A Plant Disease Classification with Residual Convolutional Neural Network

Katsuhiro Sasagaki
School of Science and Technology at Nottingham Trent University
Supervised by Dr. Hemantha Kodikara Arachchi

Abstract

In this study, the Residual Networks (ResNet) is used to train an image classification model to identify plant diseases of plant leaves. The model successfully identifies 38 different crop-diseases with 99.5% accuracy. The model is developed and trained using the MATLAB Deep Learning Toolbox. A mobile application was implemented on the iPhone 7 Plus incorporating the trained model to turn the mobile phone into a plant decease identification device.

1. Introduction

According to the Food and Agriculture Organization of the United Nations (FAO), plants cover 80% of the human diet and are an essential part of the diet, while 20% to 40% of global food production is lost due to plant pests and diseases. (Graaf, 2017) Reduced food production due to plant diseases can lead to higher food prices and even famine, in severe cases. Image classification systems trained by the machine learning approach have made rapid progress in recent years to reduce food losses due to these plant diseases. In order to minimize the food loss caused by pests and diseases, it is necessary to quickly identify the type of disease based on the patterns appearing on the leaves of the plants and take a prompt action for it, but it is not easy to diagnose them with the naked eye, which strongly depends on the experience and knowledge of the individual (Atila et al., 2021). The study presented in this paper is to discuss and analyze suitable methods to train the image classification model in terms of a recognition accuracy. The objectives are to train a classification model for plant diseases with plant leave images and to prototype a mobile app using the trained model. The section 2 explains terminologies and key concepts related to datasets, training and evaluation methods, and convolutional networks, and critically assesses these methodologies based on the related studies. The section 3 analyses the characteristics of the Residual Networks used in the project and how they differ from other Convolutional Networks. In Section 4, the dataset pre-processing, data augmentation methods and training options required for implementation are discussed, leading to conclusions with results in the recognition accuracy in Section 5.

2. Backgrounds: Terminologies and Related Studies

2-1. Dataset and Classification

Most of the datasets used in recent related research have been from an open access repository called PlantVillage. This repository is a copyright-free repository consisting of about 54,000 images, supervised and classified by experts into 38 plant-disease pairs (Table 1). The repository is available for download and commercial use on Kaggle, TensorFlow, GitHub, Mendeley and many other sites (Hughes, 2017).

1	AppleBlackRot	20	PepperBellHealthy
2	AppleCedarAppleRust	21	PotatoEarlyBlight
3	AppleHealthy	22	PotatoHealthy
4	AppleScab	23	PotatoLateBlight
5	BlueberryHealthy	24	RaspberryHealthy
6	CherryHealthy	25	SoybeanHealthy
7	CherryPowderyMildew	26	SquashPowderyMildew
8	CornCercosporaLeafSpotGrayLeafSpot	27	StrawberryHealthy
9	CornCommonRust	28	StrawberryLeafScorch
10	CornHealthy	29	TomatoBacterialSpot
11	CornNorthernLeafBlight	30	TomatoEarlyBlight
12	GrapeBlackRot	31	TomatoHealthy
13	GrapeEscaBlackMeasles	32	TomatoLateBlight
14	GrapeHealthy	33	TomatoLeafMold
15	GrapeLeafBlightIsariopsisLeafSpot	34	TomatoMosaicVirus
16	OrangeHaunglongbingCitrusGreening	35	TomatoSeptoriaLeafSpot
17	PeachBacterialSpot	36	TomatoSpiderMitesTwoSpottedSpiderMite
18	PeachHealthy	37	TomatoTargetSpot
19	PepperBellBacterialSpot	38	TomatoYellowLeafCurlVirus

Table 1. 38 Classified Plant-Disease Pairs on the PlantVillage Dataset

2-2. Data Augmentation

Data Augmentation is a method of increasing the amount of data by creating multiple variants from a limited amount of image data by applying tonal or geometric changes to the sample images - for example, changing the brightness or flipping the image left, right, up or down - in order to avoid the overtraining, where the training model responds to features that are essentially irrelevant (Shorten and Khoshgoftaar, 2019). For example, the PlantVillage repository images described above are all just a single target leaf on a plain background, and if these images are used as input data as is, overtraining on the training data will occur. When this occurs, there is a clear performance difference between the training data, on which the model has been trained, and the test data, on which the model has not yet been trained, resulting in a model with poor generalization ability. Although the uniformity of the input data is not the only cause of the overfitting, bulking up the input data by making various changes to the original image data is an important method to prevent the overfitting.

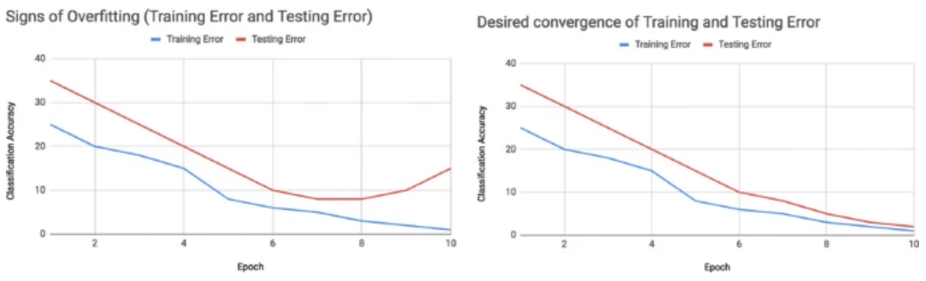


Figure 1. Overfitting (Left) and Desired Convergence (Right) (Shorten and Khoshgoftaar, 2019)

The research by Mohanty et al. applies two data augmentation methods to image datasets: greyscale and segmentation. The segmentation method removes the background from the image and standardizes the color, brightness and saturation of the leaves to prevent color casts and create a dataset free from potential bias (Mohanty et al., 2016).

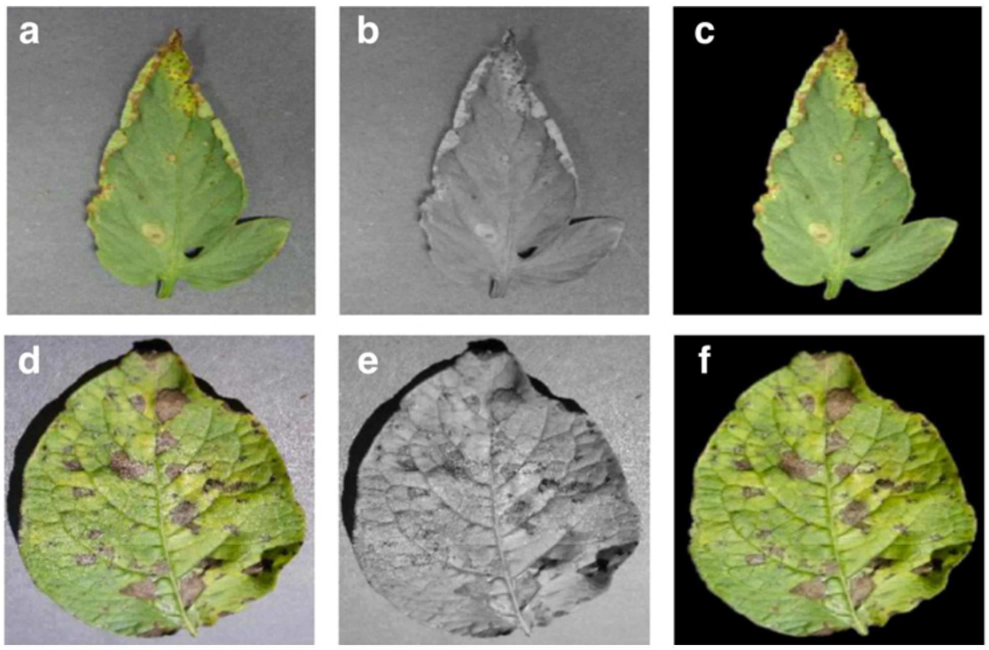
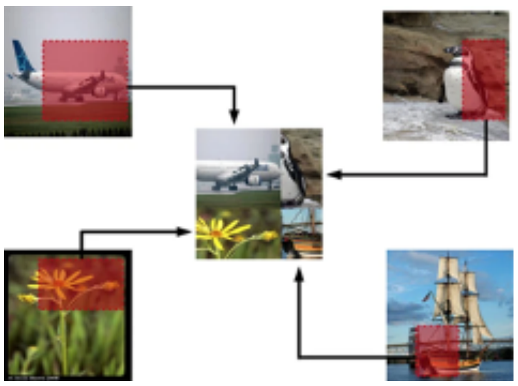


Figure 2. Data Augmentation Methods: Grayscale (b) and (e), and Segmentation (c) and (f), from original (a) and (d), (Mohanty et al., 2016)

A study by A. KP and J. Anitha applies only the most basic data enhancements such as rotation, shift and zoom (A and Anitha, 2021), while Zhao et al. add salt and pepper noise, Gaussian noise and images with altered luminance, saturation and contrast to the basic data augmentation, thereby increasing the diversity of their dataset (Zhao et al., 2021). Cropping & Patching (Fig. 3.) and Random Erasing (Fig. 4.), introduced in the paper by Shorten and Khoshgoftaar are not often used, but these methods can be used to prevent overtraining as well. Cropping & Patching is a method of combining images selected from several completely unrelated genres with a relevant image. Random Erasing is a method of randomly erasing parts of the target image.



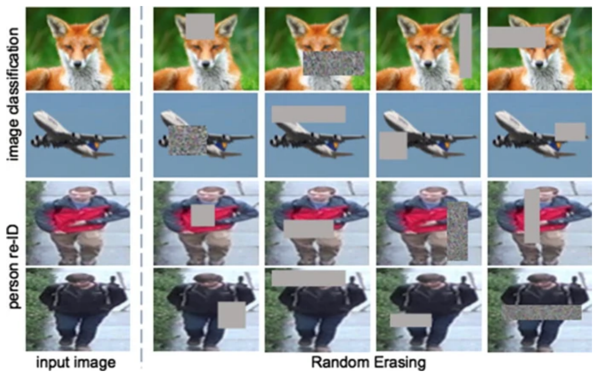


Figure 3. Cropping & Patching <left> Random Erasing <right> (Shorten and Khoshgoftaar, 2019)

2-3. Training Methods

2-3-1. Training-Testing Ratio on the Dataset Images

The training of an image recognition model using a neural network is classified into two stages: training and testing. A part of the dataset is loaded into the neural network for training, while the remaining part is used as unknown data to evaluate the trained model. If there is a large discrepancy in recognition accuracy or error rates between the two datasets, one for training and one for testing, the model is judged to be inaccurate. Training is also repeated until these rates converge.

