



Intelligent System for Diagnosis of Herbs Disease Using Deep Learning

Rabindra Kumar Singh¹(✉), B. V. A. N. S. Prabhakar Rao¹, M. Sivabalakrishnan¹,
and M. Shiny Pidugu²

¹ School of Computer Science and Engineering, Vellore Institute of Technology, Chennai, India
mail2rksingh@gmail.com

² Cognizant Technology Solutions, Hyderabad, India

Abstract. The agricultural production of the country is severely affected when herbs and crops are attacked by disease. The usual methods adopted by farmers or even agriculture experts are to make several observation to the herbs with naked eye in order to identifying and detecting the disease, and make an approximate decision for herbs treatment. This method happens to be always a time consuming and inaccurate that leads to be expensive. Now we have advanced technology such as automatic detection using deep learning, which produce results accurate and fast. This paper aims to present an approach to develop a model to detect herbs disease progress, depending on the leaf images classification, using deep convolutional network. With the advent of computer vision, it has been noticed that the precision herbal protection were improvised and therefore the computer vision applications have gained more popularity even in precision agriculture field. Here, Novel training techniques are proposed which actually enables faster and less complex implementations in order to herb diseases detection. All the necessary key steps required in order to implement disease detection model has been described in this paper. These key steps ranges from collection of images to building database, evaluated by experts in the field of agriculture with the assistance of deep CNN training are described. The described technique is nothing but intelligent system development in order to classifying herb infections by means of deep convolutional neural networks. This model were trained and tweaked in order to suit the database of herb's leaves images, that were congregated self-sufficient for diverse plant diseases. The growth and novelty of this developed model dwell in its simplicity. Healthy leaves and background images match other classes, allowing the model to use CNN to distinguish between diseased leaves and healthy leaves.

Keywords: Herb leaf disease detection · Convolutional Neural Network · Deep learning · Computer vision · ResNet

1 Introduction

Viruses, bacteria, oomycetes, fungus, nematodes, and parasitic plants are among the pathogens that cause herb diseases. Plant diseases can be an issue for any plant system,

© IFIP International Federation for Information Processing 2022

Published by Springer Nature Switzerland AG 2022

E. J. Neuhold et al. (Eds.): ICCSP 2022, IFIP AICT 651, pp. 98–115, 2022.

https://doi.org/10.1007/978-3-031-11633-9_9

but those affecting agricultural systems can have a particularly detrimental impact on human livelihoods and health. For example, the late blight of potatoes, a disease caused by *Phytophthora infestans* – a fungus-like oomycete pathogen, was first discovered in the early 1840s in Ireland. After the epidemic outbreak, approximately one million people lost their life from starvation, and approximately two 2 million immigrated to other countries to escape starvation. This example points to the tremendous human impacts that major agricultural disease outbreaks can cause. More common are less massive outbreaks that result in loss of yield, detrimental effects on economies, and particularly devastating effects on small farm holders or subsistence farmers.

Most traditional strategies in herb disease monitoring and tracking depend on visual inspection. In some cases, disease detection is aided by microscopic observation at the molecular level. These approaches tend to be accurate but are limited in the spatial extent to which they can be applied and can be biased by the previous experiences of the person making the visual inspections. These methods are likewise costly and not practical for the broad agricultural community. What is required is an accurate and affordable plant disease diagnosis system that will help farmers, especially smallholder farmers, to detect herb diseases at early stages, across extensive fields, and with the ability to be deployed many times throughout a growing season.

A major consideration for agriculture in the coming decades is the need to provide an early warning and forecast for effective prevention and control of herb disease. A primary factor in this effort could be herb disease detection, which would prevent significant economic losses and enhance smallholder agriculture's resilience. The development of accurate and affordable herb disease diagnosis systems is, therefore, an urgent priority.

Since the victory of the ImageNet Large Scale Visual Recognition, continuous advancements in deep learning have been accomplished, fueled by current breakthroughs in computer vision and machine learning systems. For some herb disease recognition challenges, it has been shown that the deep learning methodology can generate more accurate results than the classical approach. These initial studies merit further research and extension to different diseases, cropping systems, and geographic contexts. As well, much less progress has been made in the more complicated problem of diagnosing disease severity level and in differentiating the type of disease that is occurring – critical for adequately managing or mitigating a disease outbreak situation.

Machine learning algorithms are used in research to detect plant disease. Traditional machine learning techniques were utilised by some, while deep learning models were used by others. Support Vector Machine approach was proposed to detect and classify plant diseases in [3], the dataset used was small and thus the accuracy achieved was average. The authors in [1] proposed a CNN based Learning Vector Quantization algorithm, in which they classified the tomato related diseases. A dataset of 500 images that were divided into training and testing images of 400 and 100 images respectively. Totally 5 classes including one healthy class were available in classification result.

