International Journal for Modern Trends in Science and Technology, 8(08): 48-58, 2022

ISSN: 2455-3778 online
DOI: https://doi.org/10.46501/IJMTST0808008
Available online at: http://www.ijmtst.com/vol8issue08.html

Disease Detection in Plant Leaves Using Deep Learning

Krishna Kabra

Computer Science, SNBP International School

To Cite this Article
Krishna Kabra. Disease Detection in Plant Leaves Using Deep Learning. International Journal for Modern Trends in Science and Technology 2022, 8(08), pp. 48-58. https://doi.org/10.46501/IJMTST0808008

Article Info
Received: 25 June 2022; Accepted: 30 July 2022; Published: 04 August 2022.

ABSTRACT

Plant disease detection is a very critical and urgent issue that needs to be solved in order to achieve food security. Each year plant diseases and pests cause a global yield loss of more than 50% of crop production. The use of traditional manual methods for this task is inefficient as it requires expertise, labour and time. This calls for better methods to automatically detect and classify diseases in plant leaves. This project proposes a model which uses the Efficient Net V2 family of neural networks to perform plant disease classification.

The study makes use of the PlantVillage dataset containing more than 50,000 plant leaf images and also tries to use Generative Adversarial Networks (GAN) and Data Augmentation to enrich the existing dataset by increasing its size. Disease detection is performed in nine different plant species and 33 total categories to obtain an accuracy of up to 96.4%, with an increase in accuracy over random guessing ranging from 160–620%. The study also makes use of techniques like transfer learning and fine tuning of hyperparameters with an aim to increase overall classification accuracy.

KEYWORDS:Plant disease detection, Data augmentation, Efficient Net V2, PlantVillage,

1. INTRODUCTION

Agriculture is the primary source of livelihood in India[1]. It is the primary source of income for approximately 58% of India's population, mostly in the rural sector. Even though agriculture's contribution to India's GDP has halved in the last 30 years to about 15%, India is still among the world's top producers of pulses, rice, wheat and vegetables[2]. With an ever-increasing demand for improvement in both the quantity and quality of food due to a rapidly increasing population, the enhancement of technology in the sector of agriculture is being given a major push.

One of the major factors which causes a huge loss to farmers is plant diseases. Global yield losses due to crop diseases and pests may comprise more than 50% of crop production[3]. This calls for a solution to eliminate or minimise these losses as much as possible. One possible method for this is the early detection of such diseases in plants so that remedial action can be taken quickly and losses can be prevented. The current practice for detecting these diseases is simply by manual observation through the naked eye [4]. In addition to having a high scope for errors, this process is highly demanding in terms of time and labour.Thus, the use of machine learning to automate the process of detection of disease in plant leaves is fast becoming a very lucrative option. Table 1.1 shows all the plants and their diseases that the project aims to detect.

Plant	Diseases Detected
Apple	Apple Scab Black Rot

	Cedar Apple Rust
Cherry	Powdery Mildew
Corn	Cercospora (Gray) Leaf Spot Common Rust Northern Leaf Blight
Grape	Black Rot Esca (Black Measles) Isariopsis Leaf Spot (Leaf Blight)
Peach	Bacterial Spot
Bell pepper	Bacterial Spot
Potato	Early Blight Late Blight
Strawberry	Leaf Scorch
Tomato	Bacterial Spot Early Blight Late Blight Leaf Mold Septoria leaf spot Spider mites (Two spotted spider mite) Target Spot Yellow Leaf Curl Virus Mosaic Virus

Table 1.1 Diseases which the study aims to detect

The results obtained show that this model has potential. A highest accuracy of 96.4% was obtained in the Peach plant while in the Tomato plant, the model gave an accuracy of 620% better than random guessing. Moreover, this model can be used for other species of plants as well if a dataset of that plant can be found.

STRUCTURE OF PAPER

The paper is organized as follows: In Section 1, the introduction of the paper is provided along with the structure, important terms, objectives and overall description. In Section 2 we discuss the related work. In Section 3 we have the complete background information about the dataset and the image processing tools.Section 4 tells us about the methodology and the process description. Section 5 discusses and analyses the results obtained. Section 6 tells us about the future scope. Section 7 concludes the paper with acknowledgement and references.

2. RELATED WORK

The current work in the field of disease detection in plant leaves is mainly distributed into two approaches. The first uses machine learning to train a model which can then extract features from the images and use them to identify whether they are diseased. The other uses computer vision techniques to manually extract the features

Barbedo& J.G.A[5] uses two datasets having a total of 138 images of various plant species. One problem is that the background has to be as close to white or black as possible. The images were converted from RGB to L*a*b (Lightness and two colour-opponent dimensions) format. Then the detected regions were extracted. An accuracy of 96% was eventually obtained by this method. This accuracy is of obtaining the diseased regions in a leaf, not identifying which disease. My work does not require a white or black background. Also, it identifies which disease the plant is infected with.

