Comparative Study on Reliability of Transfer Learning to Classify Plant-Based Diseases

N Rohan Sai, T Sudarshan Rao, G. L. Aruna Kumari

***Abstract:** One of the essential factors contributing to a plant's growth is identifying and preventing diseases in the early stages. Healthy plants are essential for a rich production. Recent advances in Deep learning - a subset of Artificial Intelligence and Machine Learning are playing a pivotal role in solving image classification problems and can be applied to the agricultural sector for crop surveillance and early anomaly identification. For this research, we used an open-source dataset of leaf images divided into three classes, two of which are the most common disease types found on many crops; the graphical characterizations for the three classes are images of leaves with Powdery Residue, images of leaves with Rusty Spots, and images of Healthy leaves. The primary objective of this research is to present a pre-trained ImageNet network architecture that is well suited for dealing with plant-based data, even when sample sizes collected are limited. We used different convolutional neural network-based architectures such as InceptionV3, MobileNetV2, Xception, VGG16, and VGG19 to classify plant leaf images with visually different representations of each disease. Xception, MobileNetV2, and DenseNet had a considerable advantage over all the performance metrics recorded among the other networks trained.*

***Keywords**: Artificial Intelligence, Deep Learning, Neural Networks, Transfer Learning*

I. INTRODUCTION

Fungi, which are the root cause for diseases in plants, take their energy from healthy green plants or decomposed material. Their microscopic entities, which make them invisible to the naked eye, cause a great deal of damage to the crops, disturbing production. Most observed conditions include mildews (rusty, powdery, downy, etc.) that can be found both on surfaces and underneath areas on an infected plant. Early prevention is one of the very few ways that help in saving crops from these deadly fungal maladies. Several techniques are used to tackle this problem, such as using fertilizers and changing irrigation methods, which come into play only after the infection is identified. Infected plants have higher chances of survival in the long run when their infections are identified in the early stages. In this paper, we addressed this problem with a computational approach using Deep Learning - a subset of Artificial Intelligence that involves architectures inspired by the operation of the nervous system of the human brain. In Deep Learning, neural networks perceive an image as a nested hierarchy of features, and each feature is broken down into a simpler feature. The representation that is more abstract is calculated with reference to the representation that is less abstract, thus achieving powerful functions and flexibility. Deep learning specializes in solving closed-end classification problems where a vast number of potential signals must be projected onto a small number of groups. On the other hand, it is crucial to recognize the issues that Deep learning is not effective at.

Manuscript received on August 19, 2021.
Revised Manuscript received on August 23, 2021.
Manuscript published on August 30, 2021.
* Correspondence Author

N Rohan Sai*, Computer Science and Engineering, GITAM (Deemed to University), Visakhapatnam, India. Email: rohansai1186@gmail.com

T Sudarshan Rao, Computer Science and Engineering, GITAM (Deemed to University), Hyderabad, India, Email: sudarshantadakamalla@yahoo.com

G. L. Aruna Kumari, Assistant Professor, Computer Science and Engineering, GITAM (Deemed to University), Visakhapatnam, India. Email: agorli@gitam.edu

For image classification, Convolutional Neural Networks (CNN) are used to effectively consider all the 2-dimensional pixel data without having to deal with parameters explosion, which is evident when we use feed-forward networks with image data [1]**.** CNN's generally attain a good accuracy when the data collected is diverse with large numbers of samples. The typical observation with shallow networks like LeNet5 [20] is that they do not perform well with only a small data set. Deep learning demands more significant amounts of data, but with recent progress in the knowledge we have about the task-specific architectural advantage of the neural networks, research on many custom architectures came into consideration. Starting from the LeNet5, the successors AlexNet with 50 million parameters [21], VGG16 & 19 with 138 million parameters [4] considered denser network architectures with more parameter-leverage during training. This later lead to problems like *Vanishing & Exploding Gradients.* Later, ResNet with residual units -- skip connections -- showed a novel technique that addressed the *Vanishing gradient* problem. Observing this architectural benefit, the transfer learning approach has gained traction; we use the same models trained on millions of images and use them for our problem statement with the same parameters intact. Transfer Learning with the pre-trained networks available is the fundamental approach that we used in our research, with the observation being on the comparative analysis of performance metrics between the different pre-trained networks available and standard convolutional neural network dealing with plant-based data. The results section has a detailed edification about the analysis.

