accurate and can be done only in small areas at a time. By this method, the plant diseases can be identified at the initial stage itself and the pest and infection control tools can be used to solve pest problems while minimizing risks to people and the environment.

2.5 DATASET

A public dataset is provided which contains 54,305 images of diseased and healthy plant leaves collected under controlled conditions. The images cover 14 species of crops, including: apple, blueberry, cherry, grape, orange, peach, pepper, potato, raspberry, soy, squash, strawberry and tomato. It contains images of 17 basic diseases, 4 bacterial diseases, 2 diseases caused by mold, 2 viral diseases and 1 disease caused by a mite. Each class label is a crop-disease pair, and we make an attempt to predict the crop-disease pair given just the image of the plant leaf. Figure 3 shows all the classes present in the PlantVillage dataset.

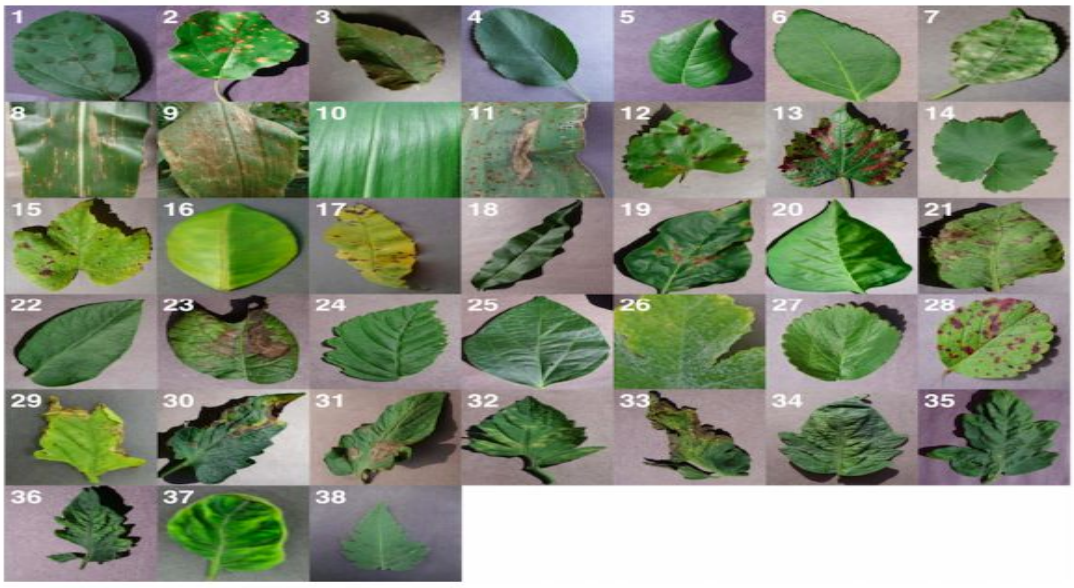

Fig 3. All the classes of plant disease present in dataset

3. PROPOSED METHOD

We have used the concept of transfer learning for the classification. The main advantage in using transfer learning is that instead of starting the learning process from scratch, the model starts from patterns that have been learned when solving a different problem which is similar in nature to the one being solved. This way the model leverages previous learnings and avoids starting from scratch. In image classification, transfer learning is usually expressed through the use of pre-trained models. A pre-trained model is a model that was trained on a large benchmark dataset to solve a similar problem to the one that we want to solve. We used five pre-trained models- Inception v3, InceptionResNet v2 and ResNet50, MobileNet and DenseNet169 as the pre-trained weights for our work.

3.1 Inception v3

Google's Inception v3 architecture was re-trained on our dataset by fine-tuning across all layers and replacing top layers with one average pooling, two fully connected and finally the softmax layer allowing to classify 2 diagnostic categories. The size of all the input images was resized to (224, 224) as expected by the model. Learning rate was set to 0.0001 and Adam was used for the optimizer. Fig 5 shows the architecture of Inception v3.

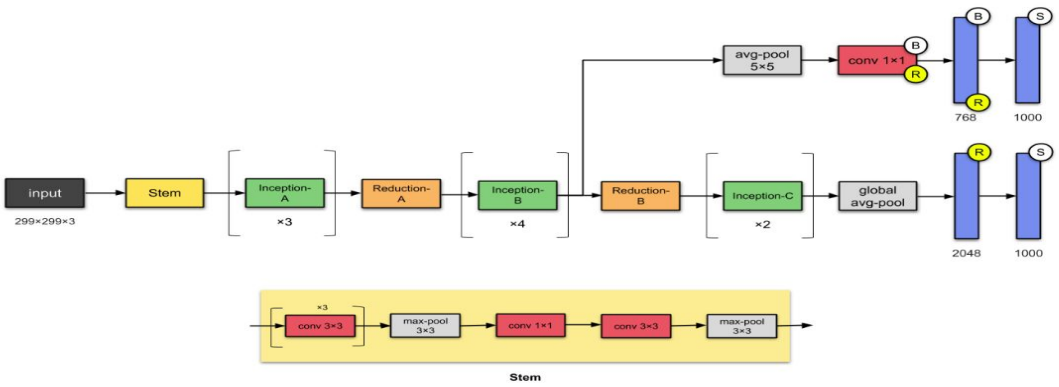

Fig 4. Inception v3 architecture

3.2 InceptionResNet v2

InceptionResNet v2 architecture was re-trained on our dataset by fine-tuning across all layers and replacing top layers with one global average pooling, one fully connected and finally the softmax layer allowing to classify 2 diagnostic categories. The size of all the input images was resized to (224, 224) as expected by the model. Learning rate was set to 0.0001 and Adam was used for the optimizer. Fig 6 shows the architecture of InceptionResNet v2.