Bauer [6] classifies the diseases in sugar beet leaves. A notable point is the fusing of images taken in a lab using an RGB and in an infrared camera which has the channels RED, GREEN and NIR (Near Infrared). The classification rate of the disease Cercosporabeticola achieved is 85%. In contrast, the classification rate of the disease Uromyces betae, however, is only 22% which the authors attributed to the small proportion of Uromyces images in their data set. Such discrepancies in accuracy between two diseases of the same plant can cause problems. Using machine learning makes sure this doesn't happen in my work.

Albashish[4] aims to automatically detect and classify diseases in plant leaves. The images were collected from the Al Ghor area in Jordan. The proposed solution starts by creating a device-independent colour space transformation structure and then converts the RGB values in the image to the colour space specified in the structure. K-means clustering is used to partition the leaf image into four clusters. They managed to achieve an accuracy of 89.5% in one of their models. Their approach differs from my project as the neural network used had been made manually and was very small as compared to the one this project uses.

Efficient Net V2 is the latest state-of-the-art model which has not been used for plant disease classification so far. Nguyen [7] used it to detect SARS-CoV-2 infections through recorded cough sounds. The dataset was provided by the AICovidVN 115M Challenge. The sounds were first converted into Log-Mel Spectrograms and then the EfficientNet V2 network was used to extract visual features. The model achieved an AUC score of 92.8% and ranked 1st place on the leaderboard of the AICovidVN 115M challenge. My project aims to reapply Efficient Net V2 to plant disease classification.

3. BACKGROUND

3.1 Dataset

Data collection is a very important step for all projects on deep learning. Most neural networks require huge amounts of data to be able to achieve decent levels of accuracy. Often the scope of research work in this field gets restricted according to the data available. The dataset used for this project is Plant Village[8]. Plant Village contains 54,303 expertly curated images of healthy and unhealthy leaf images divided into 38 categories by species and disease. Most research papers on detecting diseases in plant leaves use the Plant Village dataset to train and test their methods.

It has very consistent images with proper lighting and a monotone background which reduces the need for image pre-processing. They contain only one leaf which is placed at the center of the image. It contains images of size 256*256 pixels and are resized later for our use. These images were taken from the Kaggle datasets platform. Table 3.1 contains the exact number of training and validation images for each plant. The images had some data augmentation performed on them from the source itself. Due to this, all the categories had very minor imbalance in terms of number of images which made the training process easier. Table 3.2 contains the number of images in each category of the Grape plant to illustrate the above point.

Name	Images for training	Images for validation
Apple	7771	1943
Cherry	3509	877
Corn	7316	1829
Grape	7222	1805
Peach	3566	891
Bell pepper	3901	975
Potato	5702	1426
Strawberry	3598	900
Tomato	18345	4585

Table 3.1 Number of images of each plant

Name of Category	Number of images for training	Number of images for validation
Black Rot	1886	472
Esca (Black Measles)	1918	480
Isariopsis Leaf Spot (Leaf Blight)	1720	430
Healthy	1690	423

Table 3.2 Number of images of each category of the plant Grape

3.2 Data Augmentation

The main disadvantage of using deep learning is that it requires huge amounts of data to train the neural networks. Researchers have developed some techniques to get around this problem, such as Data Augmentation and Generative Adversarial Networks (henceforth to be referred as GAN). GAN will be discussed in section 3.3.

Data augmentation is a set of techniques that artificially increase the amount of data by generating variations on the existing images. A very important point to note here is that data augmentation cannot be used to create data, it can only be used to increase the size of the dataset. The most popular method is to perform simple manipulations on an image to create another one. A few examples of the manipulations include:

- Random Rotating
- Rescaling
- Vertical and horizontal flipping
- Adding noise
- Translation along x,y direction

The reason this works so effectively is that the neural model interprets these altered images as new ones.

Properly used, data augmentation can be used to increase the size of the dataset many times over and can help prevent data overfitting or the model developing a bias that reduces its accuracy during real-life use.

The dataset which was used in this project had already used data augmentation to increase its size, so there were multiple images of the same leaf in different orientations as depicted in Fig 3.2. and Fig 3.3 Some additional real-time data augmentation is done on the images in the data generator. It is also worth mentioning that care was taken to make sure that all the images of the same leaf were used for the same purpose – either training or validation.

Fig 3.2 Two images of the same grape leaf infected with black rot flipped horizontally in the training dataset

Fig 3.3 Two images of the same healthy grape leaf rotated randomly

3.3 GAN[9]

GAN stands for Generative Adversarial Networks. While data augmentation does simple manipulations on existing data to increase the dataset size, GANs create completely new images similar to the ones which are fed to it. Fig 4.1 tries to illustrate how a GAN model works.