II. RELATED WORK

Since the inception of deep learning research on task-specific networks, their applications are integrated into various domains. The agricultural sector showed a prominent future for deep learning approaches to maintain the fundamentals requirements with technology. A demanding role in agriculture is to constrain the festering diseases to stop crop loss. There is a lot of research work addressing this application.

In [22] by Shreya Ghosal and Kamala Sarkar, the main focus was to discover the diseases in rice leaves and classify the diseases into different categories using a deep learning architecture. The proposed architecture was a pre-trained VGGNet model, which was fine-tuned to perform with an accuracy of 92.46% on 647 test images. It is also stated that the CNN without transfer learning underperformed with an accuracy of 74%, which the proposed transfer learning model outperformed.

Rajashekaran Thangaraj, S. Anandamurugan, Vishnu Kumar Kaliappan, [23] used a transfer learning-based deep convolutional neural network model depending on the characteristics of the disease, such as viruses that caused the infection, types of spots on the leaves, types of blights. The TL-based Xception model proposed with RMSprop as its optimizer achieved an accuracy of 99.55%.

In [24] Ümit ATİLA, Murat UÇAR, Kemal AKYOL, Emine UÇAR, the primary mentioned goal was to compare the performance of EfficientNet deep learning architecture with that of other state-of-the-art TL-based CNN models for the classification of plant leaf diseases.

Junde Chen, Jinxiu Chen, Defu Zhang [25] addressed plant disease detection using VGGNet architecture. The suggested method was divided into two parts: a pre-trained module that serves as a fundamental feature extractor, and the second is an auxiliary structure that detects multi-scale feature maps. The transfer learning for DCNNs was examined in this research with a new DL architecture, i.e., INC-VGGN, to identify plant diseases, which achieved a validation accuracy of 91.83%.

III. METHODOLOGY

Using pre-trained networks loaded with ImageNet weights, we applied the concept of transfer learning to classify the images. We further state the evidence of comparative performance outcomes between the existing pre-trained architectures. The fundamental advantage of transfer learning is that it eliminates the empirical approach, naturally followed, on a typical convolutional neural network built from scratch. With Transfer Learning, the model's previous knowledge of a vast benchmarked dataset with thousands of classes will become relevant to the problem at hand, this way, the model instantly attains high accuracy scores, which is normally an arduous task for a standard network. Transfer learning is also found to be effective when the sample data collected is not significantly large. The pre-trained custom architectures were observed to be substantial in achieving good performance results. For this research, the pre-trained architectures we used were, MobileNetV2, DenseNet121, VGG16, VGG19, InceptionV3, Xception and a standard Convolutional Neural Network (ConvNet) build from scratch.

A. Data Overview

An open-sourced dataset consists of images of 3 different kinds of image classes of leaves unrelated to any specific plant species where 2 classes represent the most common kind of plant disease. Figure 1 illustrates the random samples of images for the 3 classes. The three classes are 'Healthy,' 'Powdery,' and 'Rust'. Powdery and Rust are the conspicuous disease symptoms commonly observed with many of the crop plants in the agriculture fields. *Pucciniales* and *Erysiphales*, two fungal groups, are responsible for both Rust spots and Powdery Residue on the leaves. Both of these diseases have been linked to high levels of humidity, which is when the fungi are more contagious to the neighboring leaves and shrubs. Commonplace within many high-yielding crops.