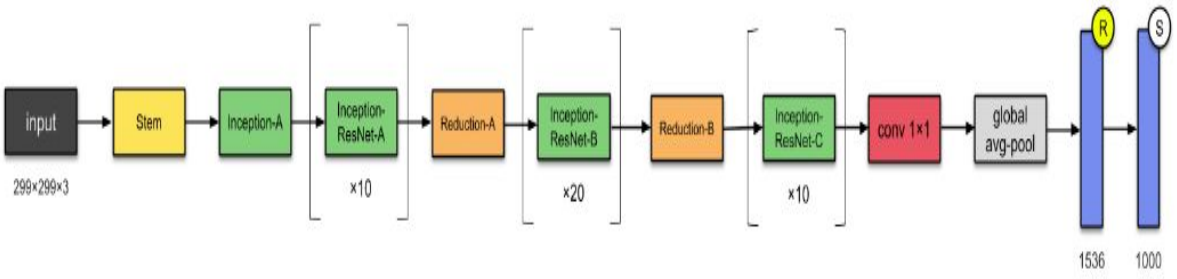

Fig 5. InceptionResNet v2 architecture

3.3 ResNet 50

ResNet50 architecture was re-trained on our dataset by fine-tuning across all layers and replacing top layers with one average pooling, one fully connected and finally the softmax layer allowing to classify 2 diagnostic categories. The size of all the input images was resized to (224, 224) as expected by the model. Learning rate was set to 0.0001 and Adam was used for the optimizer. Fig 7 shows the architecture of ResNet50.

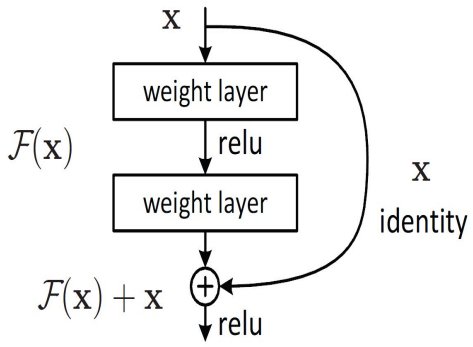

Fig 6. Residual layer

It uses identity mapping to map the inputs. This identity mapping as shown in Fig 7 does not have any parameters whose only function is to add the output from the previous layer to the layer ahead. The identity mapping is multiplied by a projection which is linear in order to expand the channels of shortcut to match the residual. This allows for the input x and F(x) to be combined as input to the next layer. A residual block with identity mappings is illustrated in Fig 7.

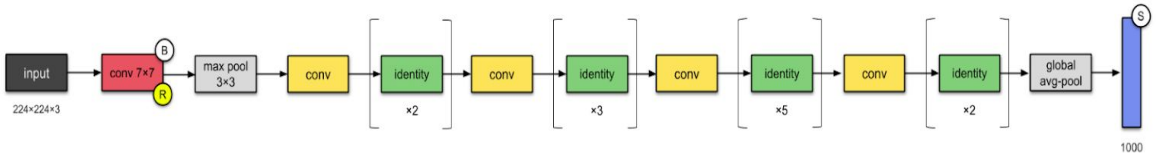

Fig 7. ResNet50 architecture

The skip connections which are present between the layers add the outputs from previous layers to the outputs of layers which are stacked on top of it. This results in the ability to train much deeper networks than what was previously possible.

3.4 MobileNet

The main difference in mobileNet and other architectures is that it uses depthwise separable filters, named as Depthwise Separable Convolution. These convolutions layers which is a form of factorized convolutional factorize a standard convolution into a depthwise convolution called a pointwise convolution. In MobileNet, the depthwise convolution applies a single filter to each input channel. The pointwise convolution then applies the convolution operation to combine the outputs of the depthwise convolution.Fig 9 shows the architecture of MobileNet.

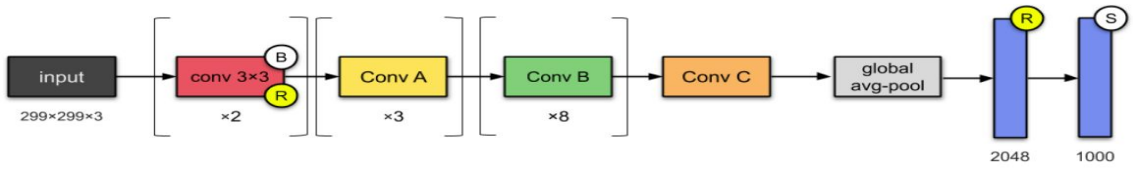

Fig 8. Mobilenet architecture

3.5 DenseNet169

To solve the vanishing gradient problem, this architecture uses a simple connectivity pattern to ensure the maximum flow of information between layers both in forward and backward computation. The layers are connected in a way such that inputs from all preceding layers passes through its own feature-maps to all subsequent layers. To facilitate the down-sampling in the architecture, the entire architecture is divided into multiple densely connected blocks. The layers between these dense blocks are transition layers which perform convolution and pooling operations. Fig 10 shows the architecture of DenseNet169.