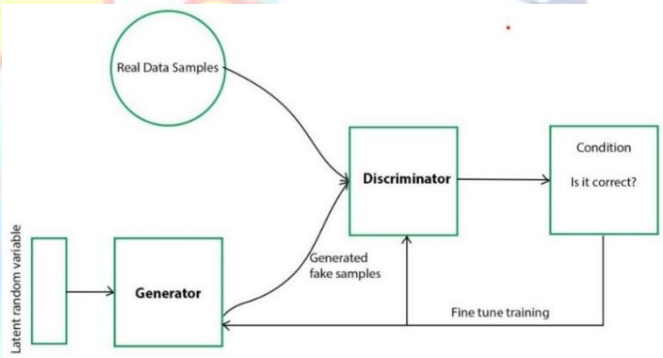

Fig 4.1 A classic GAN model [10]

The first neural network is named the Generator. It can be understood as the "artist", the one which creates the fake images based on the real input. The model consists of linear stacks of Conv2DTranspose, BatchNormalization and LeakyRelu layers. The Conv2DTranspose layer up samples its input, learning what is the best up sampling for the job. The BatchNormalization layer makes the overall process of training a neural network faster and more stable. A LeakyRelu is a modified type of activation function ReLU, with a small slope for negative values instead of a flat slope.

The second neural network is called the Discriminator. It can be understood as the "Art Critic" whose function is to distinguish a fake image created by

the Generator from a real one from the dataset. The Discriminator model is usually simpler than the generator to avoid dominating over it. So, it has fewer layers than the generator. Thus, the Generator and the Discriminator compete with each other in a zero-sum game (where one's loss is the other's gain). This competition goes on until the Generator manages to confuse the Discriminator enough by creating images identical to the original dataset.

4. MAIN WORK

The entire process of developing a model for plant disease recognition using deep learning is described in this section in detail. First, some pre-processing is done on the images from the dataset like resizing and using data augmentation. Then GAN is used to try to enrich the dataset by creating synthetic images. Next, the images are fed into the neural network for training it. Finally, validation data is given to the trained model to find out its accuracy and analyse the results. Each subcategory is described in detail as labelled below.

4.1 Data Preprocessing

The main purpose of using pre-processing instead of feeding the images directly into the model is to improve accuracy and reduce training time. Though the dataset used in this project eliminates the need to use common manipulations such as image segmentation and image morphing, a few tools are used to enrich the quality and quantity of the data being given to the neural network.

4.1.1 ImageDataGenerator

Feeding all the images in the dataset to the neural network at the same time would require a huge amount of memory resources and time. Thus, a data generator is used which generates batches of tensor image data. Using a data generator also allows us to do real-time data augmentation which helps in avoiding any biases the data and also creates a bigger dataset for the model to perform better predictive analysis. Lastly, a random seed is used for shuffling and transformations

Table 4.1 shows all the data augmentation techniques used.

Name	Description	Value Used
rotation_range	Degree range for random rotations	20
width_shift_range	Fraction of total width for random translation	0.2
height_shift_range	Fraction of total height for random translation	0.2
rescale	Multiplies the data by the value provided	1.0/256

4.1.2 GAN

The project uses DCGAN (Deep Convolutional GAN) which is an extension of the GAN architecture for using deep convolutional neural networks. The DCGAN model uses transpose convolution layers for the Generator and convolution layers for the Discriminator.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))].$$

The Conv layers use a convolutional stride, are built without max pooling and are not completely connected [10]. This model was chosen as it is used specially for image data.

A GAN model uses a value function V (G,D) depicted in Fig 4.2. The Generator tries to minimize V(G,D) by while the Discriminator tries to maximize it.

- D(x) is the discriminator's estimate of the probability that real data instance x is real.
- E_x is the expected value over all real data instances.
- G(z) is the generator's output when given noise z.
- D(G(z)) is the discriminator's estimate of the probability that a fake instance is real.
- E_z is the expected value over all random inputs to the generator (in effect, the expected value over all generated fake instances G(z)).

Fig 4.2 The value function of GAN network [10]

In my code, the training images are first converted to grayscale and resized to 28*28 pixels. Then they are converted to numpy arrays whose pixel values are changed from [0,255] to [-1,1]. All these manipulations are done to improve the efficiency of the GAN network. The numpy array is then sliced along the first dimension in a batch size of 256 and a dataset is created using the tf.data.Dataset.from_tensor_slices function. Finally, this dataset is fed into the Discriminator model.