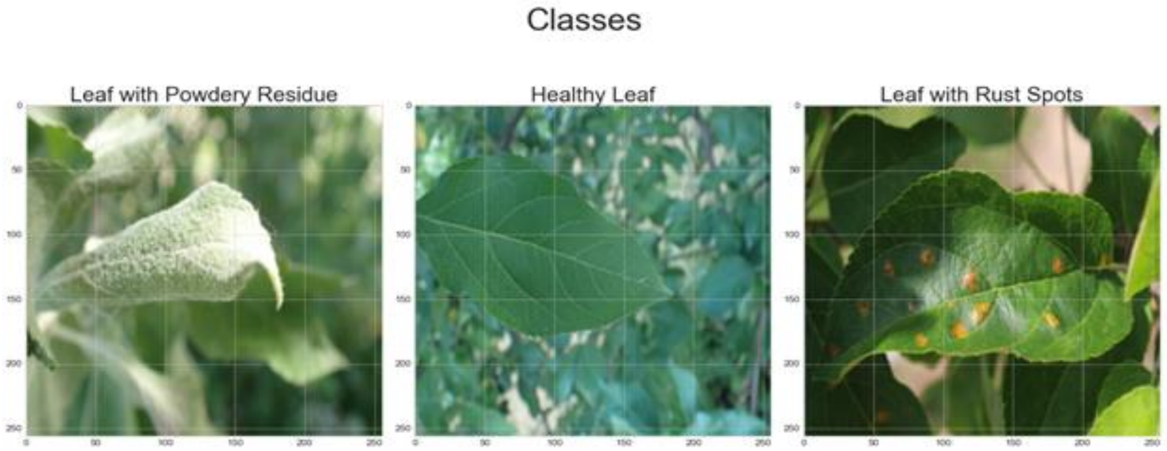

Fig. 1: Images of 3 Classes

B. Data Distributions

The dataset consists of 1532 samples of 3 classes (Healthy, Powdery, and Rust) with relatively equal distributions between the classes. The dataset was split into three distribution Train, Test, and Validations with conventional splits, i.e., 86.3% of the dataset is taken as a Training set Figure 1: Images for 3 Classes (1322 samples). 13.7% is taken for both Validation and Test Sets (210 samples), both of which are drawn out from the same. The mean dimensions of the images were (2300, 2300, 3) where 2300 x 2300 are pixels with 3 color channels. We used image augmentation techniques to *resize* the images to a lower scale of 256 x 256 pixels.

C. Normalization

Each image (pixel) is normalized to a 0-1 scale by dividing it by 255, which is the maximum RGB pixel value. Normalization is a standard step in every machine learning application, in which the features are reduced to a minimum scale after normalization, considerably reducing the model computing requirements during training.

D. Convolutional Neural Network Architecture

Convolutional neural networks are peculiar neural network architectures to deal with Image data in particular [1]. Conventional Feed-Forward neural networks, when applied to problem statements involving images as the primary dataset, the 2D vector of pixels explodes the number of parameters that should be optimized. A basic convolutional neural network architecture involves an input image, kernels, pooling layers for feature extraction, and densely connected outer layers for the actual classifications through probabilistic activation functions like softmax and logistic equations as shown in the Figure 2 [1]. Architectures of convolutional neural networks themselves play a significant role in achieving optimal performance. There is no standard template for an architecture that can be applied to every problem statement. Constructing one from scratch involves empirically finding out what works better by fine-tuning various components of the architecture and the hyper-parameters. This inherent nature of deep learning is known as orthogonality [3]. For our research application, we trained a convolutional neural network with an architecture of 2 Convolution layers with 32 filters of size 3x3 with Relu Activation followed by a Max-Pooling layer with a 2x2 size and a dropout layer with 40% probability. This same module of was repeated 2 times while connected sequentially, with filter sizes of 64 of 3x3 kernel size. The final output module was a densely connected network of 512 nodes of neurons with a dropout of 30% probability and a final output node with 3 units with softmax activation.