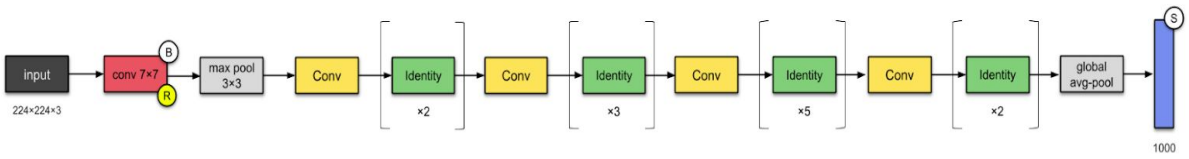

Fig 9. DenseNet169 architecture

3.6 Image augmentation techniques

The images are resized to 256 x 256 pixels, and we perform both the model optimization and predictions on these downscaled images. We used data augmentation like shearing, zooming, flipping and brightness change to increase the dataset size to almost double the original dataset size. Data augmentation techniques are often used together with traditional machine learning algorithms or deep learning algorithms to improve the accuracy of classification. In this study, the image augmentation method was used by using the Keras deep learning library in Python. Width and height change, cutting, zooming, horizontal turning, brightness, and filling operations were performed for normal class images. The image rotation degree was set to be randomly generated from 0 to 45.

In this study, the image augmentation techniques were applied only to normal images in order to balance the distribution of the samples over the classes. The number of normal samples in the dataset was increased from 1,583 to 4,266 by performing the image augmentation techniques. In this manner, the number of samples for each class was equalized. This equal distribution makes it possible to use all of the data instead of selecting random data during
the training process. It is expected that this situation increases the accuracy of the training and positively affects the classification results. Image augmentation techniques are shown in Fig 11.

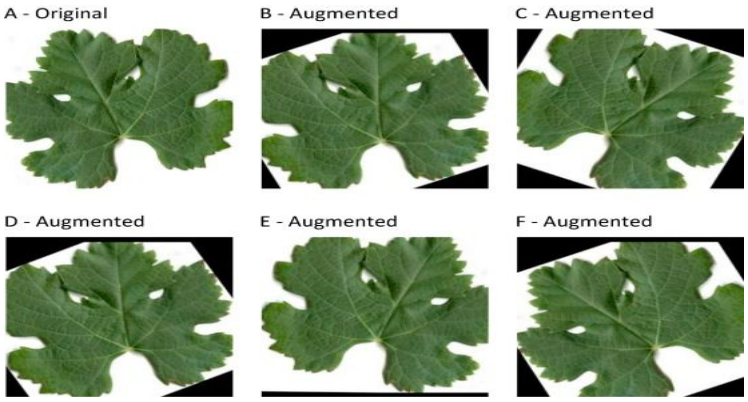

Fig 10. Data augmentation

3.7 Optimization method

The main purpose of optimization methods is to update the weights at every stage until the best learning in CNN is realized. Each method performs an update process. In the Stochastic Gradient Descent (SGD) method, the weights update is performed in every iteration for each instance present in the training set. Because of this reason, it tries to achieve the goal as early as possible.

3.8 Dropouts

Dropout is one of the common regularization techniques to prevent neural networks from overfitting. Others often used regularization methods like L1 and L2 to reduce overfitting by penalizing the cost function. Dropout on the other hand, modifies the network itself by randomly dropping neurons from the neural network during training in each iteration. When we drop different sets of neurons, it's equivalent to training an ensemble of neural networks and thus it helps in reducing variance. The different neural networks will overfit in different ways, so the net effect of dropout will be to reduce overfitting. Fig 12 shows how neural network architecture is changed after using dropouts.

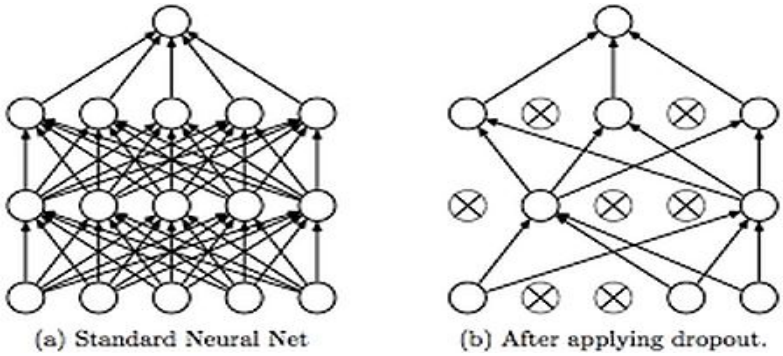

Fig 11. Dropouts

3.9 Visualization of Feature Maps

The feature maps help in explaining what the model is learning at every layer. As the depth increases, the model is able to learn more spatial information. In other words, the neural networks go from learning edges and blobs in the first layer to complete objects in the last layers.

