Group GDE/GDF-18
Plant Disease Detection and Classification using Transfer Learning Inception v3

Mounik Patel
1144127
MS in Computer Science
Lakehead University
mpatel78@lakeheadu.ca

Samip Chetanbhai Karbhari
1146116
MS in Computer Science
Lakehead University
skarbhar@lakeheadu.ca

Vairag Nileshbhai Parikh
1153939
MS in Computer Science
Lakehead University
vparikh2@lakeheadu.ca

Abstract

The identification of plant diseases is critical to preventing yield and quantity losses in agricultural products. Plant disease research entails looking at visually observable patterns on plants. Plant health monitoring and disease detection are crucial for sustainable agriculture. Manually monitoring plant diseases is quite a challenge. It necessitates a significant amount of work, knowledge of plant diseases, and extensive processing time. As a result, image processing is utilized to detect plant diseases. The approach to detecting plant diseases using leaf images is discussed in this research. The proposed method detects diseases in apple plant leaf images using Transfer Learning Inception v3. Furthermore, the model has been evaluated in terms of loss and accuracy. The results demonstrate that the Inception v3 model with Transfer Learning approach is very effective in detecting and classifying apple plant diseases.

1. Introduction

Plants are susceptible to disease for a variety of reasons, including fertilizers, cultural practices, environmental conditions, etc. These diseases have an adversely effect on agricultural yield and, as a result, on the harvest. Any approach or method for overcoming this challenge and receiving a warning before the plants become infected would help farmers cultivate crops or plants more efficiently to gain both quality and quantity. Henceforth, early disease detection in plants is crucial in agriculture.

The workflow for plant disease detection and classification consists of the following steps:

- Image preprocessing is a technique to remove the noise from the image or removing an object to simplify processing time. For example, image scaling to convert original image to a thumbnail to decrease processing time.

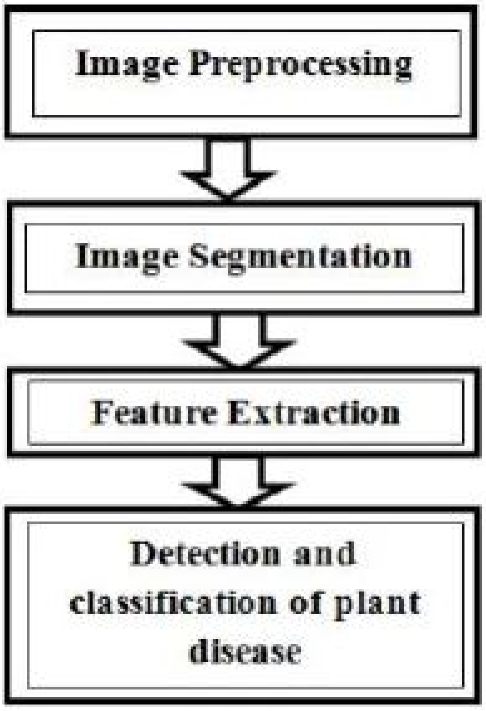


Figure 1. Basic Steps for Plant Disease Detection and Classification

- Image segmentation is to divide an image into multiple regions to easily classify the pixels in each region.
- Feature extraction is done for identification of an object by separation of features such as color, edges, morphology, texture, etc.
- Detection and classification is performed by various learning algorithms of Deep learning and ML models such as ANN and Backpropagation, SVMs, Random Forest, K-nearest neighbor, etc.

Agricultural productivity is an essential factor in most of the developing countries and so, plant disease detection plays the most important role for high yield. Appropriate research is necessary in this area for improving product quality, quantity as well as productivity. Apple Scab, Black spot, Bacterial canker, Blight, Rust and Black Knot are one of the most common plant diseases that need to be detected and prevented. By detecting these disease, there can be a study of conditions responsible behind them and crucial steps can be taken.