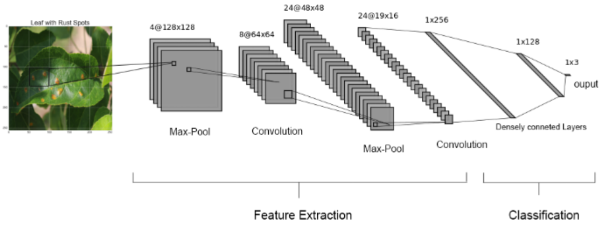

Fig.2: A standard convolutional neural network architecture

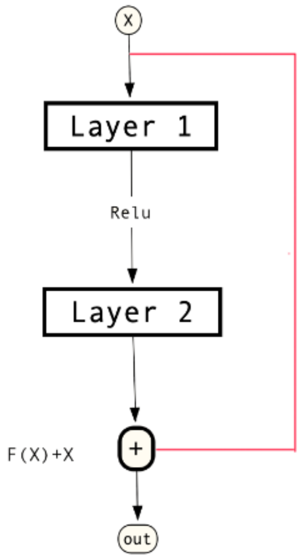

Fig. 3: Residual unit

E. Pre-trained Convolutional Neural Networks

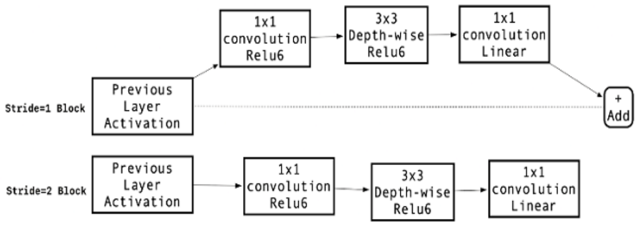

Fig. 4: MobileNetV2 Architecture

The observation on how architecture affects performance facilitated the research on custom neural network architectures. The initial approach to put forward a novel deep learning network that yields an optimal performance metric was to go deeper with the network architectures; this eventually showed prominent signs after introducing networks like VGG16 & VGG19, which are deeper versions of a standard convolutional architecture with over 100 million parameters [4]. later the convention of the regular convolutional neural network became obsolete as newer innovations to reduce the parameter explosion came into considerations. Some popular networks that introduced novel conventions to the architectures are *ResNet* which has residual blocks at the core. Residual units are essentially skipped connections, shown in Figure 3, also referred to as shortcuts. Residual blocks allow us to train much deeper networks without having to worry about weight decay [5]. The MobileNet architecture, a successor to the ResNet architecture, inherits the residual unit concept where the residual skip connections are placed between the bottleneck layers to enhance performance.

The second iteration of MobileNet, which was termed *MobileNet_V2* [6], has two essential layer structures called *Depth-wise convolution*, where a single kernel is applied to each channel and another layer structure called *Point-wise convolution*, where a 1x1 kernel is applied to decrease the dimensionality while preserving the contextual feature information. MobileNet's approach showcased a substantial reduction of the network's complexity and the overall network size, making it lightweight, consuming fewer processing resources and facilitating a better performance on mobile devices that are usually computationally limited. *DenseNet* [**7**] broke the deeper network design barrier while preserving the residual skip connection concept intact. In the DenseNet architecture, the individual layers are connected to each other as a feed-forward network along with the short cut connections from one layer to each subsequent layer. Figure 5 Illustrates 1 module of DenseNet architecture. GoogLeNet (*Inception_V1*) has stacked up convolutional layers making it a single *Inception module*, and the entire network consists of sequential connections of these modules. [8]. Later generations of the Inception V2, V3, and then Xception were significantly improved in areas where the first iteration lacked. Inception V3 had conventional inception blocks, but Xception featured radical kernel enhancements within the inception blocks [9], as shown in Figures [6, 7]