Visualization of feature maps is important to understand what the filters are learning at each layer. The hyperparameter tuning is easier because when an error is made by the neural network we can get the reason for going wrong. The functionality and expected behaviour of the neural networks can be explained especially to non-technical stakeholders who wouldn't accept deep learning algorithms results until there is a reasoning behind them. This also makes extending and improving the overall design of models since we'd have knowledge of the current design, including how it performs.

By visualising the learned weights we can get some idea as to how well our network has learned. For example, if we see a lot of zeros then we'll know we have many dead filters that aren't going much for our network, a great opportunity to do some pruning for model compression. Fig 13-18 shows the activation maps for the filters present in four convolution and four max pooling layers.

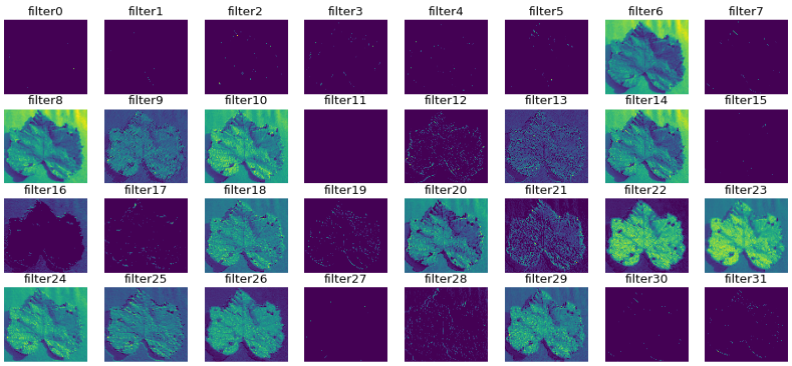

Fig 12. Visualization of activations in the first convolution layer

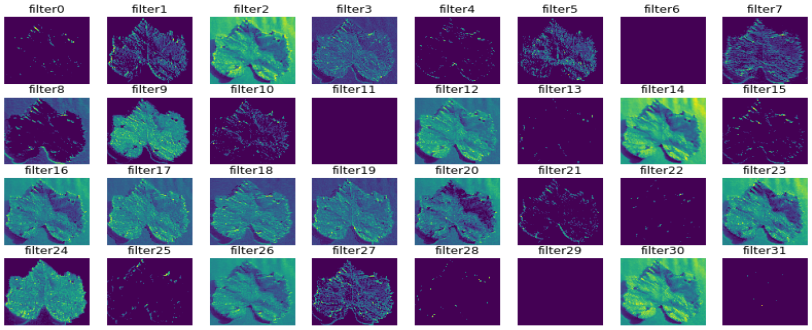

Fig 13. Visualization of activations in the first pooling layer

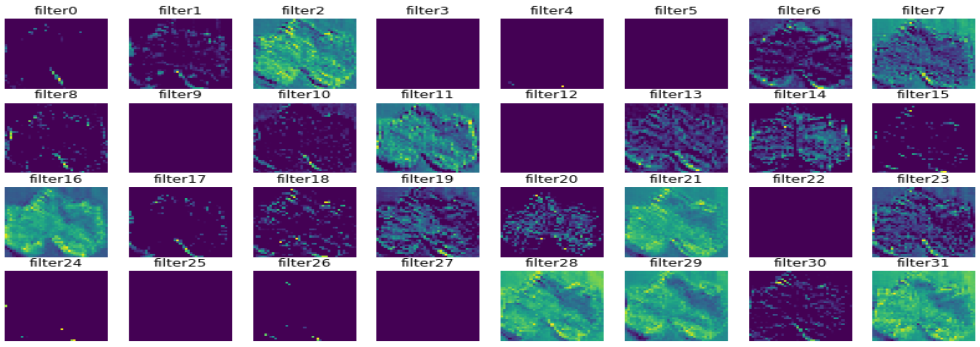

Fig 14. Visualization of activations in the second convolution layer

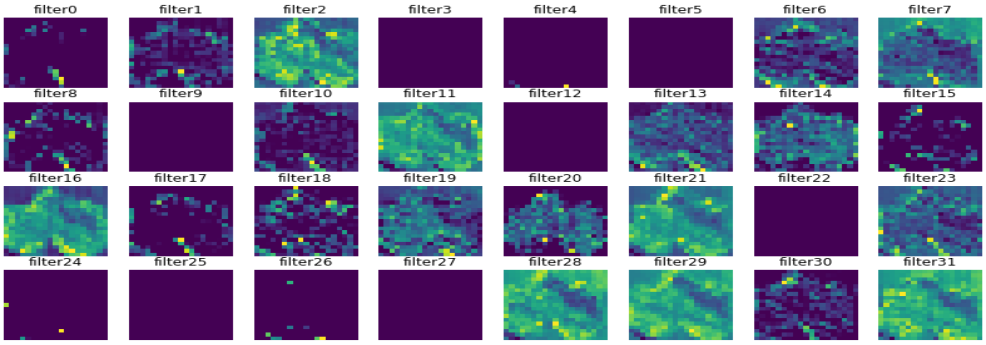

Fig 15. Visualization of activations in the second pooling layer

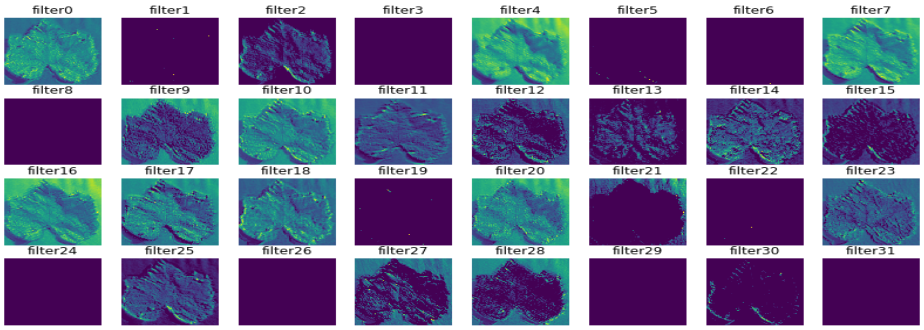

Fig 16. Visualization of activations in the third convolution layer

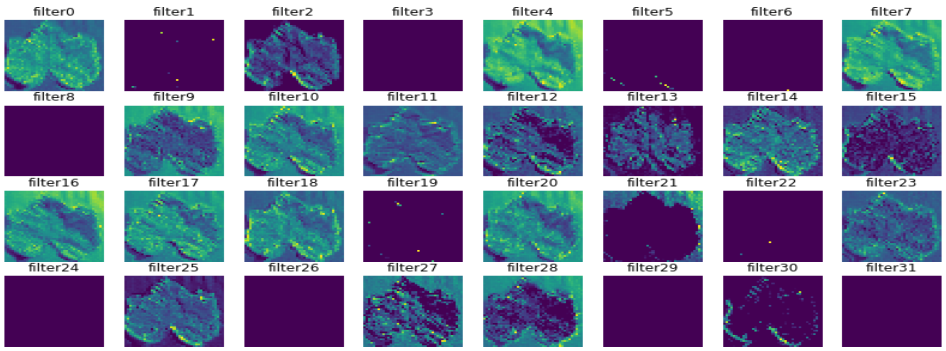

Fig 17. Visualization of activations in the third pooling layer

Neural networks are often thought of as black boxes. In this way when we deploy the model in production, the feature maps come in handy as the non technical people like stakeholders, business people, doctors etc often don't understand what neural network does behind the scenes. This makes it easy for them to get convinced and accept the results.

We used two additional techniques while training ModelCheckpoint and EarlyStopping.

ModelCheckpoint is used when training requires a lot of time to achieve a good result, often many iterations are required. In this case, it is better to save a copy of the best performing model only when an epoch that improves the metrics ends.