2. Related Work

The authors in [10] introduced an image segmentation model that uses genetic algorithm. The research paper also surveys various techniques for classifying diseases that can be utilized for the detection of plant leaf diseases. Image acquisition and pre-processing is done before it is segmented to classify the leaf diseases. Image segmentation is an important technique used to detect and classify leaf diseases of plants. Once an image is segmented, genetic algorithm is used with k-Means clustering to cluster each pixel accordingly. Genetic algorithms are evolutionary algorithms that provide solutions to problems of optimization. In addition, crucial and unique elements are extracted using a methodology of color co-occurrence that takes into consideration both the texture and color of the image. Once the features are extracted and compared, they are stored in the feature dataset. Disease classification is performed using Minimum Distance Criterion and Support Vector Machine classifiers. Eventually, the model is evaluated with the help of classification gain. Results show that SVM with proposed algorithm improves detection accuracy in comparison to other approaches. Optimum results were achieved with very less computational efforts and plant diseases were identified at an early stage, which makes the proposed algorithm more efficient in detecting and classifying the plant leaf diseases.

Different image processing methods used for plant disease detection as well as classification is discussed in [8]. It provides emphasis on most basic image processing techniques following the sequence: image acquisition, image pre-processing, image segmentation, feature extraction and classification. For image acquisition purpose, plant leaf images are captured in RGB (Red, Green, Blue) form and a device-independent color space transformation is implemented. Image pre-processing steps, firstly include cropping of leaf image part from the original plant image. Secondly, Image smoothing is performed using smoothing filter and image enhancement is done by increasing the contrast. At last, the image is converted to grayscale using color conversion. Furthermore, image is partitioned on the basis of similarity for segmentation. This included three methods such as boundary and spot detection algorithm using HIS model, k-means clustering and Otsu threshold algorithm. The purpose of Otsu threshold algorithm is to convert grayscale level image into binary images having values 0 and 1, on the basis of estimated threshold. Feature extraction was performed using Color co-occurrence method and Color Extraction by H B components. Finally, the research proposed disease classification techniques such as – Artificial Neural Network (ANN), which includes modified SOM (Self Organizing Map) and multiclass SVMs in addition to Back-Propagation Neural Network algorithm (BPNN).

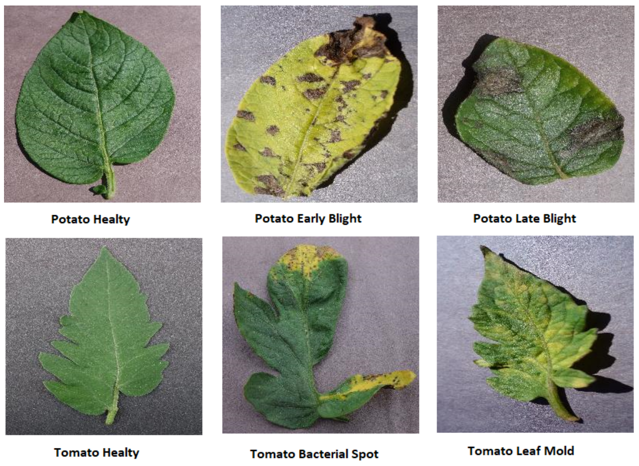


Figure 2. Types of plant diseases identified from leaves

Ferentinos in [5] developed specialized deep learning models, based on specific convolutional neural network architectures, to perform plant disease detection using simple leaves images of healthy and diseased plants. The basic deep learning tool used in this work is Convolutional Neural Networks (CNN). CNNs are an evolution of traditional artificial neural networks, focused mainly on applications with repeating patterns in different areas of the modeling space, especially image recognition. The five basic CNN architectures that were trained and tested in this work concerning the identification of plant diseases from images of their leaves, were the following: AlexNet, AlexNetOWTBn, GoogLeNet, Overfeat, and VGG. The CNN models were trained and tested on an open database containing images of leaves of healthy and infected plants. The database includes images of leaf in both laboratory conditions and in real conditions. The database was initially divided into training set and testing set by 80/20 ratio using 3 approaches. Pre-processing of image was done for only the size of the image, and any additional pre-processing or segmentation was not done because CNN models are highly capable of identifying important and non-important features of an image. Comparison of the CNN models were based on their performance on the testing set, as all the models were 100 percent accurate during training. The VGG model represented the final model for plant disease detection because