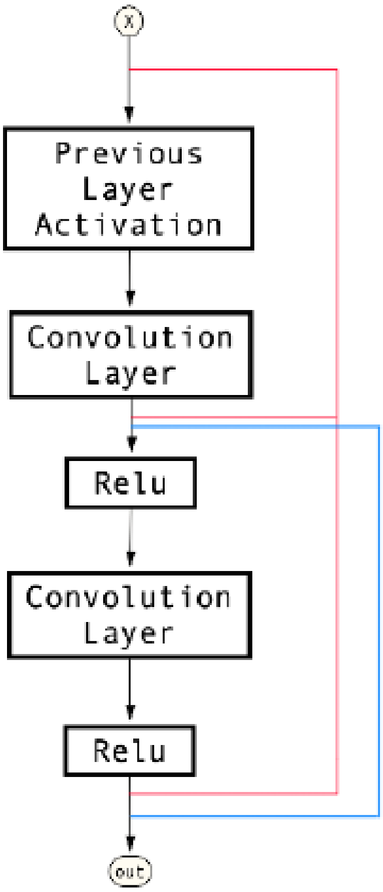

Fig 5: Simple DenseNet Architecture

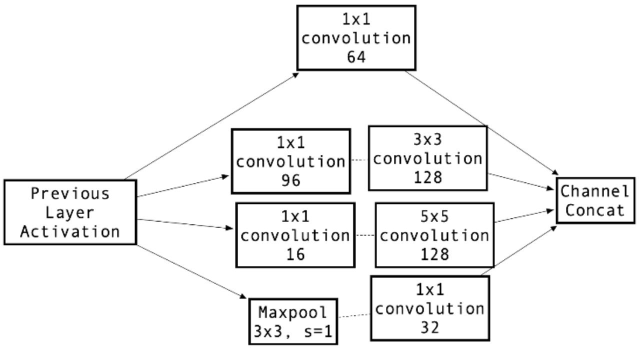

Fig.6. InceptionV3 Module

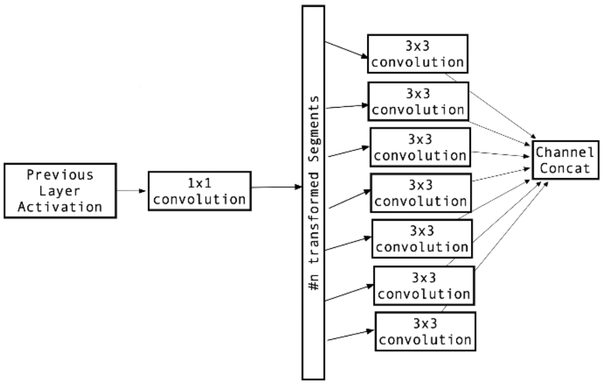

Fig. 7: Xception Module

F. The ImageNet Database

All the available networks are referred to as "pre-trained" because all these architectures are trained on the ImageNet database consisting of 14 million images of 22 thousand classes which are organized and indexed for easier access [11]. It aims to assist applications like Object [10] localization/detection, Image classification, Object Clustering.

G. The Transfer Learning Approach

Transfer Learning in practice can be approached in 3 conventional ways. All the methods include joining the network with a custom output module that predicts the classes that we have in our problem statement by using either *Softmax* or *Logistic Activation* for multi-class or binary classification tasks. The following transfer learning shows how a pre-trained architecture's knowledge can be transferred to another task completely irrelevant to its original domain. When we have very large samples of data, the common ways to train using a pre-trained network is to either "*Partially Freeze*" layers, to train some portion of the pre-trained network architecture and the custom end portion of the network, which classifies our task as shown in Figure 8. And the other way is to train the entire architecture along with an optimization technique and fine-tune it through the numerous epochs compiled to it. This way, we are essentially training it from scratch, making use of the "custom" nature of the network architecture conjoined with our own custom output layer.

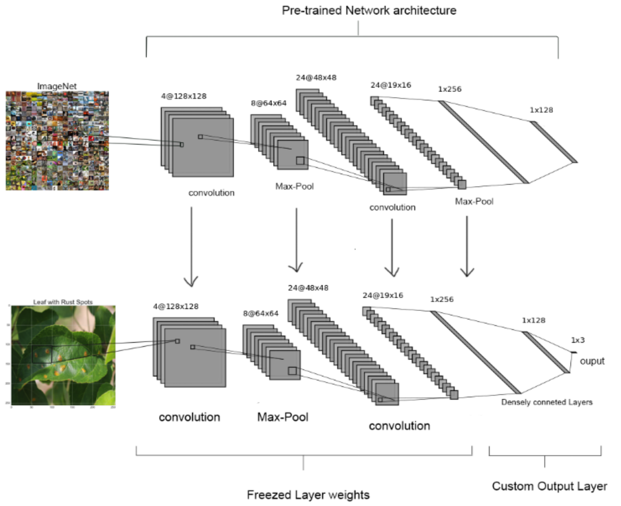

Fig. 8: A schematic illustration of the transfer learning approach

The third most common approach is to "Freeze" the network's weights from training between the epochs and train only the outer custom layer. This way the networks, knowledge from the previous domain remains intact now being applied to our problem statement. This assertion of knowledge transfer can be evidently shown if we look at the decoded probabilities for the custom architectures along with their pre-trained weights. Figure 9 describes the classification probabilities of different ImageNet classes yielded from 2 different networks for a sample Image (Healthy Class) from our dataset. From this generalization, it is evident from the probabilities that both *Bell Pepper* and *Leaf Beetle* have similar features as the input sample image. The most plausible reason being the commonality of the color green associated with the images of these high probabilities of decoded classes [12].