Sometimes during training we can notice that the generalization gap i.e. the difference between training and validation error starts to increase, instead of decreasing. This is a symptom of overfitting that can be solved in many ways (reducing model capacity, increasing training data, data augmentation, regularization, dropout, etc). Often a practical and efficient solution is to stop training when the generalization gap is getting worse. This process is known as early stopping as is illustrated in Fig 19.

Fig 18. Early stopping illustrated

We split the data-set into three sets — train, validation and test sets. We tried with pre trained models like Inception v3, InceptionResNet v2 and ResNet 50, MobileNet and Densenet169 by fine tuning the last layers of the network.
On top of the transfer learning architectures, we have added 4 custom convolutional and max pooling pooling layers. We used two dense layers with 64 neutrons and 2 neurons respectively at the last. The last layer is used for the classification with softmax as the activation function. The loss function used is binary cross-entropy. We trained the model for 20 epochs with a batch size by changing the hyper-parameters like learning rate, batch size, optimizer and pre-trained weights. We used 30% dropouts to reduce overfitting in between the layers and batch normalization to reduce internal covariate shift. This also helped the model avoid getting stuck in the local optimum or a saddle point. Multi class log loss was chosen as the evaluation metric. Activation function used was Relu throughout except for the last layer where it was Sigmoid as this is a binary classification problem.

Further we made a couple of data generators: one for training data, and the other for testing data. A data generator is capable of loading the required amount of data (a mini batch of images) directly from the source folder, converting them into training data and training targets.

To enable a fair comparison between the results of all the experimental configurations, we also tried to standardize the hyperparameters across all the experiments, and we used the following hyperparameters in all of the experiments

1. Base learning rate: 0.001
2. Learning rate policy: Step
3. Momentum: 0.9
4. Weight decay: 0.0005
5. Gamma: 0.1
6. Batch size: 32
7. Optimizer: Adam

4. EXPERIMENTAL RESULTS

In this section we present our findings. We plotted the loss vs epochs, accuracy vs epochs and confusion matrix for the classifier. The loss vs epochs, accuracy vs epochs figure is shown in Fig 20 and Fig 21 respectively.