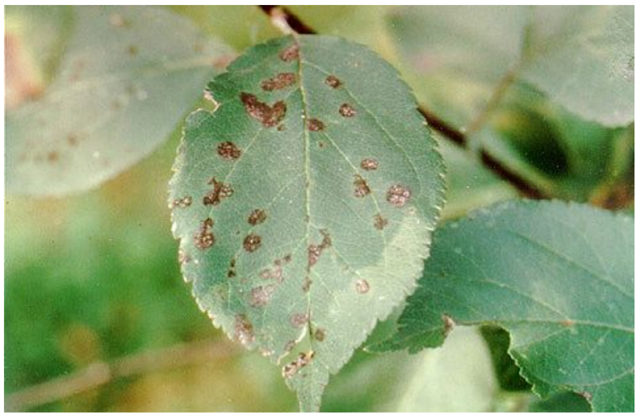


Figure 3. Apple Scab

of the highest successful classification percentage of 99.53 percent during testing. Based on its great performance, it is apparent that convolutional neural networks are highly suitable for the automated detection of plant diseases through the analysis of simple leaves images.

The researchers in [7] proposes a real-time detection approach that is based on improved convolutional neural networks (CNNs) for apple leaf diseases. The deep-learning based approach can automatically extract the discriminative features of the diseased apple images and detect the five common types of apple leaf diseases with high accuracy in real time. In this paper, the apple lead disease dataset (ALDD), which is composed of laboratory images and complex images under real-field conditions, is generated via data augmentation and image annotation technologies. Furthermore, a new apple leaf dataset detection model, namely, INAR-SSD (single-shot multibox detector with Inception module and Rainbow concatenation) that used deep-CNNs is designed by introducing the GoogLeNet Inception structure and integrating the Rainbow concatenation. The experimental results show that the INAR-SSD model realizes a detection performance of 78.80 percent mAP (mean average precision) on ALDD, with a high detection speed of 23.13 FPS. This result demonstrates that the novel INAR-SSD model provides a high-performance solution for the early diagnosis of apple leaf diseases that can perform real-time detection of these diseases with higher accuracy and faster detection speed than the previous methods.

3. Methodology

This section introduces the Inception v3 model [11], which is used to explore the Inception architecture [12], as well as the Transfer Learning approach, which is a reinforcement to the proposed Inception v3 model.

3.1. Inception v3

The CNN is a neural network that was specifically created to solve image recognition problems. It simulates the multi-layered process of human image recognition. CNNs have five layers: input, convolutional, pooling, fully connected, and output [4]. We need to increase the depth or width of the model to boost the feature extraction capabilities of CNNs. This causes two problems: overfitting as the number of parameters increases, and a significant rise in computational resource usage [6].

Google's GoogLeNet network is a CNN that was proposed in 2014. It uses the Inception network structure, which decreases the number of network parameters while increasing network depth. As a result, it is commonly employed in image classification tasks. Because the Inception network structure is at the core of the GoogLeNet network, it is also known as the Inception network [11]. GoogLeNet occurs in several variants, including Inception v1 (2014), Inception v2 (2015), Inception v3 (2015), Inception v4 (2016), and Inception-ResNet (2016) [12].

The Inception architecture combines parallel layers with varying convolutional kernel sizes and concatenates their outputs at the end to improve network flexibility at different scales and reduce overfitting. Fig. 4 depicts the Inception network structure. Inception v3 is primarily a Keras-developed network structure that is pre-trained in Image Net. The Inception v3 network structure used in this paper is shown in Fig. 5 [4]. The Inception v3 model is made up of symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, concats, dropouts, and fully connected layers. The model makes considerable use of Batchnorm, which is applied to activation inputs. To compute losses, Softmax is utilized [2].

Compared to the previous versions (Inception v1 and v2), the Inception v3 network structure uses a convolution kernel splitting method to divide large volume integrals into small convolutions. Through the splitting method, the number of parameters can be reduced; hence, the network training speed can be accelerated while the spatial features can be extracted more effectively [4].