It is known that optimizing this training-testing data ratio can prevent overfitting and improve recognition accuracy, but a 80-20 training-testing ratio is commonly used. In the research of Mohanty et al., which aims to recognize all the 38 types classified in the PlantVillage dataset, several dataset partitioning ratios were tried, and the recognition accuracy was compared for each combination of CNNs used as shown below. As a result, the combination of GoogleNet as CNN, 20-80 training-testing Ratio and Transfer Learning achieved the highest recognition accuracy of 99.34% (Mohanty et al., 2016).

2-3-2. Pre-Trained Model or Model Trained from Scratch

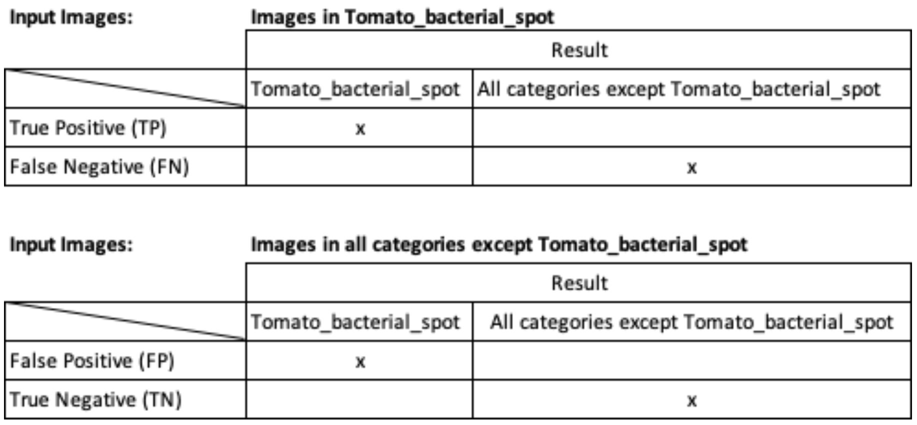
There are various methods used to build the training model, but there are mainly two types of methods: one is to train the model from scratch, and the other is to train the model based on a model trained on a relevant data set, which is called the Transfer Learning. Training from scratch generally requires a large amount of training data and is expensive in terms of time and computational resources, but it is more customizable and is used when there are no models trained on similar data sets. In the study of Venkataramanan and Agarwal, who built a training model for leaf disease recognition from scratch using VGGNet16, the accuracy of the training and testing sets was only about 50.26 % at 20 epochs. Therefore, the model was switched to transfer learning to achieve higher accuracy. As a result, an accuracy of 78% was achieved, indicating that a model trained from scratch is not practical in terms of recognition accuracy. In this experiment, the combination of 80:20 training-testing ratio and transfer learning also showed the highest recognition accuracy (Venkataramanan and Agarwal, 2019). Also, in the work of Oztel, Yolcu and Oz, who experimented with two CNNs, VGG16 and AlexNet, comparing models built from scratch and transfer learning models, the transfer learning model shows an excellent result in terms of training speed and recognition accuracy (Oztel, Yolcu and Oz, 2019).

2-4. Evaluation Methods & Metrics

The performance of the image recognition model is evaluated by Accuracy (mean Average Precision: mAP) and F1 score. Although if the amount of data set for each recognition category is distributed in a balanced way, it would be evaluated with Accuracy, the F1 score may be a more accurate assessment when the number of images in the dataset for each category varies greatly, as in the case of PlantVillage, overfitting occurs in the category with the larger number of images (Shi, C., 2020). These two measures are calculated from the values of Recall and Precision, which are calculated by the following formula:

- Recall (%) = (True Positive / True Positive + False Negative) * 100
- Precision (%) = (True Positive / True Positive + False Positive) * 100

True Positive (TP) is the number of correctly recognized images in the target category. For example, these images would be classified as TP if the TomatoBacterialSpot images are recognized as TomatoBacterialSpot. False Negative (FN) is the number of images the target category is recognized as a different category. For example, if an input image of TomatoBacterialSpot is recognized as TomatoLeafMold, this would be counted as FN. False Positive (FP) is the sum of the number of wrongly recognized images in categories other than the target category. For example, the FP is the total number of images that are recognized as TomatoBacterialSpot in the 9 Tomato categories other than TomatoBacterialSpot in the PlantVillage dataset. In addition, the total number of correctly recognized images in the categories other than TomatoBacterialSpot category is counted as True Negative (TN) for the TomatoBacterialSpot category (Atila, Ü, et al., 2021).



Input Images: **Images in Tomato_bacterial_spot**

	Result	
	Tomato_bacterial_spot	All categories except Tomato_bacterial_spot
True Positive (TP)	x	
False Negative (FN)		x

Input Images: **Images in all categories except Tomato_bacterial_spot**

	Result	
	Tomato_bacterial_spot	All categories except Tomato_bacterial_spot
False Positive (FP)	x	
True Negative (TN)		x

Fig. 6, Chart for TP, FN, FP, and TN for TomatoBacterialSpot Class

- Accuracy (mAP)

Using the above indicators, Accuracy can be calculated using the following formula:

Accuracy (%) = [(TN + TP) / (TN + FP + TP + FN)] * 100

(N B, H., 2019)

- F1 Score

F1 Score, on the other hand, can be calculated by Precision and Recall as follows:

F1 Score (%) = 2 * [(Precision * Recall) / (Precision + Recall)] * 100

(N B, H., 2019)

2-5. State-Of-The-Art Neural Network

Convolutional Neural Networks (CNNs) are one of the main artificial neural network algorithms used in recent machine learning for image recognition, inspired by the neuronal connectivity patterns in the visual cortex of animals that process visual information. CNNs are particularly popular in the fields of image and video recognition and object classification because they require minimal preprocessing of the input data and can efficiently extract features by learning the functions of filters traditionally designed by humans. There are various architectures of CNNs, but they are mainly composed of the following four layers (Indolia et al., 2018).

1) Convolution Layer
2) Pooling Layer
3) Flatten Layer
4) Fully Connected Layer

2-5-1. Convolution Layer

The convolution layer extracts high-level features, such as edges, from data that contains a lot of information, just as humans see things. To do this, it performs a convolution operation on the input data using a filter called a kernel filter, and reconstructs the features extracted from the original data within the size of the kernel filter (Indolia et al., 2018). The following is an example of feature extraction performed by applying a 3 x 3 x 1 kernel filter to a 5 x 5 x 1 input image.

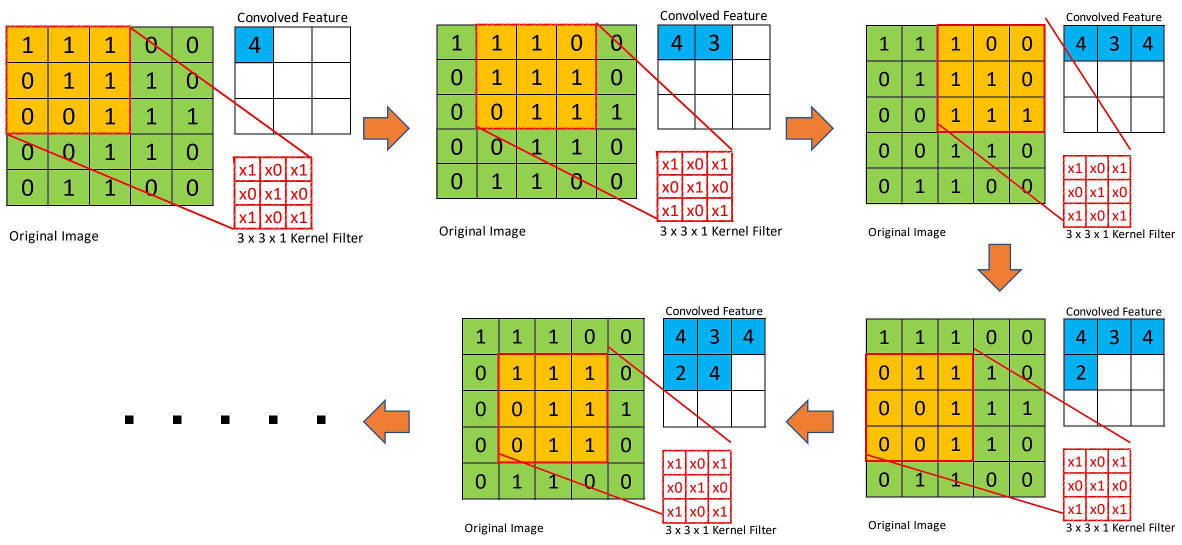


Figure 4. A Key Concept of the Convolutional Operations

The convolution layer of CNN can thus obtain an output with a smaller spatial size without losing the features of the original image data. Although it shows only one layer for this example, for normal RGB image data, the final result is the sum of the convolutional outputs for each of the three RGB layers (Saha, 2018).

2-5-2. Max Pooling Layer

In the max pooling layer, the operation of extracting dominant features is performed on the convolved features output from the convolution layer. Similar to the convolution layer, this process can reduce the spatial size of the convolved features and thus reduce the load required for data processing (Indolia et al., 2018). In the following example, 3 x 3 max pooling is performed on a 5 x 5 convolved feature.