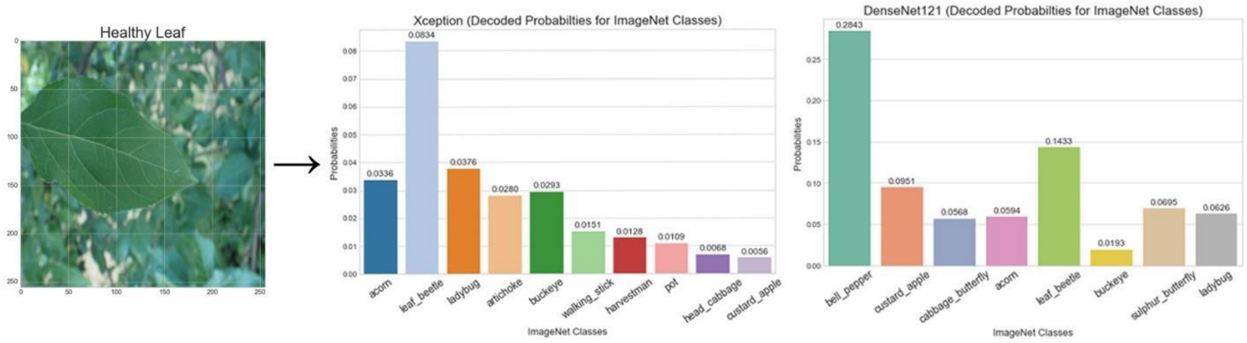

Fig.9: Decoded Probabilities of ImageNet weight classification for 2 different architectures

For the transfer learning part of this research, we used the technique of freezing the whole network's weights to leverage the ImageNet weights and their priority over the final classification. we used *model.trainable = False* [13] to disconnect the weights from being trained while iterating over the epochs. Python was used for the programming interface and TensorFlow-2.x with Keras as the Deep Learning framework of choice.

All the pre-trained network architectures used were connected with custom output layers to classify 3 classes as the data suggests. The custom output layers are comprised of two densely connected hidden layers with 256 and 64 neurons, respectively, that are coupled using Relu activations. The final layer consists of 3 units with *Softmax* activation to get classification probabilities for 3 classes. An active *Dropout* Layer with a 40% probability was inserted between the final output unit and the 64-unit hidden layer to control the overfitting problem. The pre-trained architectures and their parameter vector sizes used for this research are described in Table 1.

Table-I: Pre-trained Neural Network Architecture Details

Model	ImageNet Accuracy	Parameters	Layers
Xception[9]	0.945	22,910,480	126
VGG16[4]	0.901	138,357,544	23
VGG19[4]	0.900	143,667,240	26
InceptionV3[19]	0.937	23,851,784	159
MobileNetV2[6]	0.901	3,538.984	88
DenseNet121[7]	0.923	8,062,504	121

Programmatically, we used *tf.keras.applications* [13] library with python to access these pre-trained custom architectures from the internet along with their weights (ImageNet) and configured them.

H. Training

To train and record the performance indicators for all of the networks we selected to work with, we developed a single pipeline. Each model was compiled with *Adam optimizer* for the optimization of the updated weights with a learning rate of 0.01. *Categorical Crossentropy* for the loss function also referred to as Softmax loss, is extensively used for multi-class classifications. It's an aggregate of each class label's individual losses per sample. It is mathematically represented as the following.

$$Loss = -\sum_{i=1}^{\text{output}} yi \cdot \log \hat{y}i$$

- yi is the i^{th} value of the true vector of classes.
- ŷi is the i^{th} value of the predicted probability vector of classes.

Retrieval Number: 100.1/ijeat.F30800810621
DOI:10.35940/ijeat.F3080.0810621

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication

Early stopping is a predefined *Call-back* method used to monitor and terminate training if the network starts to overfit on the data while cross-checking with the accuracy of the validation set. The ImageDataGenerator in TensorFlow was used to read all of the needed image data from the directory and construct a generator object that could be directly fed to the *model.fit* function. The training set accuracy and loss, as well as the validation set accuracy and loss, are recorded throughout the train cycle.

IV. RESULTS AND DISCUSSION

In this section, we examine our findings to provide a highly efficient model that addresses the problem statement. Miss-classifications of are the commonly observed anomalies in almost every classification task, to quantify these miss-classifications we considered the metrics *Precision*, *Recall* and the *F1 Score*. Precision and recall values calculated by considering the classification terms - True Positives, True Negatives, False Positive, False Negative, True Positive and True Negatives represent the actual classifications, where False Positive and False Negatives represent the negated represented for the true class. Precision is defined as the fraction of correctly classified True Positive observations to the total positive predictions. Recall is fraction of accurately classified True Positives to the total number of True Classifications. F1 Score is the harmonic mean of both precision and recall. Records of accuracy, recall, and F1 scores for the networks trained are provided in Table 2. Total Correct on Test/105 represent the total number of samples -- out of 105 -- which are correctly classified on the test set.