4.2 Deep Learning

4.2.1 Efficient Net V2

The Efficient Net V2 (hereafter referred to as V2) is the state-of-the-art family of convolutional neural networks [11] released in 2021. They are the successors of the EfficientNet V1 (hereafter referred to as V1) neural networks. V2 builds upon V1 by using a few different features which increases the training speed and improves the parameter efficiency as compared to the existing popular models. As a result, they can reach state-of-the-art results while training faster (up to 11x) and smaller number of parameters (up to 6.8x). A few of these features are described below:

i) Fused-MBConv – V2 extensively uses both MBConv and Fused-MBConv. Fused-MBConv is a substituent to the MBConv layer which is slightly less accurate but more efficient. Hence it is used in the earlier layers which reduces the training time while the more accurate MBConv is used in the latter layers to finetune the weights (Lukyanenko, 2021). Fig 4.3 shows the structure of a MBConv and Fused MBConv layer.

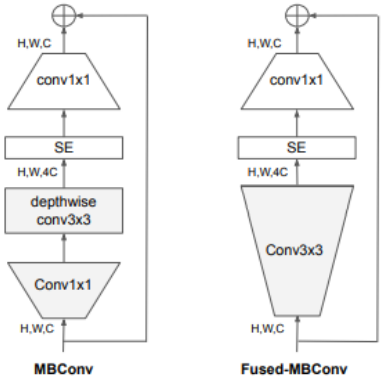

Fig 4.3 Structure of MBConv and Fused-MBConv[11]

ii) NAS search – NAS stands for Neural Architecture Search. It is a technique for automating the design of neural networks that makes the computer decide the architecture of the network rather than doing it manually. This is done by providing it with a search space which defines the region within which the NAS will search for the best possible architecture. It selects the one that best meets the objective of a given problem by maximizing a fitness function[12]. In V2, the search space consists of different convolutions (MBConv, Fused-MBConv), number of layers, kernel sizes and expansion ratios[13]. Fig 4.4 illustrates how an NAS works.

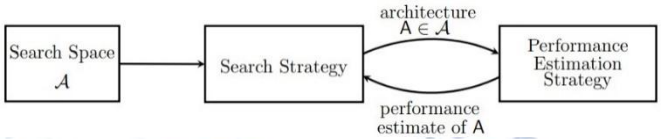

Fig 4.4 Abstract illustration of Neural Architecture Search methods. A search strategy selects an architecture A from a predefined search space A. The architecture is passed to a performance estimation strategy, which returns the estimated performance of A to the search strategy[14]

iii. Progressive Learning – Progressive learning refers to gradually increasing input image pixel sizes while training. It has a smaller expansion ratio for MBConv since smaller expansion ratios tend to have less memory access overhead. The model prefers smaller 3x3 kernel sizes, but adds more layers to compensate for the reduced receptive field resulting from the smaller kernel size. The creators of Efficient Net V2 show that if we gradually increase regularization along with image size, the results will be much better [13]

This project also makes use of transfer learning by importing the weights from the pre-training of EfficientNet V2 on the "imagenet" dataset containing over 14 million images and freezing most of the layers of the model which significantly reduced the training time. Fig 4.4 shows the architecture of the Efficient Net V2 S network in detail.

Stage	Operator	Stride	#Channels	#Layers
0	Conv3x3	2	24	1
1	Fused-MBConv1, k3x3	1	24	2
2	Fused-MBConv4, k3x3	2	48	4
3	Fused-MBConv4, k3x3	2	64	4
4	MBConv4, k3x3, SE0.25	2	128	6
5	MBConv6, k3x3, SE0.25	1	160	9
6	MBConv6, k3x3, SE0.25	2	272	15
7	Conv1x1 & Pooling & FC	-	1792	1

Fig 4.4 EfficientNet V2 S architecture [11]

4.2.2 Hyperparameter Tuning

Hyperparameters are the parameters which control the learning process within a specific model[15]. Changing the values of hyperparameters can affect the training time and accuracy. Hence, they need to be set to the most optimum values to get the best accuracy in the least training time possible. These values can vary for different datasets in the same model too.

During the training of the EfficientNet V2 network four hyperparameters were fine-tuned to give the best combination of accuracy vs training time as described below:

i. Batch Size – It signifies the number of the image samples to be used to calculate the loss percentage before changing the model weights again. The optimum batch size usually depends upon the total number of images in the dataset as well as the processing power available. This study experimented with batch sizes of 10,20,50 and 100 to find out which gives the best accuracy.

ii. Learning Rate – It decides how big of a change will be made to the model weights after the loss gradient is calculated. A smaller learning rate means the training will take more time as the weights will be updated by a small amount. If the learning rate is too high, the model might change the weights by a bigger amount than required. This study experimented with learning rates of 1e-2, 1e-3 and 1e-4 to find out which gives the best accuracy. The weights are governed by the below formula where "Wn" is the new weight, "Wo"' is the old weight and "a" is the learning rate.

$$W_n = W_o - a\left(\frac{\partial Error}{\partial W_o}\right)$$

iii. Optimizer – It is the algorithm/function that is used to change the weights or the learning rate of the network with an aim to minimize the loss function. This study experimented with Adam and SGD optimizers to find out which gives the best accuracy.

iv. Image size – The number of pixels in an image can also have an impact on how well a neural network works. A bigger image gives the network more data from which to analyse and extract features. On the other hand, a smaller image requires less training time. This study experimented with image sizes of 28*28, 128*28 and 256*256 to find out which gives the best accuracy.