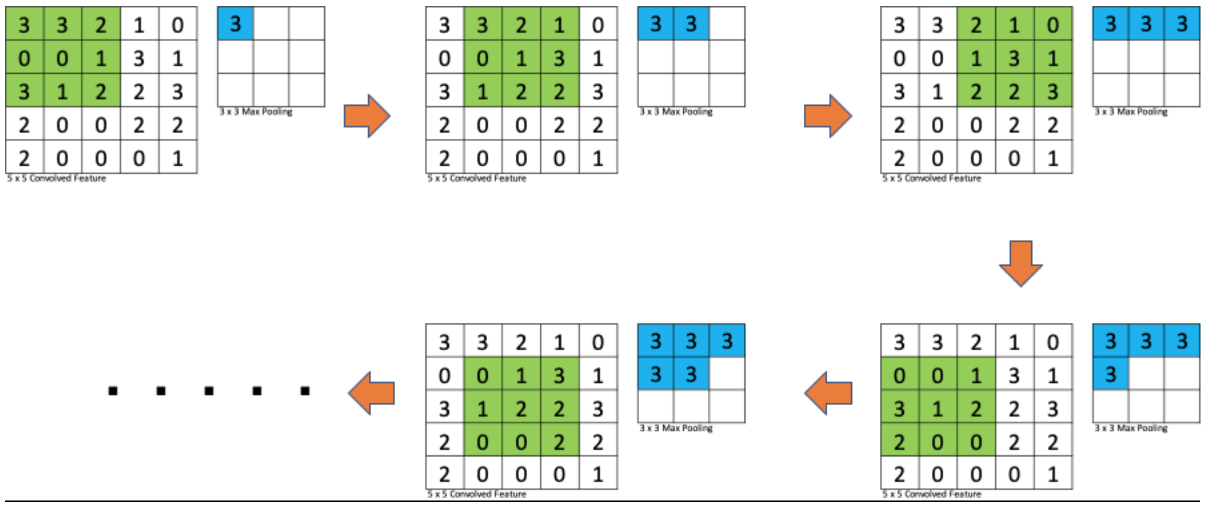


Figure 5. A Key Concept of the Max Pooling Operations

This process returns the maximum value in the 3 x 3 range and repeats the shift in the same way as the convolution layer. Another pooling process is the average pooling, which returns the average value over the range.

2-5-3. Flatten Layer

The flatten layer converts the output of the n x m matrix obtained from the pooling layer into a 1 x m column vector and outputs it to the next fully connected layer (Saha, 2018). The following is an example of flattening a 3 x 3 pooled feature into a 1 x 9 Column Vector.

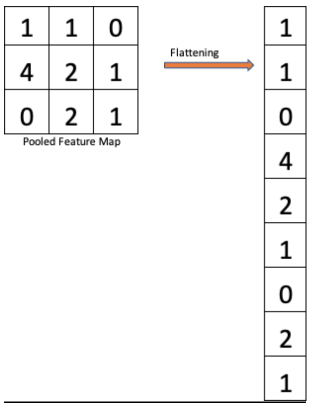


Figure 6. A Key Concept of the Flattening Operation

2-5-4. Fully Connected Layer

The fully connected layer accepts all the outputs of the flatten layer as inputs. Each of these inputs is connected to the same number of outputs as the recognition categories, and the outputs from these outputs are passed to a non-linear function such as the Rectified Linear Unit (ReLU) to return the final recognition result. The process of outputting the recognition result by passing through each layer of the convolution, pooling, flatten, and fully connected is called the forward propagation. The weights used in the fully connected layer and the kernel filters used in the previous layers are recalculated and updated successively by the loss function after the output. This process is called the back propagation (Yamashita et al., 2018).

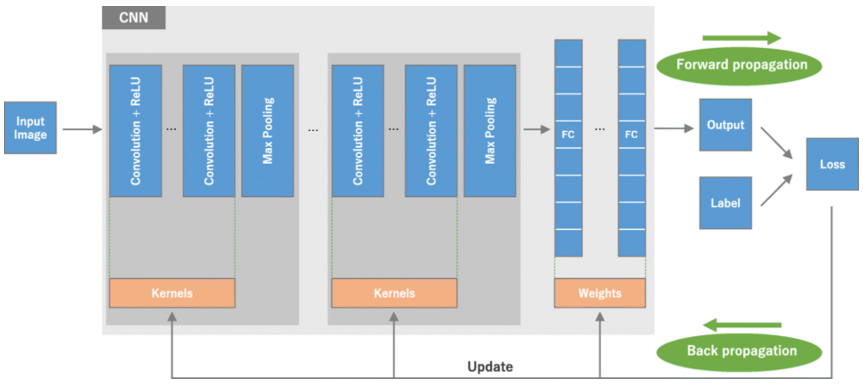


Figure 7. Flow Chart for Forward & Back Propagations (Yamashita et al., 2018)

3. Residual Networks

3-1. Features That Make ResNet Different from Other CNNs

For an overview of the technological development in the field of image classification, the evolution of the winning model in the ImageNet Large Scale Visual Recognition Competition (ILSVRC), an image recognition competition, provides an insight into how CNNs have been developed. Until around 2010, the mainstream methods were a combination of feature vectors and classifiers such as Support Vector Machines (SVMs), but after AlexNet won the ILSVRC in 2012 by a wide margin over conventional models, CNNs attracted even more attention. The ResNet used in this project is the winning model of the 2015 ILSVRC (Zhai et al, 2020), but it was a major breakthrough in models after AlexNet. even in 2022, most new CNN models since then have been based on improvements of this ResNet. The revolutionary aspect of ResNet is that it reduces the number of convolution parameters, which in turn reduces the depth and ResNet does not simply pass the transformation F(x) by some processing block to the next layer, as in normal networks, but shortcuts the input x to that processing block and passes H(x)=F(x)+x to the next layer (He et. al, 2015). The processing unit including these shortcuts is called the residual module: in ResNet, the gradient is transmitted directly to the deeper layers during backpropagation through the shortcuts, enabling efficient learning even in very deep networks.

3-2. Analysis and Configurations for Residual Module for ResNet50

In this project, ResNet50 is used and it consists of a multi-layered structure of blocks called the Residual Module. A single Residual Module is made up of Convolution, Batch Normarization and Rectified Linear Units (ReLU) layers connected in series, and shortcuts in parallel to them (He et. al, 2015). ResNet50 is made up of 16 Residual Modules connected in series with other functions, and the total number of layers is 177.

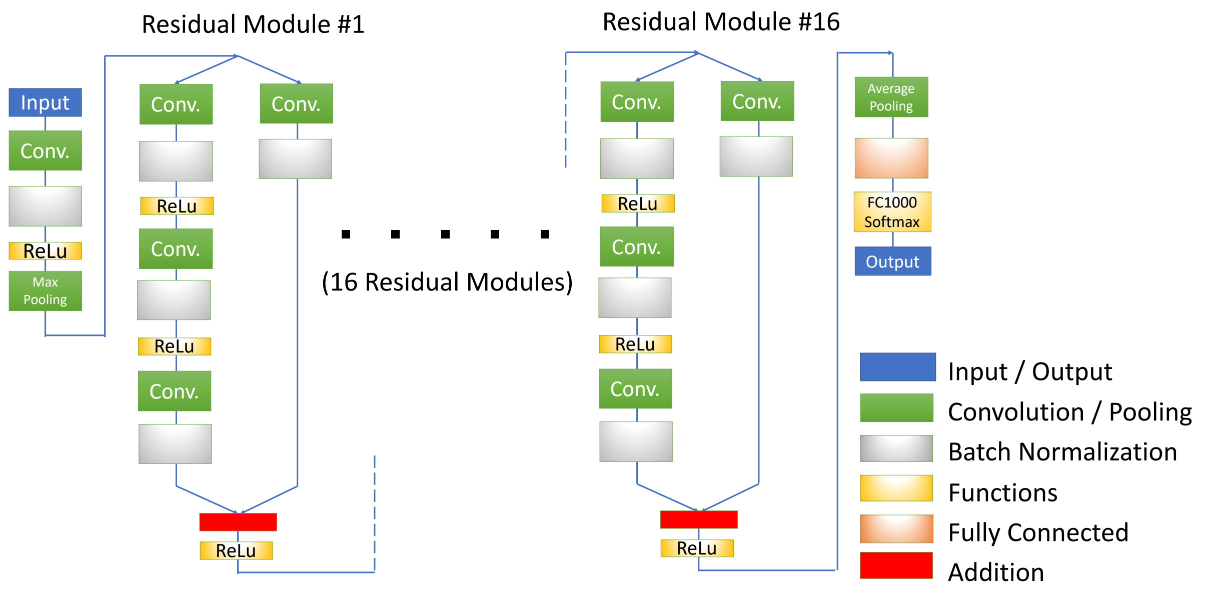


Figure 8. Layer Structure of ResNet50

3-2-1. Batch Normarization

In deep networks, there is a problem that learning does not proceed efficiently due to the internal covariate shift, in which updating the parameters of one layer causes the distribution of inputs to the next layer to change significantly from batch to batch. Batch normalization (Boxes in gray, Figure 8) is a method for stabilizing and speeding up learning by normalizing this internal covariate shift and allowing each layer to learn as independently as possible (Ioffe et al, 2015). Incorporating it into deep networks enables efficient learning of deep networks, and batch normalization has become standard in models since ResNet emerged.

3-2-2. Rectified Linear Unit (ReLU)

Traditionally, $f(x) = tanh(x)$ and $f(x) = (1 + e^{-x})^{-1}$ have been used as nonlinear activation functions, but learning has been accelerated by using the ReLU (Boxes in orange, Figure 8) defined as $f(x) = max(0, x)$. It speeds up the learning process. This is because it solves the gradient vanishing problem that occurs when using conventional activation functions in deep networks.

3-2-3. A Bottleneck Building Block for ResNet50

ResNet50 and later ResNet versions have achieved a significant improvement in recognition accuracy compared to earlier ResNet versions, ResNet34, by incorporating the Bottleneck Building Block in its Residual Modules (He et. al, 2015). In ResNet34, the 3 x 3 convolution is performed twice in one Residual Module, whereas in ResNet50, a 1 x 1 convolution is used for a dimension reduction, followed by a 3 x 3 convolution and then a further 1 x 1 convolution is performed to restore the original dimension (Figure 9), thus allowing deeper models to be built while maintaining the same computational complexity as in ResNet34.