Three matrices of R, G, and B channels were used as inputs to the CNN model, and the output was fed to a neural network known as Learning Vector Quantization (LVQ). Due to its great efficacy in image processing, deep learning models, particularly CNN, have been frequently deployed. In [2] the authors compared different classifiers such as logistic regression, KNN, SVM, and CNN to identify and classify plant leaf

diseases. The dataset comprised of about 20,000 images and total of 19 classes. They applied different classifiers separately and recorded the accuracy obtained in each case. The CNN classifier performed the best and the other classifiers gave below average performance. Thus verifying the power of CNN in image classification. In [4] the authors used deep learning approach to classify banana leaf disease. They classified the disease in two classes: banana sigatoka and banana speckle. The Computer Vision technology is not just limited to leaf disease detection, it can also diagnose fruits as described in [5] where the authors used grab cut segmentation to segment the image of pomegranate and identify the diseased part of the fruit. There are many pre-trained models are available in which better performance achieved even with less data. In literature many methods have been used as a pre-trained model. One such methodology is cited in [7] in which a comparison between SqueezeNet and AlexNet has been done. SqueezeNet was found to have an accuracy of 94.3%, whereas AlexNet had 95.6%.

Deep convolutional neural network have headed to numerous discoveries in the field of image categorization. The network depth is important and nearly all prominent picture classification methods use models that are extremely deep. The performance of the Residual Network (ResNet50) for identifying and classifying plant diseases is addressed in this research. The use of Residual Network is because of a great deal of success in computer vision field. Working with ResNet is primarily motivated by the need to provide alternatives to traditional connections while also generating residual connections. The use of ResNet to solve the problem at hand has been well known, as studied in previous research. Although this study reported superhuman findings, one of the primary issues addressed in this study is that the model can be fooled by the resemblance between different leaf diseases, causing it to make inaccurate predictions.

ResNet50 [18], which has 50 layers, is used to see if Residual Networks produce superior plant disease classification results. ResNet50 was successfully applied on a dataset including 20,000 photos for the trials.

The following are the research contributions of this paper:

- To detect diseases in herb leaf photos.
- To classify the detected disease into different classes.
- To demonstrate that residual networks (ResNet) may be used to classify plant diseases.
- Understanding the role of ResNets in increasing disease detection and classification scores.

The remaining part of the paper is sub-divided in the following sections: Section 2 covers the basic method of disease identification and classification. In Sect. 3, CNN is discussed. The ResNet, which is the suggested model, is described in length in Sect. 4. The outcomes of the experiments are presented in Sect. 5. Section 6 is devoted to the paper's conclusion. Section 7 discusses potential future projects.

2 Classification and Detection of Plant Diseases

It comprises of the following steps:

2.1 Acquisition of Image

2.2 Image Pre-processing

2.3 Feature Extraction

2.4 Classification

2.1 Acquisition of Image

The dataset consists of 20000 photos separated into 15 different classes and covers three crops. This information comes from the Plant Village dataset (Figs. 1 and 2).

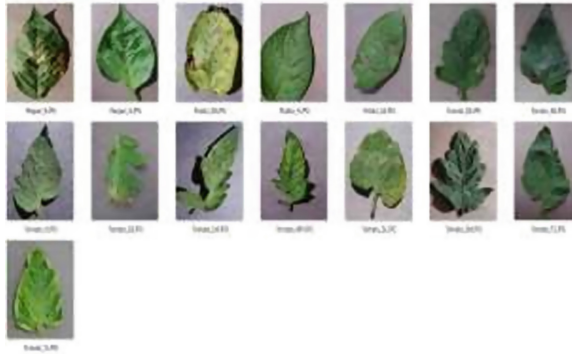


Fig. 1. Leaf image samples for PlantVillage dataset

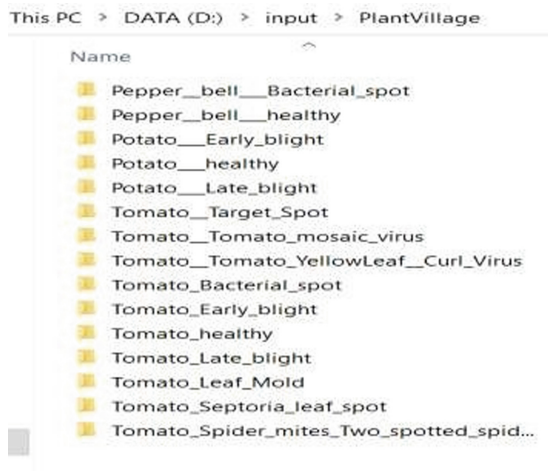


Fig. 2. Category path for the PlantVillage dataset

2.2 Image Pre-processing

It is carried out in order to convert the data into a format that will allow the feature extraction method and future steps to function properly. This process involves data augmentation and normalisation.

1. **Data Augmentation:** It is a common regularization system, which delivers a concrete solution to the problem of overfitting. In this system image is rotated to 90° and the probability of the image getting rotated is 0.75. This used to improve the learning speed of the model.
2. **Data Normalization:** Normalization is performed so that all pixel values have the same mean and standard deviation. This speeds up the learning of the model.