5. RESULTS

Equipment: The code for the GAN and the network training was written on Google Colab and a GPU backend was used during the execution.

5.1 GAN Results

The GAN model was trained for 100 epochs. However, though the images formed after 100 epochs looked leaf shaped, but they weren't clear enough to be used for training the dataset. A possible explanation for this could be that the discriminator model was dominating over the generator and as a GAN works on a zero-sum game, the fake images produced couldn't improve. This is a very common problem while using GAN's and the same was encountered in this project.

The attempts to improve the GAN images' quality included changing the loss function, adding more layers to the generator network, increasing the dropout rate of the discriminator and changing the original images from RGB color format to grayscale. Small improvements were seen with each change but the overall image quality still had too much noise and hence the images were deemed unusable to be fed to the neural network. Figure 5.1 a, b, c, d, e shows the images produced by the generator after epoch 1, 25, 50, 75 and 100 respectively in the final code run.

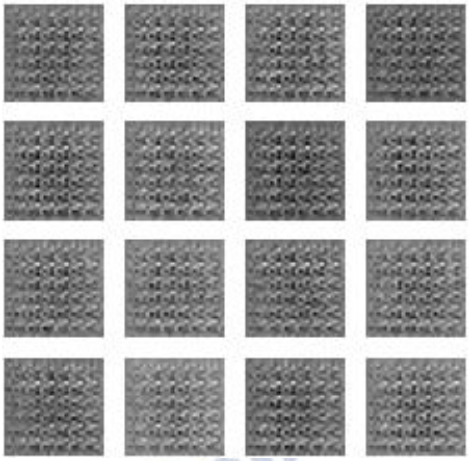

Fig 5.1 (a) After 1 epoch

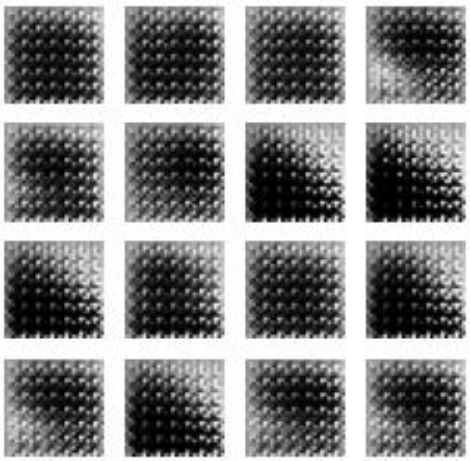

Fig 5.1 (b) After 25 epochs

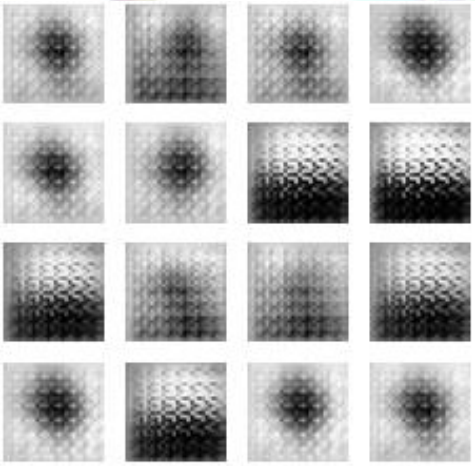

Fig 5.1 (c) After 50 epochs

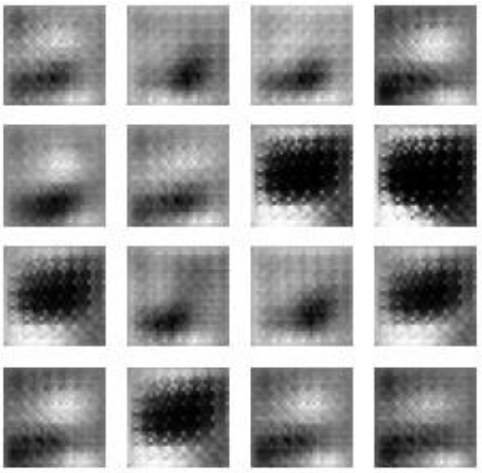

Fig 5.1 (d) After 75 epochs

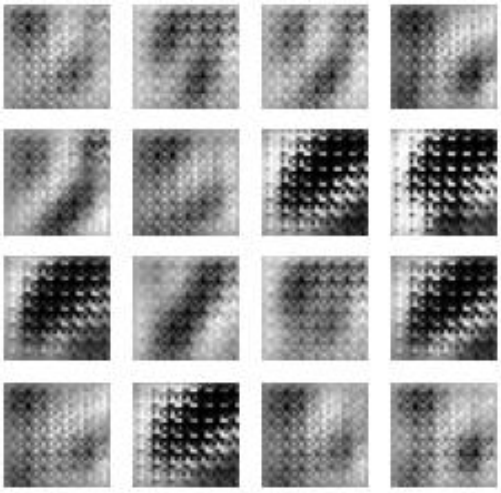

Fig 5.1 (e) After 100 epochs