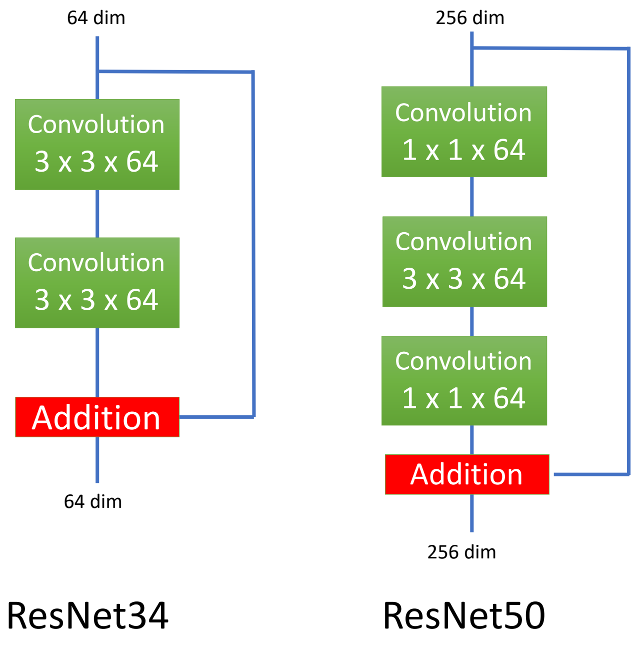


Figure 9. ResNet50 Building Block (Right)

4. Implementation

4-1. Dataset Structure

Images are obtained from the Mendeley Data providing the PlantVillage dataset. Datasets are provided with and without augmentation (J and Gopal, 2019). The dataset with augmentation is used for a training process, and the dataset without augmentation is used for a validation process after the model is trained. Each dataset has already been classified into 14 crops including 38 crop-disease pairs (Table 2).

	Classes	Number of Images			
		/w Augmentation			/wo Aug
		Original	Training	Testing	Validation
1	AppleBlackRot	1987	800	200	100
2	AppleCedarAppleRust	1760	800	200	100
3	AppleHealthy	2008	800	200	100
4	AppleScab	2016	800	200	100
5	BlueberryHealthy	1816	800	200	100
6	CherryHealthy	1826	800	200	100
7	CherryPowderyMildew	1683	800	200	100
8	CornCercosporaLeafSpotGrayLeafSpot	1642	800	200	100
9	CornCommonRust	1907	800	200	100
10	CornHealthy	1859	800	200	100
11	CornNorthernLeafBlight	1908	800	200	100
12	GrapeBlackRot	1888	800	200	100
13	GrapeEscaBlackMeasles	1920	800	200	100
14	GrapeHealthy	1692	800	200	100
15	GrapeLeafBlightIsariopsisLeafSpot	1722	800	200	100
16	OrangeHaunglongbingCitrusGreening	2010	800	200	100
17	PeachBacterialSpot	1838	800	200	100
18	PeachHealthy	1728	800	200	100
19	PepperBellBacterialSpot	1913	800	200	100
20	PepperBellHealthy	1988	800	200	100
21	PotatoEarlyBlight	1939	800	200	100
22	PotatoHealthy	1824	800	200	100
23	PotatoLateBlight	1939	800	200	100
24	RaspberryHealthy	1781	800	200	100
25	SoybeanHealthy	2022	800	200	100
26	SquashPowderyMildew	1736	800	200	100
27	StrawberryHealthy	1824	800	200	100
28	StrawberryLeafScorch	1774	800	200	100
29	TomatoBacterialSpot	1702	800	200	100
30	TomatoEarlyBlight	1920	800	200	100
31	TomatoHealthy	1926	800	200	100
32	TomatoLateBlight	1851	800	200	100
33	TomatoLeafMold	1882	800	200	100
34	TomatoMosaicVirus	1790	800	200	100
35	TomatoSeptoriaLeafSpot	1745	800	200	100
36	TomatoSpiderMitesTwoSpottedSpiderMite	1741	800	200	100
37	TomatoTargetSpot	1827	800	200	100
38	TomatoYellowLeafCurlVirus	1961	800	200	100

Table 2. Classified Crop-Disease Dataset and Number of Images for Each Class

For each class, 1,000 images are randomly chosen from the original dataset with augmentation, then the selected images are randomly split in a ratio of 80:20, for the training and for the testing, respectively. Therefore, 38,000 images are used for the training process.
For the validation process, the dataset without augmentation is used. 100 images are randomly chosen for each class from the dataset.

4-2. Pre-Data Augmentation

The PlantVillage dataset is a very high-quality dataset that has been reviewed by plant disease experts, but the original dataset is known to have a bias in the number of images by class. For example, Orange's Citrus Greening, Soybean Healthy, and Tomato's Yellow Leaf Curl Virus classes, have more than 5,000 images each, whereas the Potato Healthy (152 images) and Cedar Apple Rust (275 images) classes have relatively a smaller number of images (Table 3).

Dataset without Augmentation					
	Label	Count		Label	Count
1	AppleScab	630	20	PepperBellHealthy	1478
2	AppleBlackRot	621	21	PotatoEarlyBlight	1000
3	**AppleCedarAppleRust**	**275**	22	PotatoLateBlight	1000
4	AppleHealthy	1645	**23**	**PotatoHealthy**	**152**
5	BlueberryHealthy	1502	24	RaspberryHealthy	371
6	CherryPowderyMildew	1052	**25**	**SoybeanHealthy**	**5090**
7	CherryHealthy	854	26	SquashPowderyMildew	1835
8	CornCercosporaLeafSpotGrayLeafSpot	513	27	StrawberryLeafScorch	1109
9	CornCommonRust	1192	28	StrawberryHealthy	456
10	CornNorthernLeafBlight	985	29	TomatoBacterialSpot	2127
11	CornHealthy	1162	30	TomatoEarlyBlight	1000
12	GrapeBlackRot	1180	31	TomatoLateBlight	1909
13	GrapeEscaBlackMeasles	1383	32	TomatoLeafMold	952
14	GrapeLeafBlightIsariopsisLeafSpot	1076	33	TomatoSeptoriaLeafSpot	1771
15	GrapeHealthy	423	34	TomatoSpiderMitesTwoSpottedSpiderMite	1676
16	**OrangeHaunglongbingCitrusGreening**	**5507**	35	TomatoTargetSpot	1404
17	PeachBacterialSpot	2297	**36**	**TomatoYellowLeafCurlVirus**	**5357**
18	PeachHealthy	360	37	TomatoMosaicVirus	373
19	PepperBellBacterialSpot	997	38	TomatoHealthy	1591

Table 3. Numbers of Images for each class of the PlantVillage Dataset /wo Augmentation

A dataset with widely varying numbers of images depending on classes introduces an unnecessary bias in the training model. To prevent this, the number of images in each class needs to be matched to the class with the lowest number of images, Potato Healthy (152 images) in this case. However, this number of images is not sufficient to improve recognition accuracy. For example, below are the results of a training conducted using three different CNN models for 14 classes and 100 images per class. For AlexNet and Xception, these show below 90%, and even ResNet50 has shown only 92% (Table 4).

CNN Models	AlexNet	Xception	Resnet50
Achieved Accuracy	87.3 %	87.3 %	92.0 %
Epochs done	6	6	6
Train-Testing Ratio	80:20	80:20	80:20
Num of Classes	14	14	14
Num of Images per Class	100	100	100

Table 4. Training Experiment with 3 CNNs and 100 Images

On the other hand, the results with ResNet50 in 3 different conditions with more than 1,000 images for each class, generally achieved around 99+% (Table 5). This indicates that around 1,000 images per class are required to train this model.

ResNet50	Condition 1	Condition 2	Condition 3
Accuracy	98.25 %	99.27 %	99.30 %
Epochs done	6	6	6
Train-Eva Ratio	80:20	80:20	80:20
Num of Classes	2	10	10
Num of Images per Class	1,000	2,127	1,000

Table 5. Training Experiment with 3 Conditions and 1,000+ Images in ResNet50

The project uses an augmented data set pre-processed by Arun Pandian J and Geetharamani Gopal (J and Gopal, 2019). As mentioned in the previous section, there are various methods of data augmentation. The dataset provided by Mendeley Data is bulked up to four times the number of images by rotation, which is one of the most common augmentation methods. Below is one of the original images (Top-Left, Figure 9) of the Apple Black Rot class and three images that have been augmented from it. Three angles of rotation are used: 30˚ (Top-Right, Figure 9), 90˚ (Bottom-Right, Figure 9) and 270˚ (Bottom-Left, Figure 9).

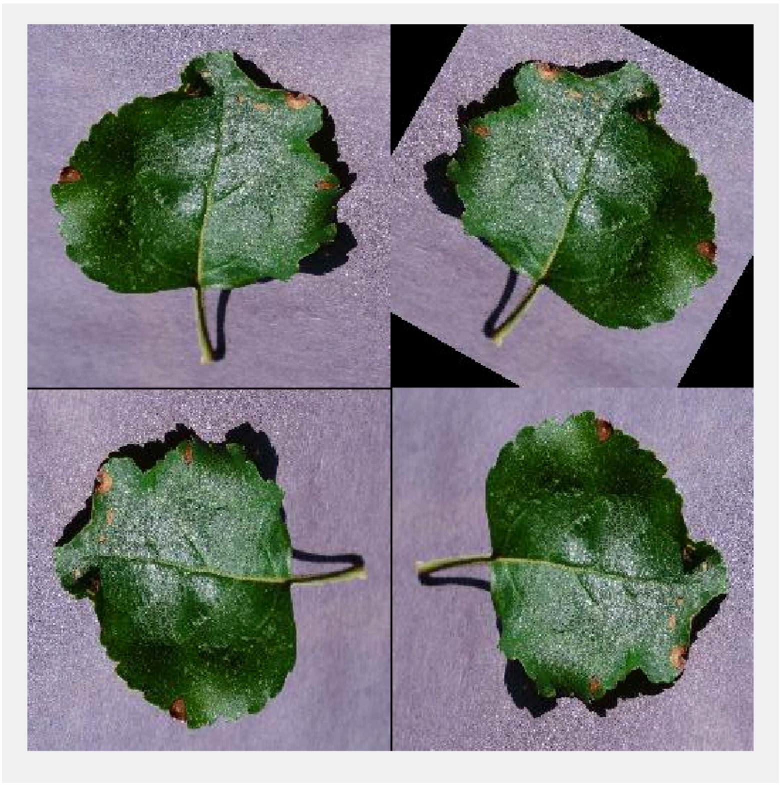


Figure 9. An Image Rotation for Image of the Apple Rot Class

4-3. Data Augmentation in MATLAB

The dataset is fed by MATLAB script, then augmented again by some MATLAB functions before the training starts. Here, 3 common augmentation methods are used, Random Reflection, Random Translation, and Random Scaling. Here are some details of the augmentation options applied to the dataset.