2.3 Feature Extraction

Relevant features are extracted first to solve the classification problem at hand. Color, texture and shape of images are known as features. Structures that discover diseases with the help of leaf images emphasis better on the texture feature. Examples of methods that can be used include gray-level co-occurrence matrices, autocorrelation, Gabor transforms, and 2D Gabor functions. Grey level co-occurrence matrix (GLCM) is a statistical technique that describes the image texture by calculating how often pairs of pixels with certain values and in a definite spatial relationship appear in an image.

Autocorrelation represents the degree of correlation over a continuous period between a particular time series and previous versions. The Gabor transform is a type of short-time Fourier transform used to calculate the phase component of the local part of the signal over time and the appearance of the sine wave. The Gabor function can model simple neurons in the visual cortex of the mammalian brain. The 2D Gabor function is used to simulate the space-based total properties of simple cells (receptive fields) in the visual cortex. In addition to high accuracy, automatic feature extraction has proven to be one of the greatest advantages of using deep learning models. Since the deep learning model ResNet50 is used with classification, it also handles automatic feature extraction. Therefore, the proposed approach does not require the use of separate feature extraction methods.

2.4 Classification

There are more than one picks to be had for classification. Some of the classifiers that may be used on this step are:

Logistic regression, radial basis functions, linear vector quantization, ANN, classification tree, support vector machine, CNN, KNN, etc. ResNet50-the proposed method uses the CNN architecture for classification purposes.

3 Convolutional Neural Network

The way human brain works inspired the development of a classification algorithms that belong under the deep learning umbrella. It entails teaching ANN how to make

predictions. ANN stands for “a network of neurons structured in a multilayer form.” The output of previous layer develops the input for the following layer. Because deep learning models extract features automatically during training, there is no need for a separate approach for the extraction feature step of the fundamental process outlined previously. The first layer of ANN acquire low level features (such as edges), and as they get deeper, they learn higher-level characteristics (such as whole objects).

CNN is a form of ANN, which is specifically designed for image processing. CNN has been proved in studies to be capable of providing excellent accuracy in image processing jobs. Because single neuron in a layer of an ANN receives input from previous layer’s neurons, learning a large number of parameters for image-based tasks becomes a difficulty. In compared to ANN, CNN is preferred for image-related jobs since it involves less parameters. This reduction can be attributed to parameter sharing. To extract all the activations in the output volume from an input volume of activations, similar parameters (called filters in CNN nomenclature) are utilised. Parameter sharing is the term for this. A Convolutional Network’s purpose is to reduce the size of an image without losing important elements that aid in issue solving. Figure 3 shows the general architecture of CNN. There are four sorts of layers in a CNN architecture:

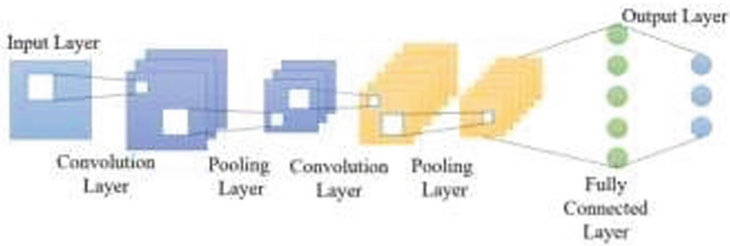


Fig. 3. CNN architecture

3.1 Convolutional Layer

Convolutional layer was given the term CNN. The image size is lowered by using a number of convolutional processes. A filter is set in the upper left corner of the image and then moved along the width of the image by a stride value towards the right. After covering the entire width, the filter bounces down by the same stride value and starts over from the left to complete the coverage. This technique is repeated until the entire image has been explored. In a single step, the sum of product of comparable values in the coinciding region of the image and filter is evaluated. From the input matrix, a new matrix (or volume) is created (or volume). Figure 4 demonstrates convolution applied to a 5×5 image in a convolutional layer with a 3×3 filter.

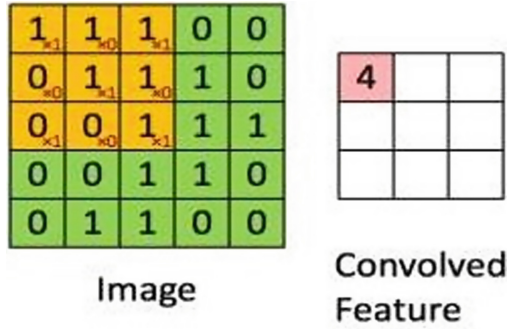


Fig. 4. Convolutional operator applied to 5 × 5 input and 3 × 3 filter

3.2 Pooling Layer

Functions this layer include shrinking image and extracting most important elements. A filter is positioned as it was done in convolutional layer and moved to the end. A function is applied in this layer as single step. This function can be either a max-function that determines the maximum of all the values in the overlapping section of the input image and the filter (called Max-pooling) or an average function that determines the average of all the values in the overlapped section of the input picture and the filter (called avg-pooling). The most common filter size and stride is 2. Avg-pooling is preferable to max-pooling. Figure 5 shows max-pooling.