5.2 Hyperparameter Tuning Results

The results presented in this section are based on training the original database and the augmented images only. Tests were performed on the data to find the optimum value for each of the four hyperparameters discussed in section 4. It is worth noting that the default values used for the other hyperparameters while fine tuning one were, SGD for optimiser, 256 for batch size, 10^-3 for learning rate and 10 for batch size.

For the optimiser, Adam and SGD were the two optimisers that were tested, the results of which are depicted in Fig 5.2. It was found that overall Adam performed slightly better than SGD and hence was chosen.

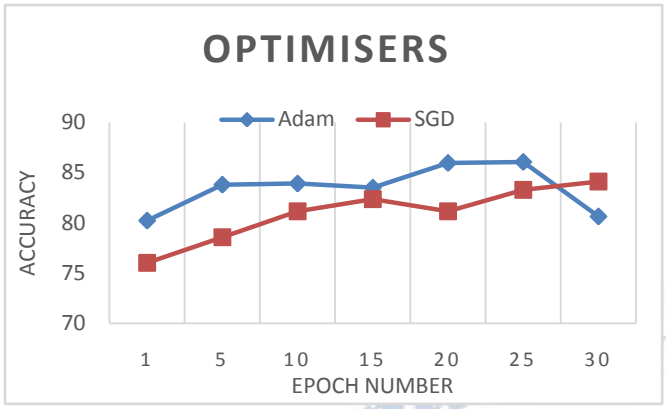

Fig 5.2 Optimisers

For the batch size, contrary to expectations and common practice, it was found that a batch size of 50 performed better than 10 or 20. A possible explanation for this could be the relatively high number of images. Also, the fact that due to data augmentation many images were similar to each other so the neural network had to look at more images at once to extract features more effectively and update the weights accordingly. The complete results are shown in Fig 5.3

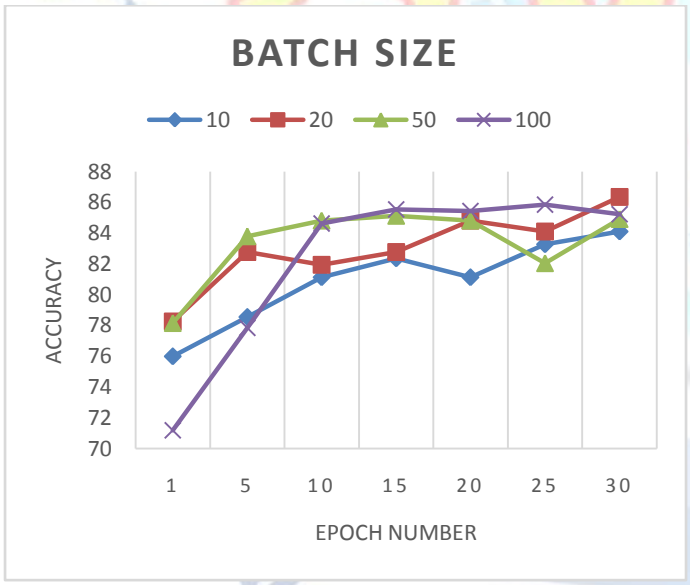

FIG 5.3 BATCH SIZE

For the learning rate, as expected it was found that 1e-3 and 1e-4 were better than 1e-2 as a very high learning rate led to the optimiser overshooting the minima of the loss function. The jumps in the graph also show how big the changes in the weights were for 1e-2. The complete results are shown in Fig 5.5