Table -II: Metric evaluation of custom architectures

Model	Test Accuracy	F1 Score	Precision	Recall	Total Correct on Test-Set/105
DenseNet121	0.990	0.961	0.961	0.962	101
VGG16	0.933	0.904	0.904	0.908	95
InceptionV3	0.971	0.932	0.933	0.935	98
MobileNetV2	0.971	0.962	0.961	0.965	101
Xception	0.942	0.942	0.942	0.943	99
VGG19	0.857	0.839	0.838	0.867	88
ConvNet	0.760	0.656	0.657	0.660	69

In this research, the obvious finding is that with the ImageNet weights as preset, DenseNet121, MobileNetV2 performs considerably better. It is undeniable that the architectural advantage has a significant impact on the performance. And it's apparent that given the smaller batch amount of data we have, attaining better results from a ConvNet trained from scratch with this limited data constraint is challenging. Finally, to get a better understanding of the two best performing models, both DenseNet121 and MobileNetV2, We plotted confusion matrices to find the apparent miss-classifications which are insightful to acknowledge for future context when using these networks with similar data. Confusion Matrix is a 2-Dimensional Matrix with Rows and Columns presenting each of the classes. The left-to-right diagonal in this matrix organization, defines the actual number of accurate predictions for the class represented across the row. There are instances where Healthy class leaf is classified as either Powdery class or Rusted class, these generalizations are easily noticed when we use confusion matrix.

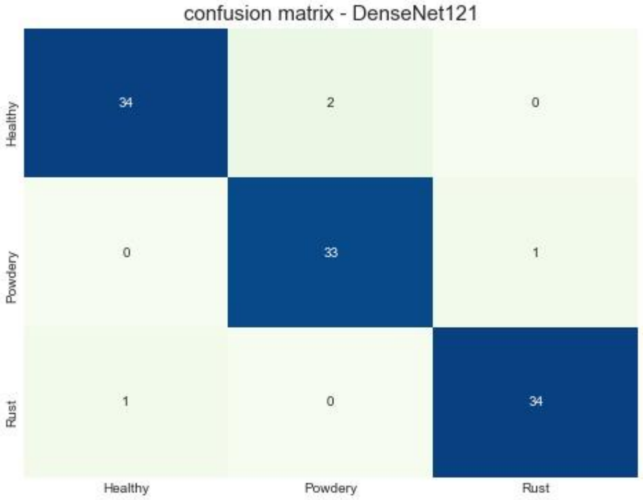

Fig. 10: Confusion Matrix (DenseNet121)

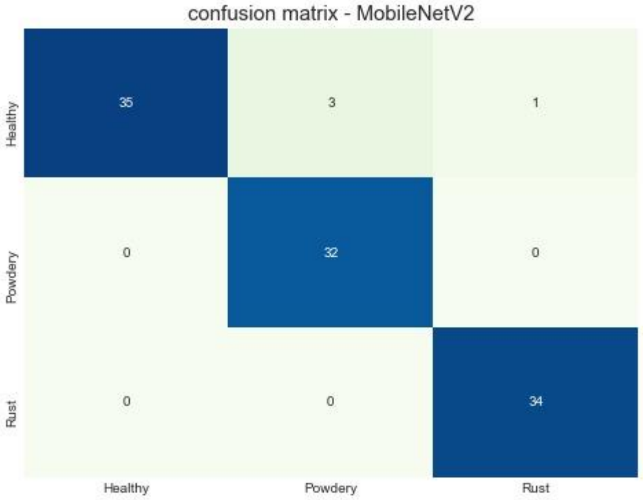

Fig. 11: Confusion Matrix (MobileNetV2)

V. CONCLUSION

With the latest advances in deep learning and its integral contribution to computer vision applications, crop surveillance for disease identification is one such example. Deep learning can have a notable edge over the manual identification of these diseases. For our research, where we had 1532 samples from an open-sourced data set of images with two classes being the most common type of diseases observed in crops, we utilized Transfer Learning with networks of custom architecture trained for ImageNet classes. Using the ImageNet weight parameters as the precedence over this task, we trained InceptionV3, MobileNetV2, DenseNet121, VGG16, VGG19, and Xception networks. From the comparative analysis of the results obtained, it is apparent that MobileNetV2 and DenseNet121 had attained maximum accuracy of 97.1% and 99.0% on the test set to classify these diseases compared to the other networks. The F1 scores respective models recorded are 96.2% and 96.1%.