In comparison to VGGNet, Inception Networks (GoogLeNet/Inception v1) have proved to be more computationally efficient, both in terms of the number of parameters generated by the network and the economical cost incurred (memory and other resources). If any changes are to be made to an Inception Network, care needs to be taken to make sure that the computational advantages are not lost. Thus, the adaptation of an Inception network for different use cases turns out to be a problem due to the uncertainty of the new network's efficiency. In an Inception v3 model, several techniques for optimizing the network have been put suggested to loosen the constraints for easier model adaptation. The techniques include factorized

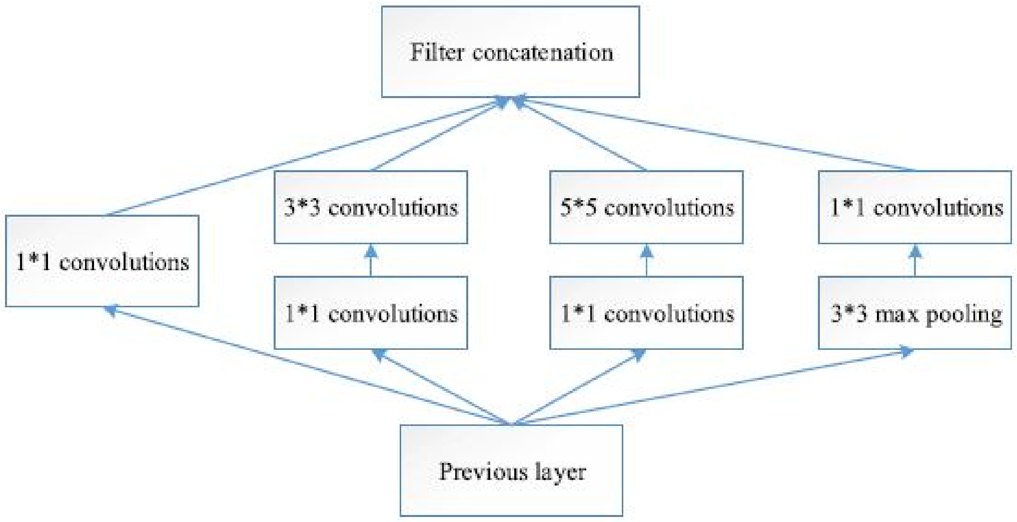


Figure 4. Inception network structure [4]

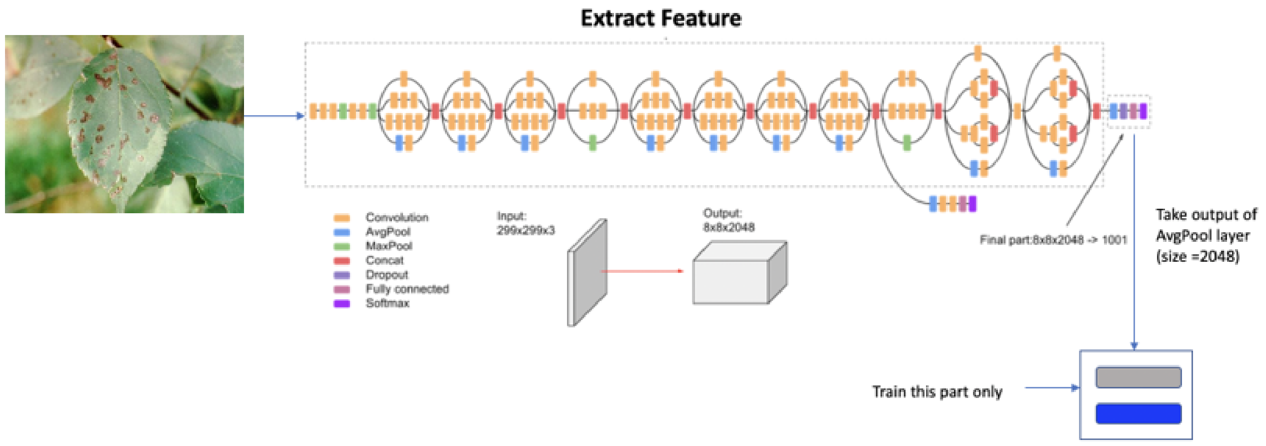


Figure 5. Transfer Learning Inception V3

convolutions, regularization, dimension reduction, and parallelized computations.

3.2. Transfer Learning

Transfer learning is a machine learning approach that can leverage previously acquired knowledge from one environment to tackle a new problem that is different but has the same relationship as the old problem. In comparison to traditional neural networks, it simply requires a small amount of data to train the model and attain high accuracy in a short amount of time [14].

