Vol 7 No 1 (2025) *e*-ISSN: 2682-8383

Journal of Engineering Technology and Applied Physics

An Improved Convolutional Neural Network (CNN) for Disease Detection and Diagnosis for Multi-crop Plants

Florence Choong Chiao Mei* and Bryan Ng Jan Hong
Engineering and Physical Sciences, Heriot-Watt University Malaysia, Putrajaya, Malaysia.
**Corresponding author*: f.choong@hw.ac.uk, *ORCiD*: 0000-0002-6958-8725
https://doi.org/10.33093/jetap.2025.7.1.2
Manuscript Received: 17 March 2024, Accepted: 8 July 2024, Published: 15 March 2025

Abstract — Agriculture is one of the largest sectors that contribute to the economic growth of countries, including Malaysia. However, plant diseases affect the quality of the harvest and impede farmers' maximum yield output. Therefore, early detection of diseases in plants is vital to curb infection, reduce food waste, and reduce their carbon footprint. However, many detection methods are complex, require high computational power and time to perform the required analysis and focus only on a particular species or strain of the disease. These requirements would likely deter most users in remote areas or poorer economic states. This paper proposes a convolutional neural network to determine multi-class plant diseases that is memory efficient, has a small trainable parameter number, and is compact enough to work even on mobile devices. The plant images were pre-processed to ensure that they were validated accurately and to minimise overfitting. Then, the proposed convolutional neural network was trained using a publicly available dataset consisting of 54306 images, followed by validation and testing. Finally, the completed model is saved, and the data obtained is transferred to a cloud network using wireless sensor networks. The proposed method obtained 96.87% accuracy with 100 epoch training iterations, rivalling famous architectures such as VGG16 and MobileNetV2. The experimental results demonstrate the feasibility and robustness of the method for disease detection in multi-crop plants.

Keywords—Plant disease, Agriculture, Convolutional neural network, Image processing, Wireless sensor network

I. INTRODUCTION

Agriculture is one of the largest sectors that contribute to the economic growth of countries, including Malaysia. Increasing the crop yield and global trade of agricultural products is crucial to meeting the demands of the population and ensuring food security. However, plant diseases impede farmers' maximum yield output, earning fewer returns and fewer profits for their harvest. Research has shown that plant-borne diseases and pests could deprive the population of 82% of cotton and 50% of other crops [1]. Plant diseases could also directly affect human health by contaminating human food with toxic compounds [2].

Plant disease detection and diagnosis have become at the forefront of agriculturalresearch due to their devastating effects. Attempts at automating plant disease detection and decision-making have been carried out throughout the years. However, there is a general lack of appropriate knowledge regarding the management of plant diseases [3]. The most common disease management methods include insecticides and pesticides. Although they offer a higher yield and improved quality of crops, the chemicals used in these deterrents pose a potential risk to humans orother life forms. Long-term exposure to these chemicals can lead to disastrous health effects such as immune suppression, hormone disruption, and cancer.

Deep learning is a technique that is widely researched and implemented rapidly in the agricultural industry. It can help to patch the gaps that exist in previous disease detection methods and produce highly accurate results [4]. Image processing is one of the most common methods for implementing deep learning technology [5, 6]. For example, images of diseased plants can accurately determine what disease they are suffering from, the colour of their leaves, the spots that they have, and other discerning features of healthy leaves. However, many research approaches focus on a particular species or strain of the disease, which tends to ignore glaring factors such as interactions between species of the same phenotype or other potentially harmful diseases that could affect the same crops. The severity of the disease is often overlooked, as each disease stage comes with different symptoms and effects, bearing different outcomes at different discovery and treatment times.