<u>- 4-3-1. RandXReflection</u>

When the *RandXReflection* option is set as *true* in the *imageDataAugmenter* function, each image in the dataset has a 50% chance of being flipped horizontally (MathWorks a- ImageDataAugmenter, 2022).

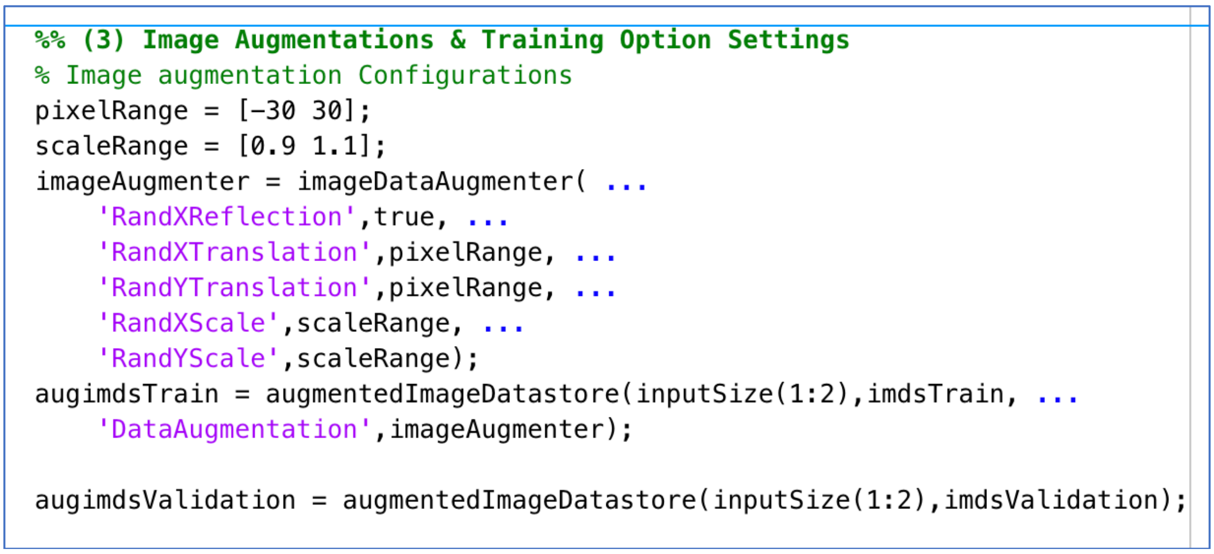
- 4-3-2. RandXTranslation / RandYTranslation

Specifying these options followed by a two-element numeric vector in pixels randomly shift the image horizontally or vertically within that pixel range (MathWorks a - ImageDataAugmenter, 2022).

- 4-3-3. RandXScale / RandYScale

Specifying these options followed by a two-element numeric vector in ratios randomly stretch the image horizontally or vertically within that ratio (MathWorks a - ImageDataAugmenter, 2022).

In the MATLAB script, *RandXReflection* is set as *true*, the pixel range is set as [-30, 30] for *RandXTranslation* and *RandYTranslation,* and the scale range is set as [0.9, 1.1] for *RandXScale* and *RandYScale,* as the source code shows below.



```
%% (3) Image Augmentations & Training Option Settings
% Image augmentation Configurations
pixelRange = [-30 30];
scaleRange = [0.9 1.1];
imageAugmenter = imageDataAugmenter( ...
    'RandXReflection',true, ...
    'RandXTranslation',pixelRange, ...
    'RandYTranslation',pixelRange, ...
    'RandXScale',scaleRange, ...
    'RandYScale',scaleRange);
augimdsTrain = augmentedImageDatastore(inputSize(1:2),imdsTrain, ...
    'DataAugmentation',imageAugmenter);

augimdsValidation = augmentedImageDatastore(inputSize(1:2),imdsValidation);
```

Figure 10. A Part of the MATLAB Script for the Image Augmentation (MathWorks b - Train Deep Learning Network to Classify New Images, 2022) - *Appendix01_TrainingProcedures.html*

The input size of the image is also changed here to the size required by the CNN used (224 x 224 for ResNet50). These Augmentation options are then applied to the training dataset, but not to the Validation dataset.

4-4. Training Options/Configurations

The MATLAB script specifies various training options. Here are the most significant options for the training, which have been applied to the final trained model.

- 4-4-1. Optimizers for Training Network

There are 3 optimizers are available as the leaning solver, Stochastic Gradient Method with Momentum (SGDM), Root Mean Squared Propagation (RMSProp) and Adaptive Moment Estimation (Adam) (MathWorks - trainingOptions c, 2022). One of those can be selected. The Stochastic

Gradient Method with Momentum (SGDM) is used to train this model. The SGDM generates a momentum term by using weights that decays exponentially with respect to the iteration time (Liu and Luo, 2020).

- 4-4-2. Minimum Batch Size

This can be specified as *MiniBatchSize* in the *trainingOptions*. *MiniBatchSize* can be used to evaluate the gradient of the loss function and to update the weights for the next iteration. Although 128 is supposed to use as a default value, 10 is used for this model. A study by Masters and Luschi indicates that a smaller *MiniBatchSize* makes a newer gradient computation that gives a training more stable convergence, whereas a larger *MiniBatchSize* reduces a range of the learning rate, which can bring the training less reliable testing ability and convergence (Masters and Luschi, 2018).

- 4-4-3. Maximum Epochs

Epoch is the number of times that the training cycles through the entire data set, and *MaxEpochs* specifies this number of cycles. There is no generally recommended number, but the process needs to be repeated until Loss and Accuracy have converged sufficiently that no further change is expected (MathWorks - trainingOptions c, 2022). For this model, 6 is used.

- 4-4-4. Initial Learning Rate

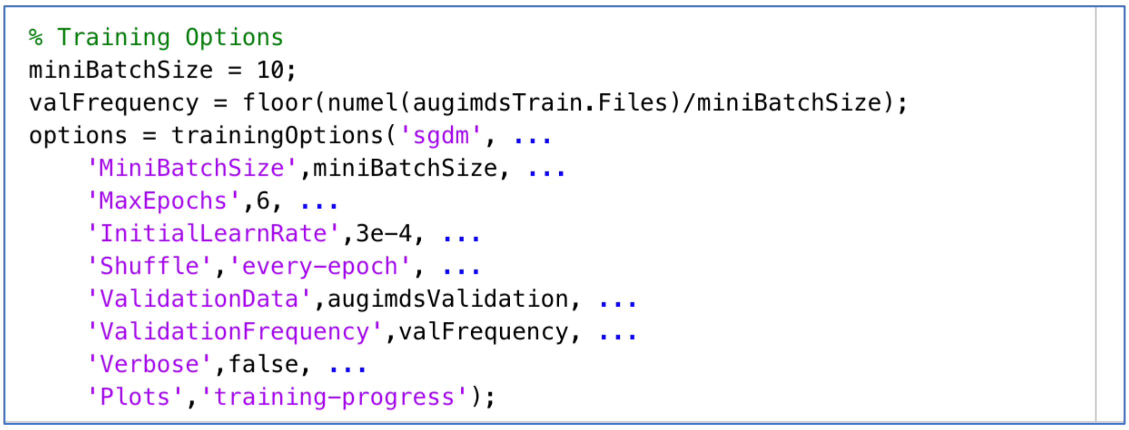
The value of the *InitialLearnRate* affects the balance between the time required for training and the optimisation of the training results; the default value for SGDM Optimizer is 0.01, but 0.0003 is used in this model. If the Leaning Rate is too small, learning takes longer, and if it is too large, the learning results may be suboptimal or diverge without convergence (MathWorks c - trainingOptions, 2022).

- 4-4-5. Shuffle

Choose whether to shuffle the training and test datasets before each epoch runs: *once*, *never* or *every-epoch*, with *once* shuffling only once before the training starts. *never* performs no shuffling, and *every-epoch* performs shuffling on the training and test data sets before each epoch is run. Depending on the number of images in the dataset and the value of MiniBatchSize, excess data may be discarded at the end of each epoch, and shuffling the dataset prevents the same data being discarded every epoch. In this model, every-epoch is used (MathWorks c - trainingOptions, 2022).

- 4-4-6. Validation Data and Frequency

The *ValidationData* option specifies the dataset used for validation during training. Also, the number of iterations at each epoch can be specified with the *ValidationFrequency* option (MathWorks c - trainingOptions, 2022). Here, the number of images present in the training dataset 30,400 is divided by the *MiniBatchSize* value 10 to ensure that there is no surplus data to be discarded at the end of each epoch. 3,040 are used in this model, so the overall number of iterations done during the training is 18,240 (3,040 iterations x 6 epochs).



```
% Training Options
miniBatchSize = 10;
valFrequency = floor(numel(augimdsTrain.Files)/miniBatchSize);
options = trainingOptions('sgdm', ...
    'MiniBatchSize',miniBatchSize, ...
    'MaxEpochs',6, ...
    'InitialLearnRate',3e-4, ...
    'Shuffle','every-epoch', ...
    'ValidationData',augimdsValidation, ...
    'ValidationFrequency',valFrequency, ...
    'Verbose',false, ...
    'Plots','training-progress');
```

Figure 11. A Part of the MATLAB Script for the training options (MathWorks b- Train Deep Learning Network to Classify New Images, 2022) - *Appendix01_TrainingProcedures.html*

Optionally, the training process and progress can be monitored by specifying the *Plots* and *training-progress* options in the *trainingOptions* function as Figure 11 shows above.

5. Results

5-1. 1st Experiment: 15 Classes - 100 Images Per Class

The 1st experiment is performed for 15 classes and 100 images per class to figure out a practical number of images per class. The accuracy of 92.0% is achieved with this experiment (Figure 12).