As the depth of the network increases, the performance of the neural network improves, but at the expense of time and computational resources. As a result, deep learning based on transfer learning has evolved as a method for lowering training costs [13]. By learning new data distributions and fine-tuning parameters across all layers of the model, a pre-trained model can be transferred to execute a similar task [9]. Transfer learning can be used to boost and optimize the learning efficiency of the new model by sharing the parameters of the trained model with the new model rather than starting from scratch [13].

Models can be downloaded and used as feature extraction tools as well. The result of a layer starting from the first to the model's output layer is used as input to a new classifier model in this case.

The convolutional layers closer to the model's input layer quickly learn low-level features like lines, layers in the middle of the layer learn complex abstract features that combine the lower-level features extracted from the input, and layers closer to the output layer interpret the extracted features in the context of a classification task.

With this knowledge, we may choose a level of detail for the feature extraction from an already existing pre-trained model. If the new goal is substantially distinct from categorizing objects in images (for example, different from ImageNet), the output of the pre-trained model after a few layers might be appropriate. If a new task is very similar to classifying objects in images, it may be possible to use the output from layers much deeper in the model, or even the output of the fully linked layer prior to the output layer.

The pre-trained model can be used as a standalone feature extraction program, in which case the model or a component of the model can pre-process each input image to produce an output (e.g. a vector of numbers) that can subsequently be utilised as input for training a new model.

The pre-trained model, or a selected section of the model, can also be immediately merged into a new neural network model. The pre-trained model's weights can be frozen in this case so that they are not modified as the new model is trained. Alternatively, the weights could be changed throughout the new model's training, possibly at a slower rate, allowing the pre-trained model to operate as a weight initializer while training the new model.

Transfer Learning is where we use a pre-trained model, is a super-effective strategy when we have a relatively limited dataset. We would be able to transfer weights gained during hundreds of hours of training on numerous high-powered GPUs because this model was trained on such a big dataset. Many of these models, such as VGG-19 and Inception-v3, are open-source. They were trained on millions of photos using incredibly high computational power, which is difficult to achieve from scratch.

In our project, we have applied the Inception-v3 model. Transfer Learning has extremely increased in popularity since it has very less training time and takes far less data to improve the performance. Transfer Learning also has great applications in Computer Vision. Because not everyone can afford weeks of training and access to millions of images to construct a high-performing CNN model for their specific use case, Transfer Learning comes out as a option. We can gain the ability and freedom to retrain some of the lower layers (in accordance with the goal of classes) with a decent number of resources.

4. Experiment Details

This section describes the dataset utilized in this research and describes how the Transfer Learning-based Inception v3 model for plant disease detection and classification was implemented.

Simple and powerful way to create better deep learning model is to make it deeper (i.e., more complex). However, this will lead to overfitting, it can be easily visualized by smaller dataset. Also, increasing parameters will require more computational resources. Hence, as a solution we can create sparsely connected architecture rather than densely connected.

4.1. Dataset Description

This project makes use of the New Plant Diseases Dataset, which can be obtained on the Kaggle website [3]. To improve the performance and generalization of training data, this dataset was recreated using offline augmentation from the original dataset. This dataset contains RGB images of healthy and diseased crop leaves that have been classified into different disease classes. For training purposes, the complete dataset has been divided into an appropriate ratio of training and validation sets. For prediction purposes, a separate test set of a few images (previously unseen by the model) has been created. In this study, we used images of only one crop/plant for plant disease classification because the dataset is too large, and it exceeds the available computational resources.