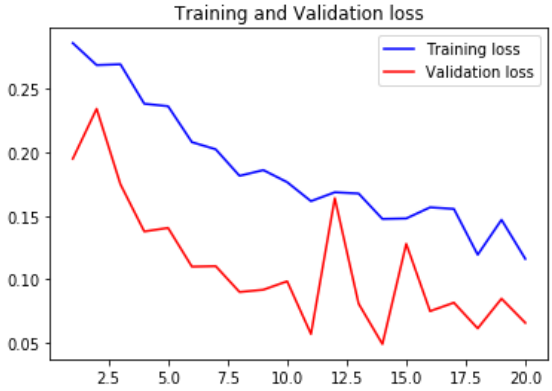

Fig 19. Loss vs epochs

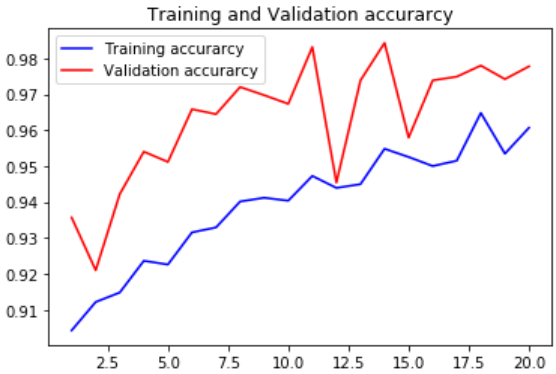

Fig 20. Accuracy vs epoch

We chose Multi Class Log Loss evaluation function, which is defined as :

$$L = -\frac{1}{N}\sum_{i}^{N}\sum_{j}^{M} y_{ij} \cdot Ln(p_{ij})$$

- *N* is the number of classes present in the test set
- *M* is the number of class labels which is 38 in this case
- *y ij* depicts if the i-th object in the test set belongs to the j-th label which is given by a boolean value..
- *p ij* is the probability that the i-th object belongs to the j-th label.
- *Ln* is the natural logarithmic function.

For a better look at misclassification, we often use the following metric to get a better idea - true positives (TP), true negatives (TN), false positive (FP) and false negative (FN). Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. Recall is the ratio of correctly predicted positive observations to all the observations in actual class. F1-Score is the weighted average of Precision and Recall.

In this paper, a total of 19 types of plant disease categories are used. The evaluation and results of trained models is calculated by common classification metrics which are mathematically illustrated in Fig 22.

$$\begin{aligned} precision &= \frac{TP}{TP + FP} \\ recall &= \frac{TP}{TP + FN} \\ F1 &= \frac{2 \times precision \times recall}{precision + recall} \\ accuracy &= \frac{TP + TN}{TP + FN + TN + FP} \\ specificity &= \frac{TN}{TN + FP} \end{aligned}$$

Fig 21. Evaluation metrics

The acronyms, TP, TN, FP, FN refer to true positive, true negative, false positive and false negative. We also used F1 for evaluation which combined both precision and recall in a single term. The higher the F1-Score, the better the model. For all three metrics, 0 means the model is performing the worst while 1 means it is performing the best. The precision, recall, F1 and support values for all the individual classes is shown in Fig 23. The average accuracy and weighted class wise accuracy is found to be 0.91 and 0.94 respectively.

	precision	recall	f1-score	support
0	1.00	0.77	0.87	22
1	1.00	1.00	1.00	17
10	1.00	1.00	1.00	44
11	0.96	0.98	0.97	44
12	0.98	0.96	0.97	46
13	1.00	1.00	1.00	44
14	0.93	0.93	0.93	15
15	1.00	1.00	1.00	233
16	1.00	0.98	0.99	88
17	0.67	1.00	0.80	4
18	0.91	0.80	0.85	25
19	0.93	0.92	0.92	60
2	1.00	0.92	0.96	12
20	0.96	1.00	0.98	44
21	0.92	0.82	0.87	40
22	0.17	0.33	0.22	3
23	0.97	0.94	0.96	36
24	0.94	1.00	0.97	179
25	0.99	1.00	0.99	75
26	1.00	1.00	1.00	38
27	0.83	1.00	0.91	20
28	0.95	0.79	0.86	92
29	0.91	0.28	0.43	36
3	0.92	0.91	0.91	53
30	0.95	0.84	0.89	73
31	0.81	0.93	0.87	28
32	0.78	0.90	0.84	62
33	0.84	1.00	0.91	68
34	0.78	0.96	0.86	51
35	0.96	1.00	0.98	220
36	0.93	0.93	0.93	15
37	1.00	0.93	0.96	67
4	0.94	1.00	0.97	45
5	1.00	0.96	0.98	45
6	1.00	0.97	0.99	40
7	0.86	0.60	0.71	20
8	0.92	1.00	0.96	46
9	0.91	0.93	0.92	44
accuracy			0.94	2094
macro avg	0.91	0.90	0.90	2094
weighted avg	0.94	0.94	0.94	2094

Fig 22. Evaluation metrics results

Next we plotted the confusion matrix for all the pairs having and not having disease. Confusion Matrix is often used as an evaluation metric when analyzing misclassification between classes. Each row of the matrix represents the examples in the class being predicted while each column represents the examples in the class which are original. The diagonals show the classes which have been classified correctly. This tells us both which classes are being misclassified and also what they are being misclassified as. The class wise confusion matrix is shown in Fig 24. Each of these contain a pair both having the disease and not having one.