Fig. 5. .

3.3 Fully Connected Layer (FC)

A matrix is the outcome of a series of convolutional + pooling layers. This is flattened, and then an ANN-like series of completely connected layers is used. In a layer, a single neuron receives information from all neurons in the previous layer. FC Layers are only utilised after a series of convolutional plus pooling layers has decreased the size of the image to the point where the fully connected layers don't have a huge number of parameters to learn.

3.4 Activation Layer

Each element of the input matrix is given an activation function. As a result, the input and output dimensions for this layer are the same. Linear hypothesis functions can only be approached with the help of linear activation functions. The usage of non-linear activation functions is frequent. For complicated situations, there is usually a non-linear relationship between input and output. The ReLU function is commonly utilized because it allows for faster learning.

4 Proposed Model: ResNet Networks

Previous research has revealed the critical role of network depth. The accuracy of a neural network should theoretically improve as more layers are added. In truth, it proves to be a misunderstanding. When the network's depth is increased, the accuracy tends to become saturated and subsequently decline quickly. This is referred to as the degrading issue. Surprisingly, overfitting is not the root of the problem.

This well-known degradation problem is caused by the phenomenon of vanishing/exploding gradients in deep neural networks. Due to repeated multiplication during the backpropagation stage, the gradients in the vanishing gradient problem become endlessly small, resulting in negligible parameter updates. Exploding gradients is a problem when gradients build up and cause unusually high parameter updates during training, preventing the model from learning from the data. This was handled before the discovery of residual networks by the use of normalised initialization and intermediary normalisation layers.

Residual network (ResNet) is a CNN design with a residual block as its main building block. A residual block is shown in Fig. 6.

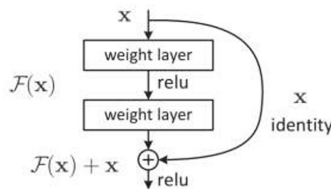


Fig. 6. Residual block

By utilizing skip connections, a residual block tackles the degradation issue. Shortcut connections are those that leap one or more levels (also known as skip connections). If the ordered connection's coefficient converges to zero during the training period, the residual shortcut ensures network integrity. It has been proposed that the layers learn the residual function $F(x)$ instead of the hypothesis function $H(x) = F(x) + x$. This is owing to the ease with which the residual function can be optimised. To demonstrate that a deeper network does not have a higher training error than its shallow version, the researchers created a deep network by concatenating a residual block at the end of a shallow network and then demonstrated that the residual block functions as an identity mapping.

5 Experiments and Analysis

5.1 Training

The dataset used in this thesis was divided into training and validation datasets by 80% and 20%, respectively, as it has shown the best performance in Table 1. The validation data is used for evaluating the performance after each epoch and did not involve in the training process. Before the training, each pixel of images was firstly normalized dividing by 224, and For all models, the input size was fixed at 224×224 by default, and the batch size was set to 32, which is the highest accuracy performance based on the experimental result as shown in Table 2. In order to expedite the training process, the CNN models ResNet50 and VGG16 were utilised in conjunction with the transfer learning strategy, and these pre-trained models were previously trained on the ImageNet dataset.

To shorten the training time and lower the computation cost, all layers in the pre-trained model were frozen during transfer learning. The last fully connected layer of all pre-trained models was taken for the fined-tuning purpose, and a dense layer with 1024 neurons was appended before transmitting to the 15 neurons in the output layer.

In the last layer, Softmax was employed as the activation function, and the categorical cross entropy was used for computing the loss function, which showed no significant difference to sparse categorical cross-entropy demonstrated in Table 3. SGD was utilised as the optimization method throughout the training, with the learning rate and momentum set at 0.001 and 0.9, respectively.

5.2 Evaluation Metrics

Choosing appropriate evaluation metrics is as important as choosing the learning algorithm, especially when the dataset is unbalanced. In this section, we will explain some differences in the statistics that we collected and shed some light on why they can be important to the study. The statistics that were collected after each epoch of training include time per epoch, cross-entropy loss, accuracy, precision, recall, and Area under the Curve (AUC).

- 1) *Accuracy*: Accuracy is the most widely used and the simplest indicator for evaluating the number of correct predictions that had been made over the data, but we must ask ourselves several questions: Does accuracy always perform well in evaluating the performance of a model? and when might accuracy not be an appropriate metric to use? In short speaking, accuracy might not be a good indicator for an unbalanced dataset. For example, if we have an unbalanced dataset that contains 999 negative samples and 1 positive sample in the dataset, as a result, the accuracy for this model will be 99.9%.

However, the result is not fairly evaluated for the positive categories whereas the dataset only contains one positive sample, and the result can be misleading and can be costly. The percentage of correct predictions is the definition of accuracy as defined below:

$$\text{Accuracy} = \frac{\# \text{ of corrected prediction}}{\text{Total Samples}}$$

As in the previous example, accuracy cannot explain well for an unbalanced dataset, and other evaluation metrics should take into consideration, such as precision, recall or F1 score.