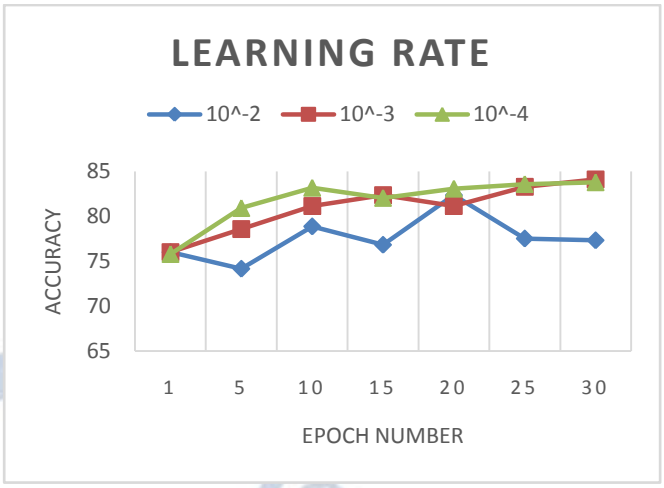

Fig 5.5 Learning rate

For the image size, it was found that 128*128 pixel images gave the best results. It is also worth mentioning here that the training time was the least for 28*28 pixel images and the most for 256*256 images which can be explained by the fact that a larger image size meant that the model had to perform more calculations. The complete results are shown in Fig 5.4

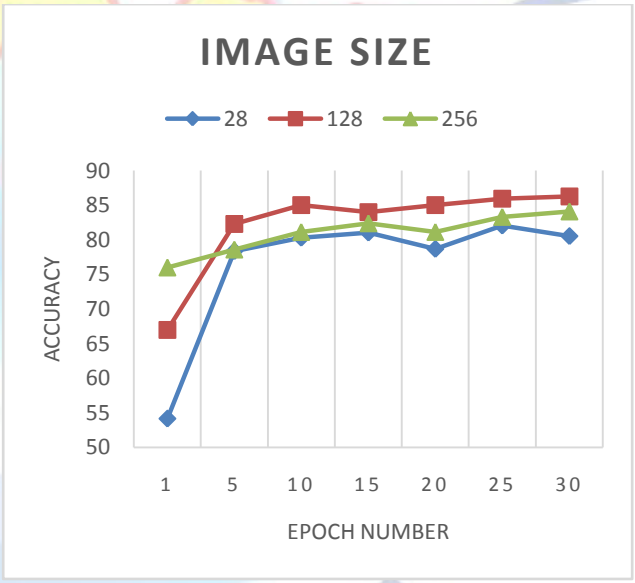

Fig 5.4 Image Size

5.3 Comparing between networks

The Efficient Net V2 family of networks consists of eight networks in all. To find out which specific network should be used, three models, namely B0, S and M, were tested. A more complex model with a higher number of parameters is often better for bigger datasets so the test was conducted on four plants: Peach (2 categories), Potato (3 categories), Apple (4 categories) and Tomato (10 categories). The results of testing the three networks

are given in Table 5.1. It was found that depending upon the data, B0 or S were the best performing networks.

Name	B0		S		M	
	Accuracy	Loss	Accuracy	Loss	Accuracy	Loss
Peach	0.94	0.17	**0.96**	**0.12**	0.90	0.26
Potato	**0.77**	**0.57**	0.75	0.61	0.57	0.91
Apple	**0.75**	**0.72**	0.68	0.76	0.55	1.02
Tomato	0.55	1.36	**0.62**	**1.13**	0.36	1.87

Table 5.1 Comparing three different neural networks

5.4 Overall Results

The overall results obtained on training are shown in Table 5.2. The possible factors influencing the results are the number and the quality of images. For example in Peach the disease being identified is the bacterial spot, which often gives the leaf an orangish colour, which makes it easier to identify from the healthy green leaves, resulting in the highest accuracy. Another important factor to consider is that not all plants have the same number of categories from which to distinguish a particular leaf. Hence, to make a comparison between plants with a different number of categories fair, the accuracy that would have been obtained in each using random guessing is compared to the improved accuracy of the neural network.

Name	Validation Accuracy	Random guess probability`	% better than random guess
Apple	75%	25%	300%
Cherry	95%	50%	190%
Corn	82%	25%	328%
Grape	67%	25%	268%
Peach	96%	50%	192%
Bell pepper	86%	50%	172%
Potato	77%	33%	233%
Strawberry	83%	50%	166%
Tomato	62%	10%	620%

Table 5.2 Overall Accuracy

6. FUTURE WORK

Possible areas for improvement in this research area include trying to obtain better images from the GAN network since this paper shows that the conventional GAN networks suffer from common issues like model collapsing and non convergence. A method to bypass these problems needs to be found.

Also, it is worth noticing that the datasets used for this project contained images with a monotone background and very low noise which reduced the need for a lot of data pre-processing and even helped in boosting the accuracy numbers. During practical use, however, getting such ideal data images will be very difficult, therefore there is a need to build deep learning models based on more realistic data. To achieve this, there is a need to build such a dataset, with images in different types of backgrounds, bad lighting, normal noise levels and also multiple leaves overlapping each other in one image.