4.2. Implementation of Transfer Learning Inception v3

Call model.summary() used to print important summary of the model, which includes the following: As indicated in [1]

- Name and type of all layers in the model.
- Output shape for each layer.
- Number of weight parameters of each layer.
- If the model has general topology (discussed below), the inputs each layer receives
- The total number of trainable and non-trainable parameters of the model

Keras ImageDataGenerator is a type of data augmentation technique using which we can perform variety of image data transformations on the training image data. This helps in improving model prediction accuracy as well as save memory overheard. Some of the transformations which can be applied are as follows:

- Horizontal flip: horizontal flip and for flipping along the the horizontal axis
- Zoom range: zoom will be done in the range [1-zoom range, 1+zoom range]

- Value smaller than 1 will zoom in — value greater than 1 will zoom out
- shear range: Shear angle in counter-clockwise direction in degrees
- Rescale: multiply the data by the value provided after applying all other transformations

Model fitting is a technique to analyze how accurate Machine learning model is working with respect to test and training data. A well-fitted model is gives more accurate results. Some parameters for model fitting are:

- Epochs: Number of times we want to train our model
- Steps per epoch: total number of steps taken from the generator as soon as one epoch is finished and next epoch has started
- Validation steps: total number of steps taken from the generator before it is stopped at every epoch

Python globbing is used in the project to output all images with similar file type. In this way, disease analysis can be performed on all the apple leaf images at a time.

5. Evaluation Metrics and Results

5.1. Evaluation Metrics

The evaluation of the model is done on the basis of accuracy and loss calculated during the training and validation of the model.

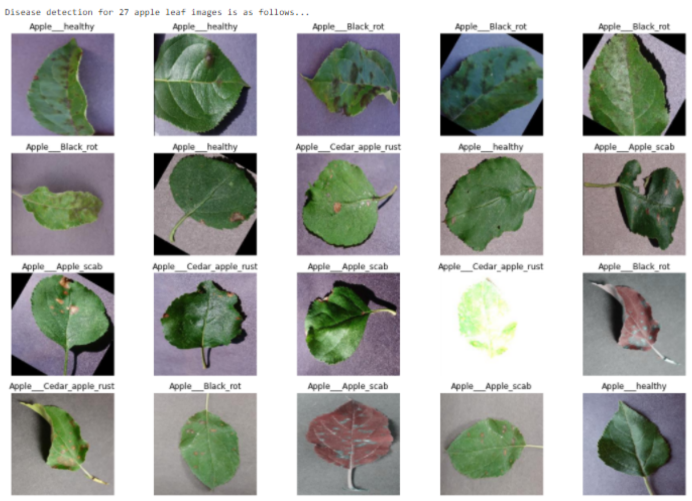
A loss function is used to improve the performance of a machine learning algorithm. The loss is calculated during training and validation and is interpreted depending on how well the model performs in these two sets. It is the total of all errors committed in training or validation sets for each example. The loss value indicates how poorly or well a model performs after each optimization iteration.

An accuracy metric is used to quantify the algorithm's performance in a meaningful way. The accuracy of a model is often determined after the model parameters and is expressed as a percentage. It is a measure of how close your model's prediction is to the actual data.

5.2. Results

The model was evaluated using different epoch values. Number of epochs indicate total number of passes or iterations performed on the training dataset. The algorithms performs better and increases in accuracy with more number of epochs. For this project, we iterated the model with dataset for 5, 10 and 15 epochs respectively.

Figures 7, 8 and 9 show the training and validation losses for each of the epoch iterations. Figures 10, 11, and 12 show the training and validation accuracy for each



```
Disease detection for 27 apple leaf images is as follows...
```