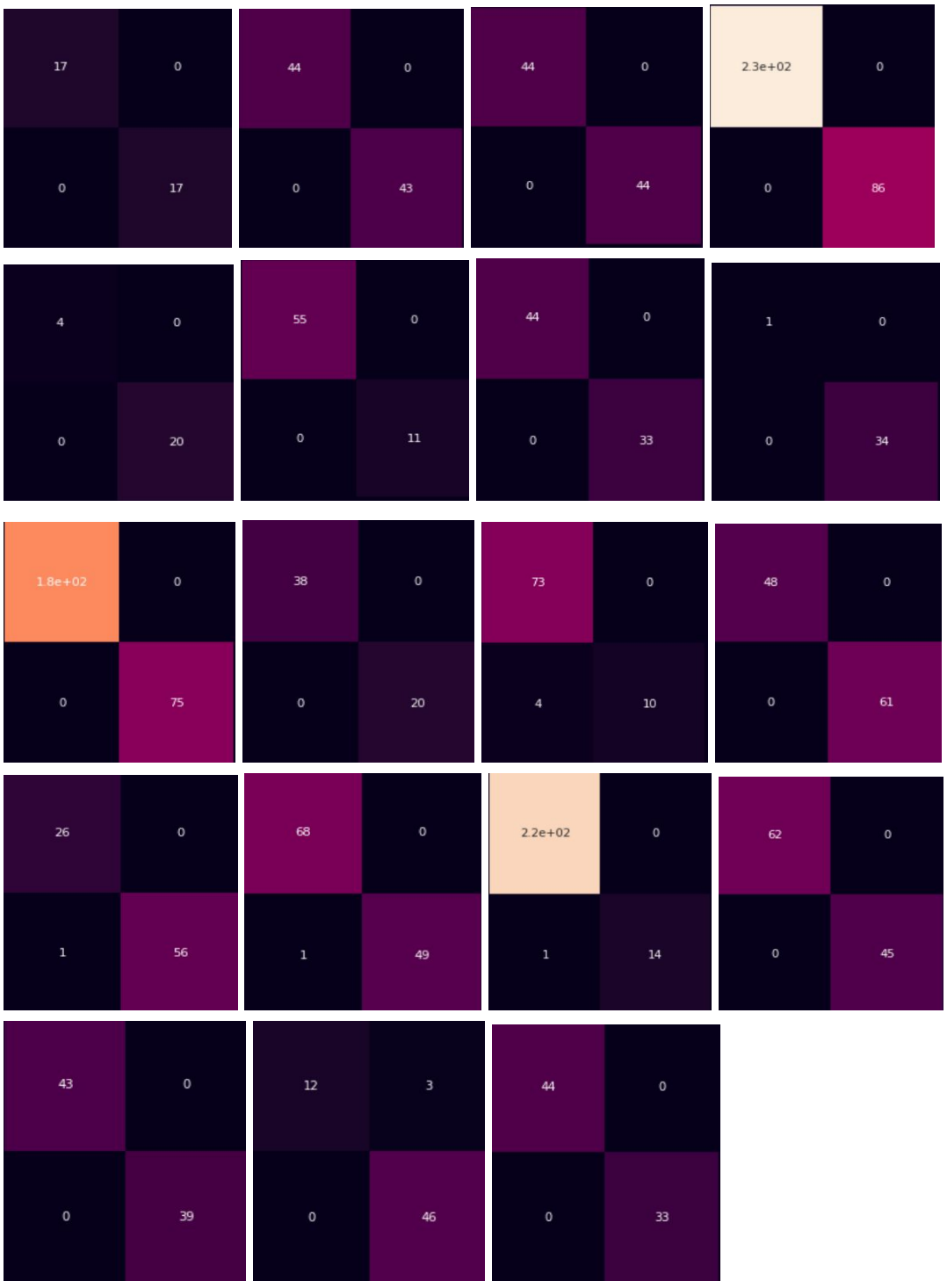

Fig 23. Confusion Matrix

Finally we present our final results i.e. comparing accuracy, precision, recall and F1 score for all the transfer learning architectures used - Inception v3, InceptionResNet v2 and ResNet50, MobileNet and DenseNet169. This is shown in Fig 25.

	InceptionV3	InceptionResNetV2	ResNet50	MobileNet	DenseNet169
Accuracy	0.971	0.978	0.982	0.971	0.974
Precision	0.92	0.91	0.94	0.94	0.92
Recall	0.94	0.93	0.94	0.93	0.93
F1	0.93	0.92	0.94	0.93	0.93

Table 1. Accuracy, Precision, Recall F1 score values

5. CONCLUSION

Diseases in plants are a major threat to food supply worldwide. This paper demonstrates the technical feasibility of deep learning using convolutional neural network approach to enable automatic disease diagnosis through image classification. Using a public dataset of 54,306 images of diseased and healthy plant leaves, a deep convolutional neural network is trained to classify crop species and disease status of 38 different classes containing 14 crop species and 26 diseases, achieving an accuracy of 98.2% with residual network architecture. In this paper, a new approach of using deep learning methods was explored in order to automatically classify and detect plant diseases from leaf images. The developed model was able to distinguish between healthy leaves and different diseases, which can be visually diagnosed. The complete procedure was described, respectively, from collecting the images used for training and validation to image augmentation and finally the procedure of training the deep CNN and fine-tuning. We summarized the final results and came to the conclusion that ResNet50 achieves the highest accuracy as well as precision, recall and F1 score.

6. REFERENCES

[1] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[2] G. Litjens et al., "A survey on deep learning in medical image analysis," vol. 42, pp. 60-88, 2017.