7. CONCLUSION

Between 720 and 811 million people in the world faced hunger in 2020[16]. That is about 11% of the world's population. This figure is predicted to increase in the coming years as the global population increases. Hence, there is a critical need to reduce food crop wastage in the world, a major reason for which is diseases in plant leaves. This project aims to develop a more accurate and efficient method to detect and classify diseases in leaves of various plants as compared to the currently used manual method. In this study a machine learning based-approach is proposed and tested on plants of nine different species. The dataset used is the Plant Village dataset both for training and testing. The approach consists of two main phases: image pre-processing and training the network.

The first phase tries to use a technique called GAN to enrich the original dataset but the results show that further developments in GAN would make this easier to implement. The second phase makes use of the latest state-of-the-art Efficient Net V2 family of neural networks which are not only accurate but also reduce the training time and the number of parameters required to achieve the same accuracy. The experimental results from the second phase yield a classification accuracy of up to 96% which shows that the proposed method is a promising one and can

significantly support the accurate and automatic detection of leaf diseases. Another important point to note is that this model can be replicated for any other plant leaves too, if their dataset is made.

Conflict of interest statement

Authors declare that they do not have any conflict of interest.

REFERENCES

[1] Agriculture in India: Industry overview, market size, role in development...: IBEF. India Brand Equity Foundation. (n.d.).Retrieved July 1, 2022, from https://www.ibef.org/industry/agriculture-india

[2] Cagliarini, Adam & Rush, Anthony. (2011). Economic Development and Agriculture in India. RBA Bulletin. 15-22.

[3] Lu, J., Tan, L., & Jiang, H. (2021). Review on convolutional neural network (CNN) applied to plant leaf disease classification. Agriculture, 11(8), 707.

[4] Albashish, Dheeb &Braik, Malik & Bani-Ahmad, Sulieman. (2011). Detection and Classification of Leaf Diseases using K-means-based Segmentation and Neural-networks-based Classification. Information Technology Journal. 10. 10.3923/itj.2011.267.275.

[5] Barbedo, J. G. A. (2014). An automatic method to detect and measure leaf disease symptoms using digital image processing. Plant Disease, 98(12), 1709-1716.

[6] Bauer, S. D., Korc, F., Förstner, W., Van Henten, E. J., Goense, D., &Lokhorst, C. (2009). Investigation into the classification of diseases of sugar beet leaves using multispectral images. Precision agriculture, 9, 229-238.

[7] Nguyen, L. H., Pham, N. T., Do, V. H., Nguyen, L. T., Nguyen, T. T., Do, V. D., ... & Nguyen, N. D. (2021). Fruit-CoV: An Efficient Vision-based Framework for Speedy Detection and Diagnosis of SARS-CoV-2 Infections Through Recorded Cough Sounds. arXiv preprint arXiv:2109.03219.

[8] Hughes, D., &Salathé, M. (2015). An open access repository of images on plant health to enable the development of mobile disease diagnostics. arXiv preprint arXiv:1511.08060.

[9] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... &Bengio, Y. (2014). Generative adversarial nets. Advances in neural information processing systems, 27.

[10] Agrawal, R. A. (2021, October 20). An End-to-End Introduction to Generative Adversarial Networks (GANs). Analytics Vidhya. Retrieved July 5, 2022, from https://www.analyticsvidhya.com/blog/2021/10/an-end-to-end-introduction-to-generative-adversarial-networksgans/

[11] Tan, M., & Le, Q. (2021, July). Efficientnetv2: Smaller models and faster training. In International Conference on Machine Learning (pp. 10096-10106). PMLR.

[12] Gey, T. (2021, September 18). What is Neural Architecture Search? | Towards Data Science. Medium. Retrieved July 7, 2022, from https://towardsdatascience.com/what-is-neural-architecture-search-and-why-should-you-care-1e22393de461

[13] Lukyanenko, A. L. (2021, April 2). Paper Review: EfficientNetV2: Smaller Models and Faster Training. Andlukyane.Com. Retrieved July 3, 2022, from https://andlukyane.com/blog/paper-review-effnetv2

[14] Elsken, T., Metzen, J. H., &Hutter, F. (2019). Neural architecture search: A survey. The Journal of Machine Learning Research, 20(1), 1997-2017.

[15] Nyuytiymbiy, K. N. (2020, December 30). Parameters, Hyperparameters, Machine Learning | Towards Data Science. Towards Data Science. Retrieved July 3, 2022, from https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac

[16] World Health Organization. (2021). The State of Food Security and Nutrition in the World 2021: Transforming food systems for food security, improved nutrition and affordable healthy diets for all (Vol. 2021). Food & Agriculture Org..