- 2) *Precision*: Precision is defined as the fraction of true positives examples among the total retrieved positive instances (a true positive(TP) + false positive(FP)). The range of precision is between 0 and 1, and can be calculated as below equation:

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{TP}{\text{Total Predicted Positive}}$$

As the formula suggested, the precision metric evaluates the number of true positive prediction within total predicted positive samples and only need to take into consideration if false positive (or Type I error rate) is significant to the study [28].

- 3) *Recall*: The recall is defined as the proposition of TP examples amount the total true positive (TP + false negative(FN)) and is also known as sensitivity or True Positive Rate (TPR). The range of recall is also between 0 and 1, and can be calculated as following equation:

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{TP}{\text{Total Actual Positive}}$$

As the formula suggested, the recall should be taken into consideration if the FN is significant to the study.

- 4) *F1 Score*: A weighted harmonic mean of recall and precision is used to get the F1 score and had been widely used for many machine learning algorithms. The equation is defined below:

$$F1 = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

- 5) *AUC*: The area under the curve(AUC) is a popular metric for evaluating how good our model performed in distinguishing between classes, and sometimes it is also known as (Area Under the Receiver Operating Characteristics) AUROC. Each recall and precision will have a corresponding instance of AUC, The AUC has values ranged between 0 and 1. The higher the AUC, the better the separation capacity of the model in distinguishing between classes. Ideally, if AUC is 1, the model can perfectly distinguish between positive and negative samples; on the other hand, if the AUC is 0, the model has no separability. Overall, accuracy is commonly used for the balanced dataset. For an unbalanced dataset, if the false positive is more significant than the false negative, we should use recall as an indicator, otherwise, precision should be used. In the case that both false positives and false negatives are important to us, we should consider the F1 score for the evaluation purpose, and the AUC curve is a valuable tool to visualize the model's separability on a skewed dataset [27].
- 6) *Intersection over union*: IoU is an accuracy evaluation metric that is commonly used in image object detection tasks. Unlike the traditional classification or object recognition tasks, the accuracy was measured based on the number of predicted results matched to the ground-truth labels. It is hard to expect two bounding boxes will align exactly in the same coordination x-y axis. The IoU calculated the accuracy by dividing the anticipated bounding box by the ground-truth bounding box. Specifically, it

measures the proportion of intersected area between two bounding boxes over the area of their union. The more overlapping between two bounding boxes indicates a better predicted result. This can be expressed as the equation below:

$$\text{IoU} = \frac{\text{area}(\text{Bp} \cap \text{Bgt})}{\text{area}(\text{Bp} \cup \text{Bgt})}$$

where Bgt is the ground truth bounding box and Bp is the predicted bounding box for detection.

5.3 TRAIN/TEST Dataset

In deep learning development, we could not initially come up with the optimal configuration parameters. Therefore, applying deep learning is a very repetitive process where we must follow the development circle from generating the idea, training the model, evaluating the performance, and over and over again. Therefore, one thing that can expedite the speed of the development process is to improve the efficiency in going through that cycle and setting up the appropriate partition ratio of the dataset in terms of training, validation, and testing. The development dataset is also known as a held-out validation set or development set, but for brevity, we commonly just call it the “dev set” or “valid set”. A valid set is what we use to compare different models’ performance and see which one performs the best during the development stage. Once we have the final model that we want to evaluate, the test set will be used to produce an unbiased estimation of the unseen data. In many applications of the machine learning algorithm, we often start with the 60/20/20 train-dev-test split, or 70/30 train-test split (if you do not need an explicit dev set).

However, in the modern deep learning application, or the big data era, we might have a million examples in total, and taking up 20% of the dataset for testing can be excessive. Since the goal of the dev set is for comparing the performance of a different algorithm, sometimes even just 1% of the dev set and 1% of the test set can suffice that purpose.

5.4 Division of Training and Validation Sets

The size of training data is an important factor in deciding the quality of generalization ability of a model. In this paper, I conducted 12 sets of experiments on 2 DNN architectures (ResNet50 and VGG16) and 6 various ratios of the train validation split to show the accuracy after 3 epochs of training. The result in Table 1 show that there exists a clear trend that, as the size of the training dataset increased, the training accuracy for the model tends to increase. However, we also noticed that the performance of ResNet50 does get degraded as the train to valid set ratio exceeds 80/20.

Therefore, it was concluded that the 80/20 train-valid set split ratio is a good balance point to have a reliable model performance.

5.5 Batch Size

The traditional SGD takes one example at a time and was trained sequentially. As the dataset grows and the availability of GPU’s resource, the mini-batch SGD become more

Table 1. Accuracy score across various experiment configurations after 3 epochs of training

Train-Validation splitANN model	ResNet50	VGG16
Train 20%, Valid 80%	0.8856	0.7456
Train 40%, Valid 60%	0.8974	0.7745
Train 50%, Valid 50%	0.9147	0.7954
Train 60%, Valid 40%	0.9248	0.8147
Train 80%, Valid 20%	0.9326	0.8756
Train 90%, Valid 10%	0.9321	0.8614

and more popular in training large scale dataset in a parallel and distributed fashion, where a subset of data is used to perform forward, and backward-propagation independently amount processors and synchronize the update through the global all reduce operation(a type of MPI communication). However, there is trade-off between the batch size, training speed, and the convergence rate.