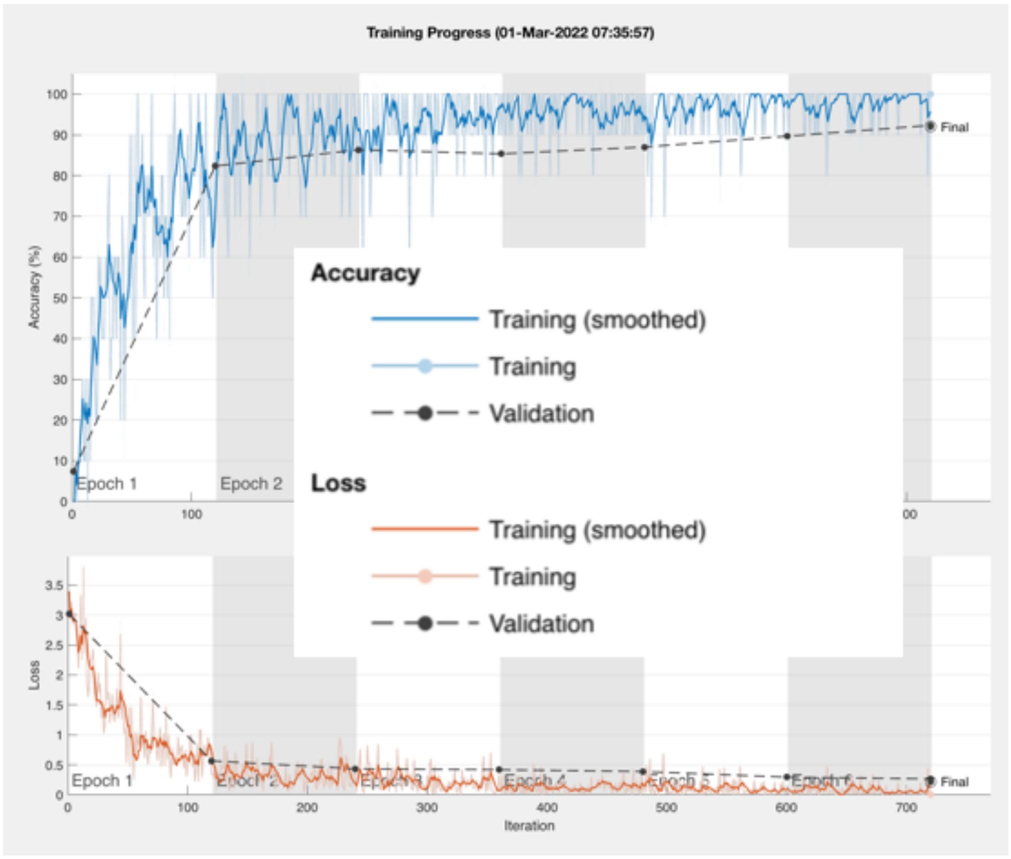
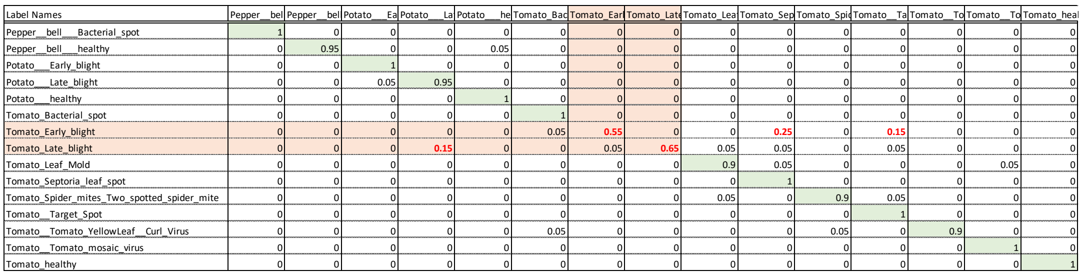


Figure 12. Training Progress Monitor and the Final Result for 1st Experiment

However, the accuracy for 2 classes, Tomato Early Blight and Tomato Late Blight show a quite low True Positive value, 55% for the Early Blight, and 65% for the Late Blight. As the confusion matrix shows below, it indicates that Tomato Early Blight has a 55% True Positive of being recognized as it is but has a 25% False Positive (being misrecognized) as Tomato Septoria Leaf Spot and a 15% False Positive as Tomato Target Spot. Tomato Late Blight is also correctly recognized with a True Positive of 65%, while it is misrecognized as Potato Late Blight with a probability of 15%.



Label Names	Pepper__bel	Pepper__bel	Potato___Ea	Potato___La	Potato___he	Tomato_Bac	Tomato_Earl	Tomato_Late	Tomato_Leaf	Tomato_Sep	Tomato_Spid	Tomato__Ta	Tomato__To	Tomato__To	Tomato_heal
Pepper__bell___Bacterial_spot	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Pepper__bell___healthy	0	0.95	0	0	0.05	0	0	0	0	0	0	0	0	0	0
Potato___Early_blight	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Potato___Late_blight	0	0	0.05	0.95	0	0	0	0	0	0	0	0	0	0	0
Potato___healthy	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Tomato_Bacterial_spot	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
Tomato_Early_blight	0	0	0	0	0	0.05	**0.55**	0	0	**0.25**	0	**0.15**	0	0	0
Tomato_Late_blight	0	0	0	**0.15**	0	0	0.05	**0.65**	0.05	0.05	0	0.05	0	0	0
Tomato_Leaf_Mold	0	0	0	0	0	0	0	0	0.9	0.05	0	0	0	0.05	0
Tomato_Septoria_leaf_spot	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
Tomato_Spider_mites_Two_spotted_spider_mite	0	0	0	0	0	0	0	0	0.05	0	0.9	0.05	0	0	0
Tomato__Target_Spot	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Tomato__Tomato_YellowLeaf__Curl_Virus	0	0	0	0	0	0.05	0	0	0	0	0.05	0	0.9	0	0
Tomato__Tomato_mosaic_virus	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Tomato_healthy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 4. Confusion Matrix for the 1st Experiment with 15 Classes and 100 images per Class

5-2. 2nd Experiment: 10 Tomato Classes - 2,127 Images Per Class

In the second experiment, the training only focuses on the 10 tomato classes, using 2,127 images per class, because in the previous experiment the misrecognition is concentrated on two classes of Tomato. The experiment achieved a high recognition rate overall as the confusion matrix shows below (Table 5), with an overall recognition rate of 99.27% (Figure 13).

	Tomato_Bac	Tomato_Earl	Tomato_Late	Tomato_Lea	Tomato_Sep	Tomato_Spid	Tomato__Ta	Tomato__To	Tomato__To	Tomato_heal
Tomato_Bacterial_spot	1	0	0	0	0	0	0	0	0	0
Tomato_Early_blight	0	0.975	0	0.01	0	0	0.005	0.01	0	0
Tomato_Late_blight	0	0	1	0	0	0	0	0	0	0
Tomato_Leaf_Mold	0	0	0	0.99	0	0	0.01	0	0	0
Tomato_Septoria_leaf_spot	0	0	0	0	0.99	0	0.01	0	0	0
Tomato_Spider_mites_Two_spotted_spider_mite	0	0	0	0	0	1	0	0	0	0
Tomato__Target_Spot	0	0	0	0	0	0	1	0	0	0
Tomato__Tomato_YellowLeaf__Curl_Virus	0	0	0.005	0	0	0	0	0.995	0	0
Tomato__Tomato_mosaic_virus	0	0	0	0	0	0.015	0	0	0.985	0
Tomato_healthy	0	0	0	0	0	0	0	0	0.005	0.995