Figure 6. Disease detection for apple leaf images

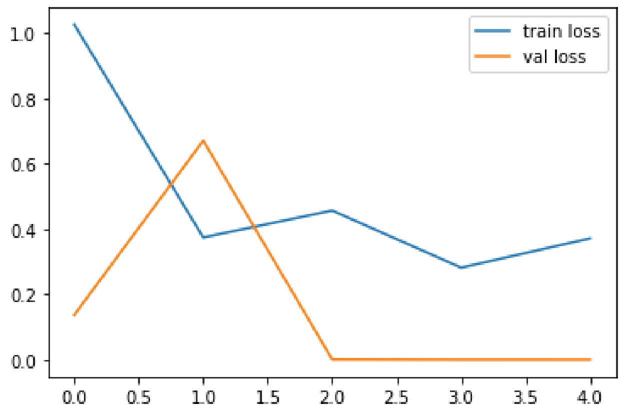


Figure 7. Training and Validation Loss for 5 epochs

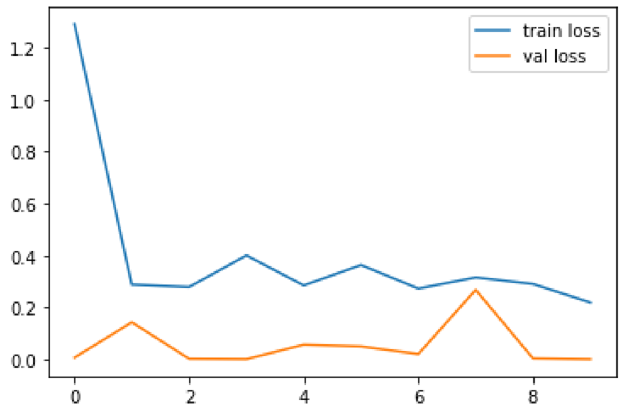


Figure 8. Training and Validation Loss for 10 epochs

of the epoch iterations. The output is shown as a grid of images with each image having a label predicted by the model,

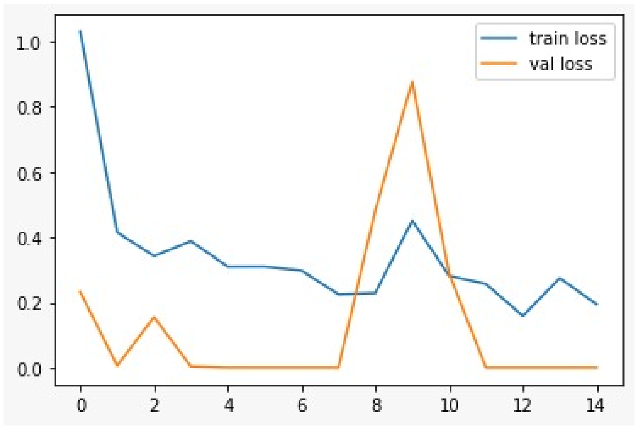


Figure 9. Training and Validation Loss for 15 epochs

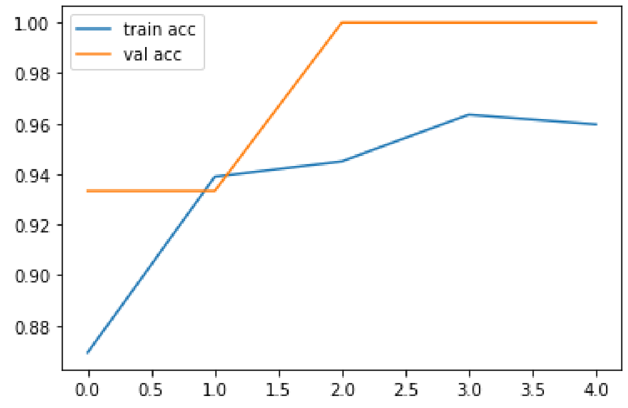


Figure 10. Training and Validation Accuracy for 5 epochs

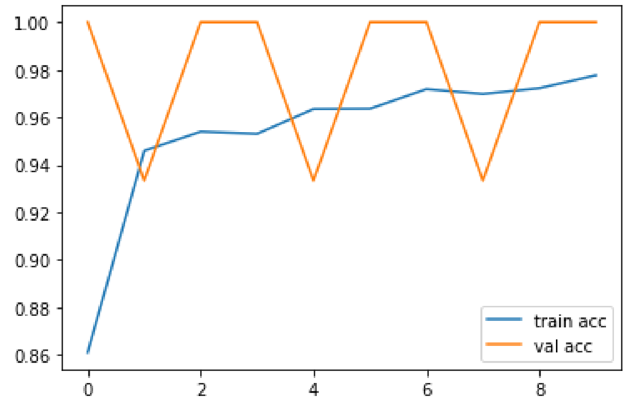


Figure 11. Training and Validation Accuracy for 10 epochs

which is shown in Fig. 6.