On the one extreme case, if the mini-batch size is too small (e.g., use single sample for weight update), we can have faster convergence to an approximate solution, but we might face the problem of I/O communication overhead and the model is not guaranteed to converge to the global optima; on the other hand, if the mini-batch size is too large (e.g., use the entire dataset to perform weight update), the model can guarantee to converge to the global optima of the objective function. However, the convergence rate can be reduced dramatically and can take much longer for a single weight update. To determine the best batch size to use for this thesis, I conducted seven set of experiments for discovering the best mini-batch size to use. All the experiments were trained on ResNet50 neural network with 80/20 train-validation split ratio, and the result is listed in Table 2:

Table 2. Training accuracy on various batch size after first epoch

Batch size	Valid accuracy	Train accuracy	Time/Epoch
32	0.8521	0.9144	587
64	0.8112	0.9127	612
128	0.7645	0.8196	619
256	0.7154	0.7988	634
512	0.6455	0.7122	654
1024	0.4789	0.6523	694
2048	0.3312	0.4345	688

From the result, it was observed that the training accuracy increased as we reduce the mini-batch size, but there is no significant impact on the training time per epoch.

As suggested in a paper, the mini-batch size b should not be larger than the T/b iteration, where T represents the total number of steps for one epoch. Therefore, 32 can be an appropriate balance-point for this study, and all the following experiments were performed with a batch size of 32.

5.6 Loss Function

As there are 15 classes in the dataset, two common categorical loss functions were selected for determining the top loss function in this study: Categorical Cross-Entropy (CCE) and Sparse Categorical Cross-Entropy (SCCE). CCE is a loss function that most used in multi-class classification, and the formula has defined below:

$$\text{generalized cross-entropy} = \sum_i^c \log(f_i(x_i; \theta))$$

where y_i and $f_i(x_i, \theta)$ are the one-hot encoded label and DNN scores for each *class* in C (as the activation function is always applied to the score function before computing the loss, the notation $f()$ is used to refer to that activation function). The CCE applied the one-hot encoding to compute the target score. For example, considering applying neuron network to train a model with 5 categories. The label can be represented as [0, 0, 1, 0, 0], and the predicted score can be [.2, .3, .5, .1, .1].

SCCE is similar to CCE except the integer encoding is used to replace the one hot encoding. For comparing two loss functions, 6 sets of experiments were conducted, and the result in our situation, Table 3 indicates that there are no significant differences between two-loss functions. CCE was chosen for the rest of the studies because it retains all of the information (it can measure both top-1 and top-5 error) and because it was suggested in a paper that it is more robust in overcoming noisy labels.

Table 3. Loss function comparison between CCE and SCCE

Loss function\Evaluation metrics	Time/Epoch (sec)	Accuracy after 1 st epoch	Accuracy after 25 epochs
Categorical Cross Entropy (CCE)	670	0.4908	0.9654
Sparse Categorical Cross Entropy (SCCE)	630	0.4881	0.9613

5.7 CNN Model Performance Evaluation

The major goal of our research is to diagnose herb leaves, and as part of that, I'm comparing the performance of two existing state-of-the-art CNN models and evaluating the success of the best deep learning architecture in the context of plant leaf disease diagnosis. As previously stated, the transfer learning strategy was employed to train all of the deep learning models employed in this work. As seen in Table 4, after 30 epochs of training, the ResNet50 model achieved the best performance with 96.26% and 96.30% on the training and validation dataset, respectively, and this result outperformed VGG16 by a large amount. Therefore, ResNet50 is a good candidate to be considered for heavily loaded backend application.

As seen in Table 5, we provide other information about the two models (e.g., Input size, total parameter, trainable parameter, throughput, and training accuracy after 30 epochs), and it reveals the relationship between the model complexity, training speed, and accuracy. We observed that the ResNet50 model has higher classification accuracy (96.30%) than VGG1, which suggest that the ResNet50 model could produce high performance in the mobile device application as well.

As seen in Fig. 7 and Fig. 8, there is no obvious sign of overfitting in the model that we choose. The ResNet50 model shows the best performance in terms of the training and validation accuracy. In Fig. 7, we see that the ResNet50 can achieve above 94% accuracy by less than 10 epochs of training, in contrast to VGG16 models that were struggling to reach the 90% accuracy even after 10 epochs of training, and the progression line for both accuracy and loss tends to fluctuate more violently. Thus, we believe that the ResNet50 model offers the fastest learning efficiency in our study and, therefore, is the most suitable model to be used in plant leaf disease recognition.