Table 5. Confusion Matrix for 2nd Training with 10 Tomato classes

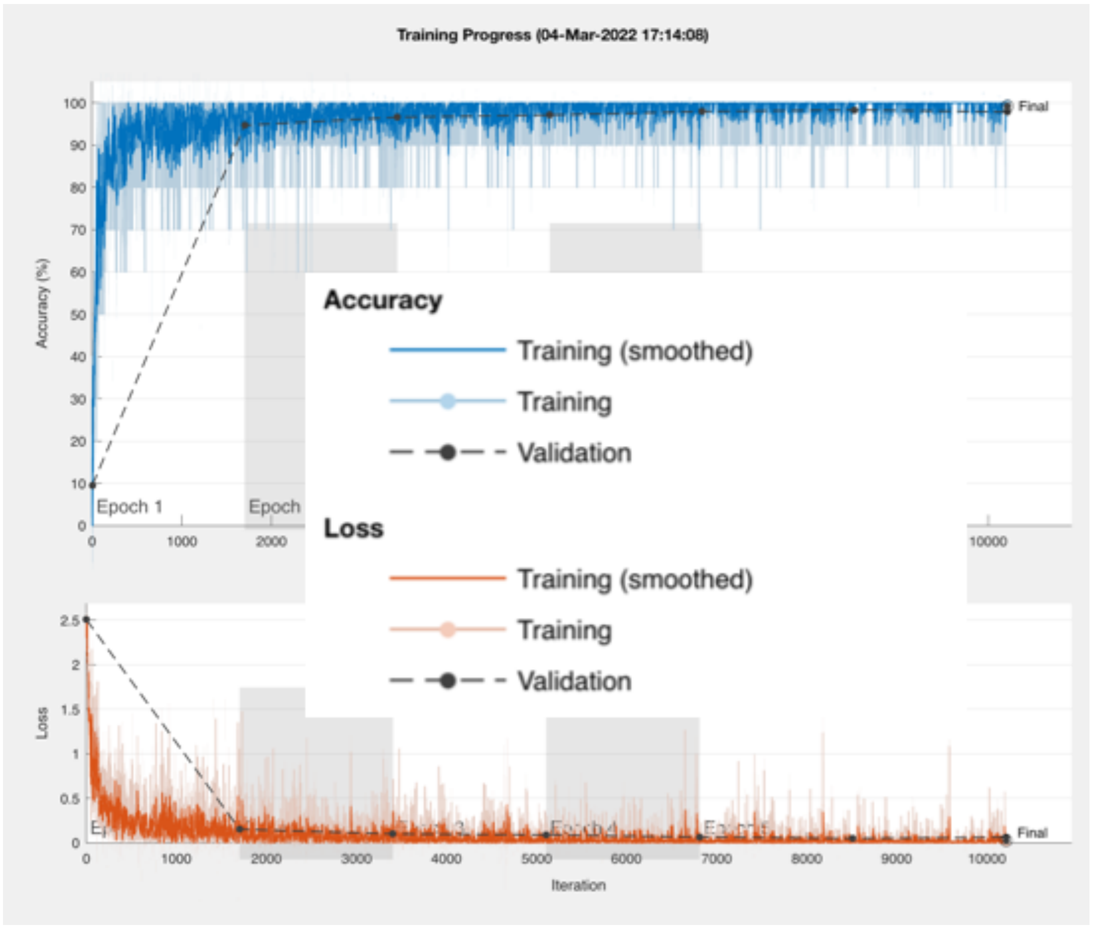
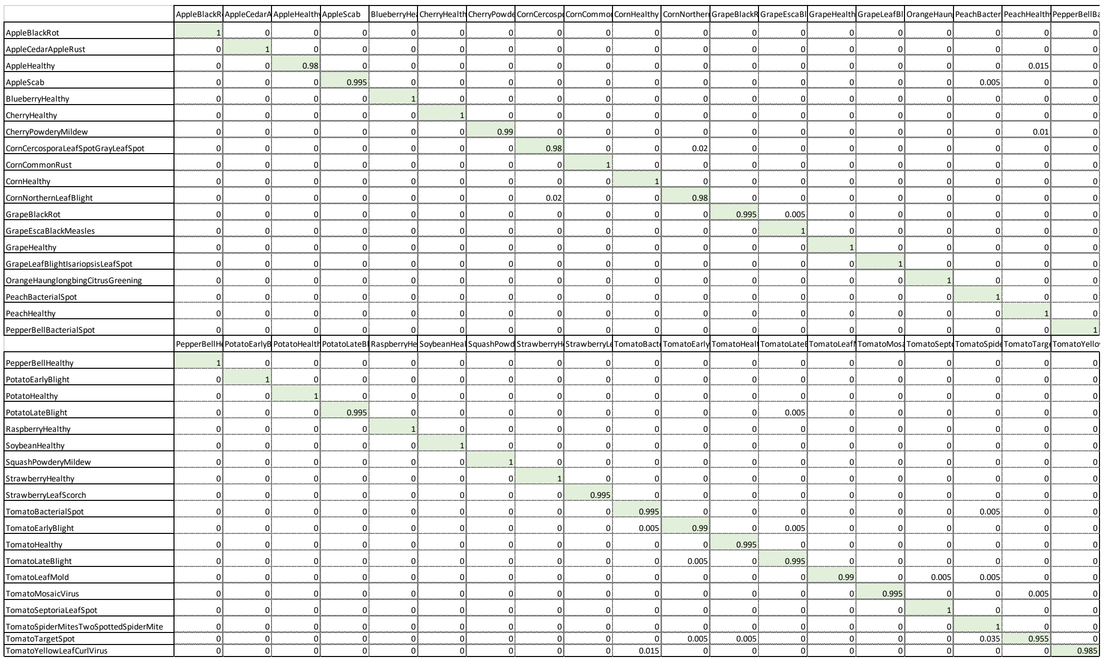


Figure 13. Training Progress Monitor and the Final Result for 2nd Experiment

5-3. 3rd Experiment: 38 Classes - 1,000 Images Per Class

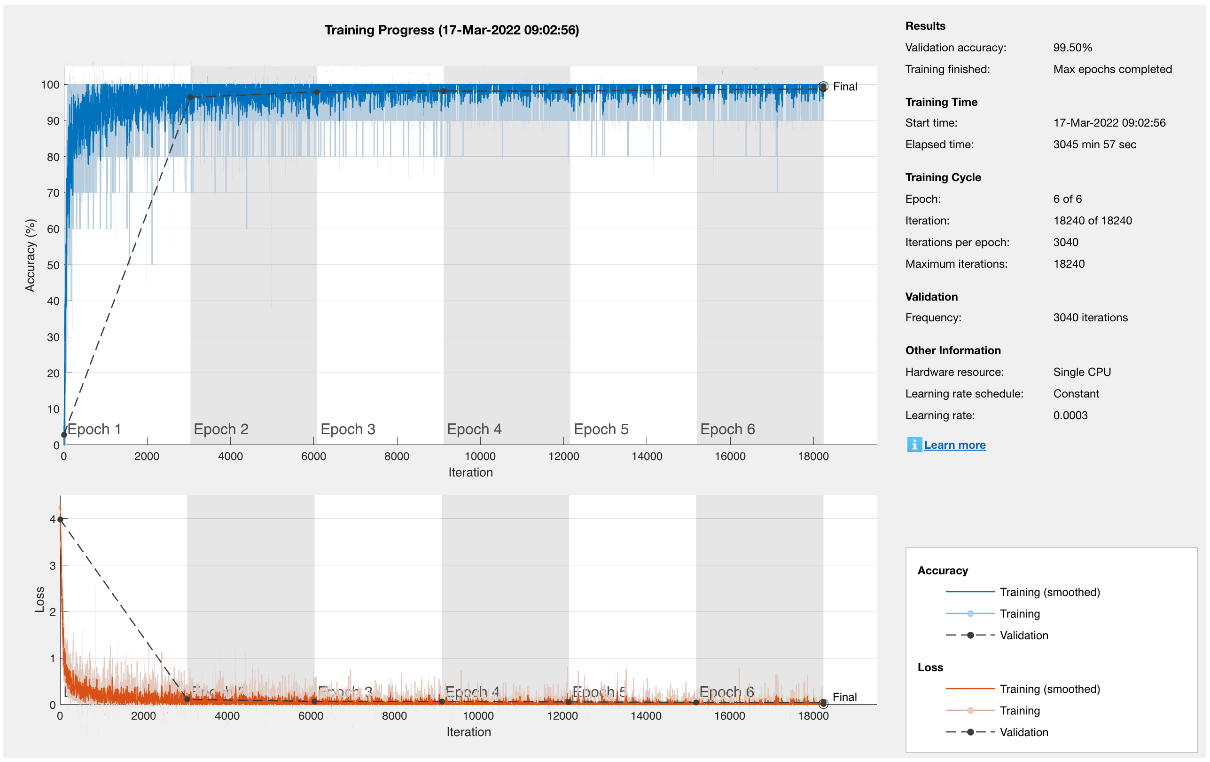
As the results of the previous two experiments indicates that a number of 1,000 or more images per class is quite effective in improving the recognition rate of several classes that are likely to be misrecognized, training is carried out with the number of images per class set at 1,000 for all 38 classes as a final experiment. An overall accuracy is 99.5% (Figure 14) and a 100% accuracy is achieved for 22 classes out of 38 classes (Table 6).



	AppleBlackR	AppleCedarA	AppleHealthy	AppleScab	BlueberryHe	CherryHealth	CherryPowde	CornCercosp	CornCommo	CornHealthy	CornNorthern	GrapeBlackR	GrapeEscaBl	GrapeHealth	GrapeLeafBl	OrangeHaun	PeachBacter	PeachHealthy	PepperBellBa
AppleBlackRot	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AppleCedarAppleRust	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AppleHealthy	0	0	0.98	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.015	0
AppleScab	0	0	0	0.995	0	0	0	0	0	0	0	0	0	0	0	0	0.005	0	0
BlueberryHealthy	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CherryHealthy	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
CherryPowderyMildew	0	0	0	0	0	0	0.99	0	0	0	0	0	0	0	0	0	0	0.01	0
CornCercosporaLeafSpotGrayLeafSpot	0	0	0	0	0	0	0	0.98	0	0	0.02	0	0	0	0	0	0	0	0
CornCommonRust	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
CornHealthy	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CornNorthernLeafBlight	0	0	0	0	0	0	0	0.02	0	0	0.98	0	0	0	0	0	0	0	0
GrapeBlackRot	0	0	0	0	0	0	0	0	0	0	0	0.995	0.005	0	0	0	0	0	0
GrapeEscaBlackMeasles	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
GrapeHealthy	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
GrapeLeafBlightIsariopsisLeafSpot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
OrangeHaunglongbingCitrusGreening	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
PeachBacterialSpot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
PeachHealthy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
PepperBellBacterialSpot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

	PepperBellHe	PotatoEarlyB	PotatoHealth	PotatoLateBl	RaspberryHe	SoybeanHeal	SquashPowd	StrawberryH	StrawberryLe	TomatoBact	TomatoEarly	TomatoHeal	TomatoLateB	TomatoLeafM	TomatoMosa	TomatoSept	TomatoSpide	TomatoTarge	TomatoYello
PepperBellHealthy	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PotatoEarlyBlight	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PotatoHealthy	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PotatoLateBlight	0	0	0	0.995	0	0	0	0	0	0	0	0	0.005	0	0	0	0	0	0
RaspberryHealthy	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SoybeanHealthy	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
SquashPowderyMildew	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
StrawberryHealthy	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
StrawberryLeafScorch	0	0	0	0	0	0	0	0	0.995	0	0	0	0	0	0	0	0	0	0
TomatoBacterialSpot	0	0	0	0	0	0	0	0	0	0.995	0	0	0	0	0	0	0.005	0	0
TomatoEarlyBlight	0	0	0	0	0	0	0	0	0	0.005	0.99	0	0.005	0	0	0	0	0	0
TomatoHealthy	0	0	0	0	0	0	0	0	0	0	0	0.995	0	0	0	0	0	0	0
TomatoLateBlight	0	0	0	0	0	0	0	0	0	0	0.005	0	0.995	0	0	0	0	0	0
TomatoLeafMold	0	0	0	0	0	0	0	0	0	0	0	0	0	0.99	0	0.005	0.005	0	0
TomatoMosaicVirus	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.995	0	0	0.005	0
TomatoSeptoriaLeafSpot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
TomatoSpiderMitesTwoSpottedSpiderMite	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
TomatoTargetSpot	0	0	0	0	0	0	0	0	0	0	0.005	0.005	0	0	0	0	0.035	0.955	0
TomatoYellowLeafCurlVirus	0	0	0	0	0	0	0	0	0	0.015	0	0	0	0	0	0	0	0	0.985