[3] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in CVPR, 2017, vol. 1, no. 2, p. 3.

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,pages 1–9,2015.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Identity Mappings in Deep Residual Networks. arXiv preprint arXiv:1603.05027v3,2016.

[6] Xu, L., Ren, J. S., Liu, C., & Jia, J. (2014). Deep convolutional neural network for image deconvolution. In Advances in neural information processing systems (pp. 1790-1798).

[7] Qayyum, A., Anwar, S. M., Awais, M., & Majid, M. (2017). Medical image retrieval using deep convolutional neural network. Neurocomputing, 266, 8-20.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385,2015.

[9] Kamavisdar, P., Saluja, S., & Agrawal, S. (2013). A survey on image classification approaches and techniques. International Journal of Advanced Research in Computer and Communication Engineering, 2(1), 1005-1009.

[10] Agarap, A. F. (2018). Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375.

[11] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.

[12] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[13] A. Veit, M. Wilber and S. Belongie. Residual Networks Behave Like Ensembles of Relatively Shallow Networks. arXiv:1605.06431v2,2016.

[14] G. Huang, Z. Liu, K. Q. Weinberger and L. Maaten. Densely Connected Convolutional Networks. arXiv:1608.06993v3,2016.

[15] G. Huang, Y. Sun, Z. Liu, D. Sedra and K. Q. Weinberger. Deep Networks with Stochastic Depth. arXiv:1603.09382v3,2016.

[16] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. In 2016 IWWW Conference on Computer Vision and Pattern Recognition, pp. 2818–2826.

[17] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? NIPS'14 Proceedings of the 27th International Conference on Neural Information Processing Systems *2*, 3320–3328.

[18] P.Revathi, M.Hema Latha, Classification Of Cotton Leaf Spot Diseases Using Image Processing Edge Detection Techniques , ISBN, 2012, 169-173, IEEE.

[19] Santanu Phadikar & Jaya Sil[2008] Rice Disease Identification Using Pattern Recognition Techniques, Proceedings Of 11th International Conference On Computer And Information Technology

[20] Geng Ying, Li Miao, Yuan Yuan &Hu Zelin[2008] A Study on the Method of Image PreProcessing for Recognition of Crop Diseases, International Conference on Advanced Computer Control, 2008

[21] H. Al-Hiary, S. Bani-Ah Mad, M. Reyalat, M. Braik And Z. A Lrahamneh, Fast And Accurate Detection And Classification Of Plant Diseases, IJCA, 2011

[22] Tejal Deshpande, Sharmila Sengupta, and K.S.Raghuvanshi, "Grading & Identification of Disease in Pomegranate Leaf and Fruit," IJCSIT

[23] Ms. Kiran R. Gavhale, Prof. Ujwalla Gawande, and Mr. Kamal O. Hajari, "Unhealthy Region of Citrus Leaf Detection using Image Processing Techniques," IEEE International Conference on Convergence of Technology (I2CT)

[24] P. Revathi and M. Hemalatha, "Identification of cotton diseases based on cross information gain_deep forward neural network classifier with PSO feature selection," *International Journal of Engineering and Technology*, vol. 5, no. 6, pp. 4637–4642, 2014.

[25] Y. Jia, E. Shelhamer, J. Donahue et al., "Caffe: convolutional architecture for fast feature embedding," in *Proceedings of the ACM Conference on Multimedia (MM '14)*, pp. 675–678, ACM, Orlando, Fla, USA, November 2014.

[26] A. K. Reyes, J. C. Caicedo, and J. E. Camargo, "Fine-tuning deep convolutional networks for plant recognition," in *Proceedings of the Working Notes of CLEF 2015 Conference*, 2015

[27] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings of the 27th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '14)*, pp. 1717–1724, IEEE, Columbus, Ohio, USA, June 2014.

[28] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: delving deep into convolutional nets,"

[29] Mohanty, S.P., Hughes, D.P., Salathé, M.: Using deep learning for image-based plant disease detection. Front. Plant Sci. 7 (2016). Article: 1419

[30] Ferentinos, K.P.: Deep learning models for plant disease detection and diagnosis. Comput. Electron. Agric. 145, 311–318 (2018).

[31] Cortes E. Plant Disease Classification Using Convolutional Networks and Generative Adversarial Networks. Stanford University Reports, Stanford (2017)

[32] Fuentes, A., Yoon, S., Kim, S.C., Park, D.S.: A robust deep-learning-based detector for realtime tomato plant diseases and pest recognition. Sensors 17(9), 2022 (2017)

[33] Ramcharan, A., Baranowski, K., McCloskey, P., Ahmed, B., Legg, J., Hughes, D.: Deep learning for image-based Cassava disease detection, frontiers in plant science. Front. Plant Sci. 8, 1852(2017)

[34] Lua, J., Hua, J., Zhaoa, G., Meib, F., Zhanga, C.: An in-field automatic wheat disease diagnosis system. Comput. Electron. Agric. 142PA, 369–379 (2017)

[35] Ronnel, R., Atole, A.: Multiclass deep convolutional neural network classifier for detection of common rice plant anomalies. Int. J. Adv. Comput. Sci. Appl. (IJACSA) 9(1) (2018)