The results demonstrate that as the number of epochs increased, the training and validation loss decreased and the training and validation accuracy increased. This shows that

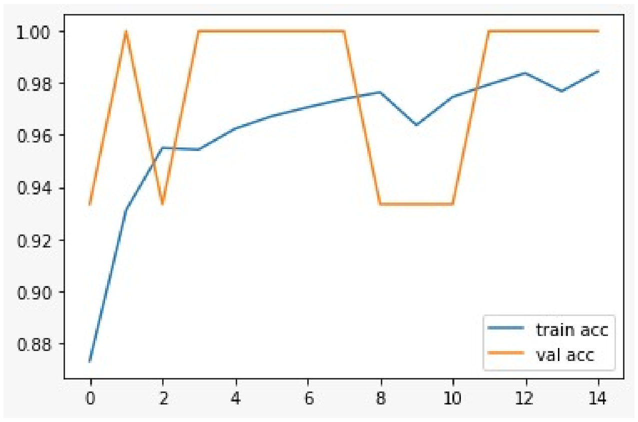


Figure 12. Training and Validation Accuracy for 15 epochs

as the number of epochs increases, the model will be thoroughly trained and validated.

The accuracy was low, found to be a little less than (93 percent) but increased with each epoch iteration. The final accuracy after the last epoch was found to be 98 percent. However, the validation accuracy remained 100 percent throughout the iterations.

6. Limitations

Despite the success of the Transfer Learning Inception v3 implementation in detecting and classifying plant diseases, this research project had a few limitations. Due to restricted processing power, we had to confine our dataset to a certain extent in order to achieve satisfactory accuracy and efficiency. In this research project, we are also deploying our classifier for detecting and classifying diseases for only one plant. As the number of epochs increases, computational resources are used up quickly and model implementation takes a very long time. We also found it logistically challenging to check and optimize each layer of the Inception v3 model due to its use of several deep layers.

7. Conclusions and Future Work

To conlude, we have implemented Plant disease detection using Transfer Learning based on Inception v3 architecture. We have used ConfigProto and InteractiveSession for easy and quick runtime environment. Tensorflow.keras is used for high level deep learning predictions and can perform complex tasks with simpler executions. At last, the leaf images are analysed using globbing techniques for all similar files and disease detection is performed successfully.

For future works we can perform the following tasks:

- We will try to create a UI so that it becomes easy for users to access our project.

- We will also try to implement functionality that will provide solutions to prevent plant diseases, such as fertilizer recommendations.
- We will try to detect plant diseases in images captured in real-time field conditions.
- We can also run our project on cloud platform for increasing computational speeds

References

[1] Models and layers : Tensorflow.js.

[2] Running Inception on Cloud TPU — Google Cloud.

[3] Samir Bhattarai. New plant diseases dataset, Nov 2018.

[4] N. Dong, L. Zhao, C.H. Wu, and J.F. Chang. Inception v3 based cervical cell classification combined with artificially extracted features. *Applied Soft Computing*, 93:106311, 2020.

[5] Konstantinos P. Ferentinos. Deep learning models for plant disease detection and diagnosis. *Computers and Electronics in Agriculture*, 145:311–318, 2018.

[6] Faizan ShaikhFaizan is a Data Science enthusiast and a Deep learning rookie. A recent Comp. Sc. undergrad. Inception network: Implementation of googlenet in keras, May 2020.

[7] Peng Jiang, Yuehan Chen, Bin Liu, Dongjian He, and Chunquan Liang. Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks. *IEEE Access*, 7:59069–59080, 2019.

[8] Sachin D. Khirade and A.B. Patil. Plant disease detection using image processing. In *2015 International Conference on Computing Communication Control and Automation*, pages 768–771, 2015.

[9] Chunmian Lin, Lin Li, Wenting Luo, Kelvin C. Wang, and Jiangang Guo. Transfer learning based traffic sign recognition using inception-v3 model. *Periodica Polytechnica Transportation Engineering*, 47(3):242–250, 2018.

[10] Vijai Singh and A.K. Misra. Detection of plant leaf diseases using image segmentation and soft computing techniques. *Information Processing in Agriculture*, 4(1):41–49, 2017.

[11] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[12] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[13] Cheng Wang, Delei Chen, Lin Hao, Xuebo Liu, Yu Zeng, Jianwei Chen, and Guokai Zhang. Pulmonary image classification based on inception-v3 transfer learning model. *IEEE Access*, 7:146533–146541, 2019.

[14] Xiaoling Xia, Cui Xu, and Bing Nan. Inception-v3 for flower classification. *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*, 2017.