Table 6. Confusion Matrix for 38 Classes and 1,000 Images per Class (See *Appendix_03.xlsx: confMat_Percentages*)



Results

Validation accuracy:	99.50%
Training finished:	Max epochs completed

Training Time

Start time:	17-Mar-2022 09:02:56
Elapsed time:	3045 min 57 sec

Training Cycle

Epoch:	6 of 6
Iteration:	18240 of 18240
Iterations per epoch:	3040
Maximum iterations:	18240

Validation

Frequency:	3040 iterations

Other Information

Hardware resource:	Single CPU
Learning rate schedule:	Constant
Learning rate:	0.0003

Figure 14. Training Progress Monitor and the Final Result (See *Appendix_03.xlsx: MATLAB_TrainProgressResult*)

6. Discussions

6-1. Trained Model Implementation on a Mobile Phone

The trained model with a 99.5% accuracy has been implemented on iPhone 7 Plus. MATLAB exports a trained model as a single file (278 MB) to be utilized on another devices. Also, the MATLAB Mobile needs to be installed on a mobile phone. The exported trained model can be shared with the MATLAB Mobile on a mobile phone via a cloud storage, called MATLAB Drive. Although the trained model almost perfectly recognizes plant leaf images and their diseases on a computer screen by capturing them via a mobile phone's built-in camera, if there is a background or a reflection of the screen appeared in the captured image, the recognition accuracy would be getting lower. For a further study, it is necessary to develop a method for verifying accurate recognition rates using a mobile phone's camera and conduct experiments based on that method.

6-2. Accuracy Validation of the Trained Model on an Unknown Dataset

As mentioned in the previous section, the PlantVillage dataset is a high-quality dataset thoroughly reviewed by plant disease experts. However, all the images in the PlantVillage dataset show only one target leaf on a background of the same color tone. Therefore, models trained on such datasets with highly homogeneous images often show poor recognition rates for exceptional images - for example, images with other leaves in the background, or images where the area of the leaf is extremely small or too large for the area of the image. For example, 12 images are randomly selected from the PlantVillage dataset (Figure 15) and the PlantDoc dataset (Singh et al., 2020) (Figure 16). Both groups belong to the Corn Northern Leaf Blight class.

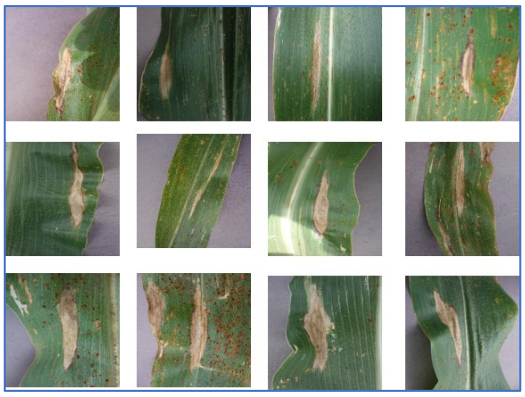


Figure 15. PlantVillage dataset

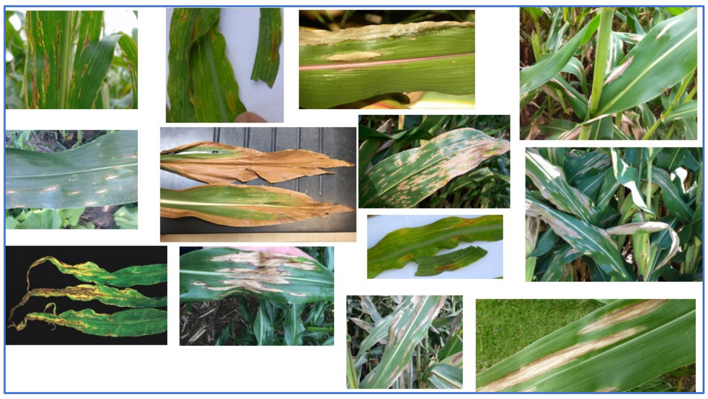


Figure 16. PlantDoc dataset

The PlantDoc dataset uses a variety of images, including images with deliberately different aspect ratios and images with other leaves, whereas the PlantVillage dataset is uniform, down to the colour of the background and the ratio of leaf area to image area. The following are the results of experiments with models trained on the PlantVillage dataset, with the PlantDoc dataset as the validation image. The experiments are limited to the nine classes present in both PlantVillage and PlantDoc datasets, resulting in a recognition accuracy of only 83.7%. Here, the CornHealthy class is perfectly recognized as it is, but Potato and Tomato classes show a relatively low True Positive score.

Labels	True Positive
CornCercosporaLeafSpotGrayLeafSpot	88.66%
CornCommonRust	92.50%
CornHealthy	100.00%
CornNorthernLeafBlight	85.86%
PotatoHealthy	59.78%
PotatoLateBlight	56.72%
TomatoHealthy	52.44%
TomatoMosaicVirus	57.27%
TomatoYellowLeafCurlVirus	71.17%

Table 7. Extra Experiment: Validating the PlantDoc dataset with the PlantVillage dataset.

Thus, a considerable loss of recognition accuracy occurs when the quality of the dataset used for training differs significantly from that of the dataset used for validation. Future research is expected to include training on datasets with mixed data of different quality and the development of methods for verifying biases caused by this.

7. Conclusion

In this study, a MATLAB platform is used to develop an image recognition training model using the PlantVillage dataset. The final recognition accuracy of the model achieves 99.5% for 38 Plant-Disease pairs. The convolutional neural network used is ResNet50, which is the dominant base model in recent years, and the shortcut structure of ResNet50 is shown to play an important role in improving the trade-off between recognition accuracy and the number of parameters. The model has been implemented on a Mobile Phone and appears to show high recognition accuracy, but a specific validation method needs to be developed for the mobile phone's camera, in the future. On the other hand, due to the homogeneity of the PlantVillage dataset used for training, it shows low recognition accuracy for datasets consisting of a variety of miscellaneous data, and future work on mixed dataset methods and other data augmentation techniques is expected to improve this issue.

A Total Word Count: 5,211

References

A. KP & J. Anitha 2021, "Plant disease classification using deep learning", - *2021 3rd International Conference on Signal Processing and Communication (ICPSC)*, pp. 407.

Atila, Ü, Uçar, M., Akyol, K. & Uçar, E. 2021, "Plant leaf disease classification using EfficientNet deep learning model", *Ecological Informatics,* vol. 61, pp. 101182.

Graaf, N.A. van der (FAO, Rome (Italy). Plant Protection Service) *The international plant protection convention,* Wageningen Pers, Wageningen (Netherlands).

He, K., Zhang, X., Ren, S. & Sun, J. 2015, *Deep Residual Learning for Image Recognition*, arXiv.

Hughes, J.W. 2017, "Reviewing Mobile Health Applications", *Health affairs Web exclusive,* vol. 36, no. 2, pp. 383-384.

Indolia, S., Goswami, A.K., Mishra, S.P. & Asopa, P. 2018, "Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach", *Procedia Computer Science,* vol. 132, pp. 679-688.

Ioffe, S. & Szegedy, C. 2015, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, arXiv.

J, A.P. & GOPAL, G. 2019, 18 Apr-last update*, Data for: Identification of Plant Leaf Diseases Using a 9-layer Deep Convolutional Neural Network*. Available: doi: 10.17632/tywbtsjrjv.1.

Liu, L. & Luo, X. 2020, *A New Accelerated Stochastic Gradient Method with Momentum*, arXiv.

Masters, D. & Luschi, C. 2018, *Revisiting Small Batch Training for Deep Neural Networks*, arXiv.

MathWorks a, , *imageDataAugmenter*.
Available: https://uk.mathworks.com/help/deeplearning/ref/imagedataaugmenter.html.

MathWorks b, , *Train Deep Learning Network to Classify New Images*.
Available: https://uk.mathworks.com/help/deeplearning/ug/train-deep-learning-network-to-classify-new-images.html.

MathWorks c, , *trainingOptions*.
Available: https://uk.mathworks.com/help/deeplearning/ref/trainingoptions.html#bu80qkw-3_head.

Mohanty, S.P., Hughes, D.P. & Salathé, M. 2016, "Using Deep Learning for Image-Based Plant Disease Detection", *Frontiers in Plant Science,* vol. 7, pp. 1419.

N B, H. 2019, Dec 10,-last update*, Confusion Matrix, Accuracy, Precision, Recall, F1 Score*.
Available: https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd [2021, Nov. 17,].

Oztel, I., Yolcu, G. & Oz, C. 2019, "Performance Comparison of Transfer Learning and Training from Scratch Approaches for Deep Facial Expression Recognition", , pp. 1.

Saha, S. 2018, Dec 15,-last update*, A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 [2021, Nov. 17,].

Shi, C. 2020, "Convolutional Neural Networks to Detect Plant Disease in Common Food Crops", .

Shorten, C. & Khoshgoftaar, T.M. 2019, "A survey on Image Data Augmentation for Deep Learning", *Journal of Big Data,* vol. 6, no. 1, pp. 60.

Singh, D., Jain, N., Jain, P., Kayal, P., Kumawat, S. & Batra, N. 2020, "PlantDoc: A Dataset for Visual Plant Disease Detection", *Proceedings of the 7th ACM IKDD CoDS and 25th COMAD*Association for Computing Machinery, New York, NY, USA.

Venkataramanan, A. & Agarwal, P. 2019, *Plant Disease Detection and Classification Using Deep Neural Networks,* .

Yamashita, R., Nishio, M., Do, R.K.G. & Togashi, K. 2018, "Convolutional neural networks: an overview and application in radiology", *Insights into Imaging,* vol. 9, no. 4, pp. 611-629.

Zhai, J., Shen, W., Singh, I., Wanyama, T. & Gao, Z. 2020, "A Review of the Evolution of Deep Learning Architectures and Comparison of their Performances for Histopathologic Cancer Detection", *Procedia Manufacturing,* vol. 46, pp. 683-689.

Zhao, S., Peng, Y., Liu, J. & Wu, S. 2021, *Tomato Leaf Disease Diagnosis Based on Improved Convolution Neural Network by Attention Module*.