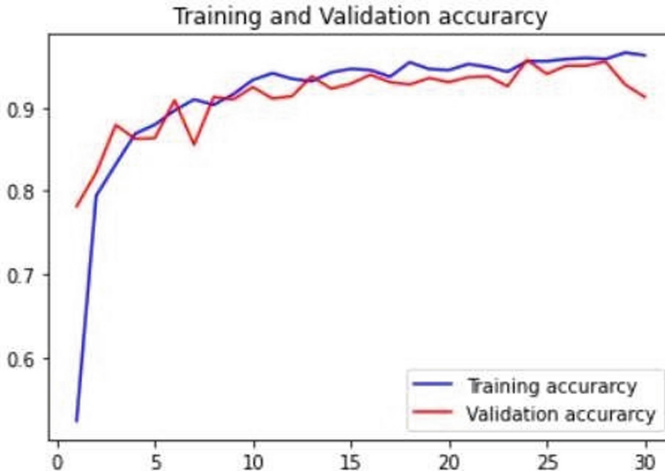


Fig. 7. ResNet50 accuracy and validation accuracy

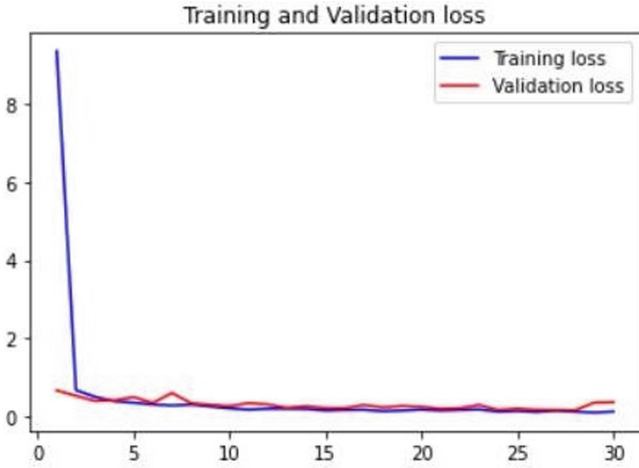


Fig. 8. ResNet50 loss and validation loss

Table 4. DNN models’ performance comparison after 30 epochs of training

Model	Train accuracy %	Validation accuracy %	Training loss	Validation loss
ResNet50	96.26	96.30	0.1370	0.17030
VGG16	93.11	93.67	0.1869	0.22714

Table 5. DNN models’ capacity comparison

DNN model	Input size	Total parameters	Trainable parameter	Throughput (s/epoch)	Train accuracy after 30 epochs (%)
ResNet50	224 × 224 × 3	74,976,143	52,443,151	580	96.26
VGG16	224 × 224 × 3	40,938,319	33,303,055	3000	93.11

6 Conclusion

The proposed approach was created with the wellbeing of farmers and the agricultural sector in mind. Farmers will be able to identify disease in their crops and apply the appropriate sort and amount of pesticides in their fields with the help of such systems. This will not only reduce the excess use of pesticides in the field but will also help in increasing the yield of the farms.

Deep learning algorithms are undergoing extensive research. To diagnose the herbs in this thesis, I used CNN and its network designs ResNet50 and VGG16. The created technology can detect illness in herbs with a 96% accuracy rate. Only if we have a thorough understanding of the disease can we apply the right cure to improve the plant's health. Python is used in the suggested system. The accuracy and speed of processing can be improved by using Google's GPU.

In our study, ResNet50 deep learning architecture was tested to be the most suitable CNN model in applying to the image-based plant leaf disease recognition on our dataset. Throughout the study, I have conducted a comprehensive literature review on the past related research on the field of deep-learning-based herb disease detection and provided a series of empirical experiments in applying the techniques, including tuning the performance by varying train-valid set split ratio, pre-trained CNN models, loss functions, and batch size. All the experiments conducted in the study were trained on the googles Colab platform with the help of their GPUs. The result of this study showed that the ResNet50 neural network architecture outperformed the VGG16 architecture. Specifically, we demonstrated in the experimental section that the ResNet50 can achieve accuracy of 96.30% in training and 96.26% in validation over 5-h training and VGG16 can achieve a accuracy of around 93% by over 12 h of training, which suggest that the ResNet50 model is suitable for both lightweight mobile applications and backend workstation purpose development within the context of plant leaf disease recognition.

7 Future Work

Plant disease poses the main threat to the agricultural development of the world, and especially critical to the smallholder farmers in many developing countries. Therefore, affordable and accurate automatic plant disease detection can be valuable tools to provide early warning and forecast that mitigating the efforts to control disease propagation. The application of the deep learning approach has been growing quickly in the field of plant disease diagnosis. The result of deep learning approaches reported in many works of literature so far had shown a promising future. However, the deep learning approach is not omnipotent. There are some shortcomings and challenges that I have not mentioned in the thesis, such as its lack of adaptively to the ever changing environment, poor interpretability of the model, and the demand for a large volume of the dataset. Therefore, one important goal that our future work will be aligned on is the development of robust image detection that only requires a small set of image samples.