The confusion matrix further solidifies the claim showing barely any miss-classifications on the test set for the two networks. In summary, our work can be extended when using more precise data related to a particular species of plants, with transfer learning using these networks as the foundational approach to another problem has better chances to fetch optimal results.

REFERENCES

1. I. Goodfellow, Y. Bengio, and A. Courville, Deep learning. Cambridge, Massachusetts: The Mit Press, 2016.
2. LeNail, (2019). NN-SVG: Publication-Ready Neural Network Architecture Schematics. Journal of Open Source Software, 4(33), 747, https://doi.org/10.21105/joss.00747.
3. Andrew Y Ng, "Lecture — Orthogonalization," COURSERA: https://www.coursera.org/lecture/machine-learning-projects/orthogonalization-FRvQe, (accessed Aug 16, 2021).
4. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
5. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778)
6. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520).
7. Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).
8. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).
9. Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1251-1258).
10. Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database." In 2009 IEEE conference on computer vision and pattern recognition, pp. 248-255. Ieee, 2009.
11. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems 25 (2012): 1097-1105.
12. Dipanjan Sarkar, Raghav Bali, and Tamoghna Ghosh, Hands-on transfer learning with Python implement advanced deep learning and neural network models using TensorFlow and Keras. Birmingham ; Mumbai Packt Publishing August, 2018.
13. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Zheng, X. (2016). Tensorflow: A system for large-scale machine learning. In 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16) (pp. 265-283).
14. K. Gurney, An Introduction to Neural Networks, Taylor & Francis, Inc., Bristol, PA, USA, 1997.
15. L. Xu, J. S. Ren, C. Liu, and J. Jia. Deep convolutional neural network for image deconvolution. In Advances in neural information processing systems, pages 1790–1798, 2014.
16. J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In Advances in neural information processing systems, pages 3320–3328, 2014.
17. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet Large Scale Visual Recognition Challenge. Int. J. Comput. Vis. (IJCV) 2015, 115, 211–252.
18. Yang, Q., Zhang, Y., Dai, W., & Pan, S. (2020). Transfer Learning. Cambridge: Cambridge University Press. doi:10.1017/9781139061773
19. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826)
20. LeCun, Y., Haffner, P., Bottou, L., & Bengio, Y. (1999). Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision* (pp. 319-345). Springer, Berlin, Heidelberg.
21. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 1097-1105.
22. Ghosal, S., & Sarkar, K. (2020, February). Rice Leaf Diseases Classification Using CNN With Transfer Learning. In *2020 IEEE Calcutta Conference (CALCON)* (pp. 230-236). IEEE.
23. Thangaraj, R., Anandamurugan, S., & Kaliappan, V. K. (2021). Automated tomato leaf disease classification using transfer learning-based deep convolution neural network. *Journal of Plant Diseases and Protection*, *128*(1), 73-86.
24. Atila, Ü., Uçar, M., Akyol, K., & Uçar, E. (2021). Plant leaf disease classification using EfficientNet deep learning model. *Ecological Informatics*, *61*, 101182.
25. Chen, J., Chen, J., Zhang, D., Sun, Y., & Nanehkaran, Y. A. (2020). Using deep transfer learning for image-based plant disease identification. *Computers and Electronics in Agriculture*, *173*, 105393.

AUTHORS PROFILE

N Rohan Sai, is currently a final year Computer Science and Engineering undergraduate at Gandhi Institute of Technology and Management (Deemed to be University), Visakhapatnam. His research interests include Deep Learning for Computational Astronomy, Computer Vision, Healthcare, and Natural Language Processing.

T Sudarshan Rao, is a final year Computer Science and Engineering student from Gandhi Institute of Technology and Management (Deemed to be University), Hyderabad. His research interests are Deep Learning, Computer Vision, Reinforcement Learning, Computational Linguistics, and Statistical Machine Learning.

G.L. Aruna Kumari, is presently working as Assistant Professor in the department of Computer Science and Engineering, GIT, Gandhi Institute of Technology and Management (Deemed to be University), Visakhapatnam. She presented research papers in national and international conferences and journals of good repute. She guided several students for Project work in department of Computer Science Engineering. Her current research interests include data mining and machine learning.