There can be other enhancements to the proposed model, one enhancement which can be very useful to the farmers is an alerting mechanism. The system as soon as detects a disease will alert the farmers using buzzer and LEDs so that the spread of the disease can be stopped. The alerting mechanism should only be triggered when a plant has a disease, if it is healthy then no alert is needed. After alerting the farmers, the system should tell the farmers the type and amount of pesticides needed to control the spread of the disease. This will not only reduce the excess use of pesticides in the field but will also help in increasing the yield of the farms.

References

1. Sardogan, M., Tuncer, A., Ozen, Y.: Plant leaf disease detection and classification based on CNN with LVQ algorithm. In: 3rd International Conference on Computer Science and Engineering (2018)
2. Sharma, P., Hans, P., Gupta, S.C.: Classification of plant leaf diseases using machine learning and image preprocessing techniques. In: 10th International Conference on Cloud Computing, Data Science & Engineering (2020)
3. Elangovan, K., Nalini, S.: Plant Disease Classification Using Image Segmentation and SVM Techniques, International Journal of Computational Intelligence Research ISSN 0973-1873, vol.13, no.7, pp. 1821–1828 (2017)
4. Amara, J., Bouaziz, B., Algergawy, A.: A deep learning-based approach for banana leaf diseases classification, Proceedings of the Datenbanksysteme für Business, Technologie und Web (BTW 2017) - Workshopband (2017)
5. Sharath, D.M., Akhilesh, S., Kumar, A., Rohan, M.G. Prathap, C.: Image based plant disease detection in pomegranate plant for bacterial blight. International Conference on Communication and Signal Processing (2019)
6. Saradhambal, G., Dhivya, R., Latha, S., Rajesh, R.: Plant disease detection and its solution using image classification. Int. J. Pure Appl. Math. **119**(14) (2018)
7. Durmuş, H., Güneş, E.O., Mürvet KÖrcü: Disease Detection on the Leaves of the Tomato Plants by Using Deep Learning. Department of Electronics and Communication Engineering Istanbul Technical University, Electrical and Electronics Engineering Faculty, Maslak, Istanbul, Turkey (2017)
8. Belkin, M., Hsu, D., Mitra, P.: Overfitting or perfect fitting? Risk bounds for classification and regression rules that interpolate (2018)
9. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? (2014)
10. Mishkin, D., Sergievskiy, N., Matas, J.: Systematic evaluation of CNN advances on the ImageNet (2016)
11. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier Nonlinearities Improve Neural Network Acoustic Models. ICML (2013)
12. Ferri, C., Hernández-Orallo, J., Modroui, R.: An experimental comparison of performance measures for classification. Pattern Recogn. Lett. **30**, 27–38 (2009)
13. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes (VOC) challenge. Int. J. Comput. Vis. **88**(2), 303–338 (2010)
14. Dryden, N., Maruyama, N., Moon, T., Benson, T., Snir, M., Van Essen, B.: Channel and filter parallelism for large-scale CNN training. In: SC 2019: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–20 (2019)
15. Geetharamani, G., Arun Pandian, J.: Identification of plant leaf diseases using a nine- layer deep convolutional neural network. Comput. Electr. Eng. **76**, 323–338 (2019)
16. Géron, A.: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, Sebastopol, CA (2019)
17. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge, MA (2017)
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. IEEE Conference on Computer Vision and Pattern Recognition CVPR, pp. 770–778 (2016)
19. Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P.: On Large-Batch Training for Deep Learning: Generalization Gap and Sharp. ICLR (2017)
20. Li, M., Zhang, T., Chen, Y., Smola, A.J.: Efficient mini-batch training for stochastic optimization. In: KDD 2014: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 661–670 (2014)

21. Pantazi, X.E., Moshou, D., Tamouridou, A.A., Kasderidis, S.: Leaf disease recognition in vine plants based on local binary patterns and one class support vector machines. In: Iliadis, L., Maglogiannis, I. (eds.) *Artificial Intelligence Applications and Innovations*. IFIPAICT, vol. 475. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44944-9_27
22. Kawasaki, Y., Uga, H., Kagiwada, S., Iyatomi, H.: Basic study of automated diagnosis of viral plant diseases using convolutional neural networks. In: Bebis, G., et al. (eds.) *ISVC 2015*. LNCS, vol. 9475, pp. 638–645. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27863-6_59
23. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning Transferable Architectures for Scalable Image Recognition. Arxiv (2017)
24. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. Arxiv (2016)
25. Zhang, Z., Sabuncu, M.R.: Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels. Arxiv (2018)
26. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the Inception Architecture for Computer Vision (2016)
27. Koushik, C., Madhav, A.V.S., Singh, R.K.: An efficient approach to microarray data classification using elastic net feature selection, SVM and RF. *Journal of Physics: Conference Series*, vol. 1911, no. 1. IOP Publishing (2021)
28. Singh, R.K., Sivabalakrishnan, M.J.P.C.S.: Feature selection of gene expression data for cancer classification: a review. *Procedia Comput. Sci.* **50**, 52–57 (2015)