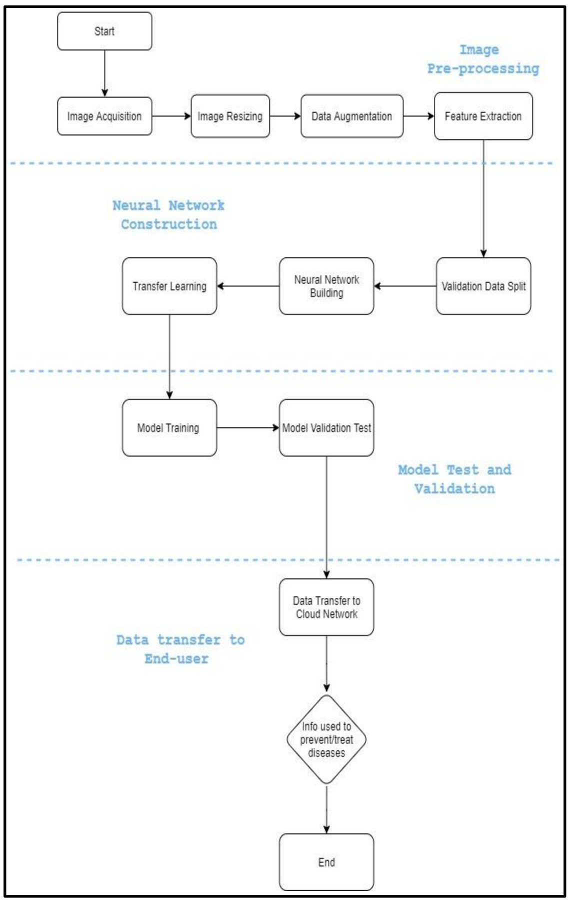
A study using deep learning with artificial intelligence using the MDFC-ResNet training algorithm, a refined algorithm based on ResNet was proposed [7]. ResNet is one of the many architectures that are available in image processing. It works by allowing users to stack building blocks of convolutional layers. The study obtained an accuracy of 93.96%, increasing up to 5.31% from conventional algorithms. MDFC-ResNet employed three stages of disease recognition, feature extraction, and compensation layers to achieve these results. A more compact solution, such as the *mobilenetv2* algorithm, may be implemented to reduce computational power. Another related work proposed aconvolutional neural network (CNN) for rice leaf diseases, and results have shown that *mobilenetv2* performed the best among the small architectures tested, boasting a 97.96% accuracy rate [8]. However, smaller neural networks contain less space for feature extraction, leading to less accurate, fine-tuned results. K-means clustering is another popular method of feature extraction. It separates the healthy parts of the leaves from the diseased parts through RGB colour channels [9]. In other related work, Thenmozi employed MATLAB to generate pictures with segmented features of healthy and diseased portions of leaves from sugarcane crops [10]. Insects that are the most damaging to these crops are sugarcane pyrilla, *Pyrilla perpusilla*; sugarcane whitefly, *Aleurolobus barodenis*; and sugarcane aphids, *Melanaphis sacchari*. The results indicate that general shapes can be obtained from the classification process to help with pest identification.

Although deep learning is a promising method, the barrier to entry into deep learning remains a challenge for farmers in rural areas or those who are not as technologically adept. In addition, having several different convolutional layers will require significant processing power to produce usable results, leading to a higher cost. The computational size required will pose a problem with the availability of the solution in more rural markets. Smaller architectures can be explored to reduce the computational space needed for feature extraction or high-definition images. However, the lack of features in an image processing algorithm means that minute details may not be considered, leading to less accurate results. Therefore, this paper focuses on disease detection and diagnosis occurring on plant leaves caused by insect-borne diseases. The data sample collected covers various valuable crops, such as potatoes, tomatoes, and cotton. The novelty of this research lies in its versatility. A lightweight yet robust solution is proposed. A smaller CNN works to be memory efficient, compact enough to work even on mobile devices, and accessible to virtually anyone. In addition, the research considers multi-variant disease detection, where different diseases can affect the same type of crop.

II. METHODOLOGY

A large-scale, pre-trained database, ImageNet, was used which covers a wide range of insect-borne diseases. Using an extensive database allowed for a more economical CNN design, as the model weight training and image selection processes are completed beforehand. A dataset from Kaggle consisting of 54306 images were used to train CNN to identify 14 different crop species bearing 26 different diseases. The flowchart of the system is shown in Fig. 1.

Fig. 1. Flowchart of the proposed system.



The images were pre-processed to ensure that they are validated accurately, and overfitting is minimized. Overfitting is a phenomenon that occurs when the training model generalises thedata it receives and makes predictions based on unseen patterns. An image that is affected by external factors such as noise will limits the system's true potential and creates biases. This effect can cause a falsely high accuracy rating on the dataset portion of the dataset but a poor one on the validation dataset [11]. Generally, this can be solved using a larger dataset, but that would mean sacrificing cost-effectiveness for extra computational power. Data augmentation works by creating new yet similar images to prevent the model from generating patterns from unseen data. The most common methods include image transformations, including flipped images, reversed images, mirrored images, and duplicated images. Figure 2(a) shows a sample image taken from the dataset while Figs. 2(b) - (h) shows the augmented images. The images were resized to 256 ×

256 pixels to ensure that the training of the model does not consume substantial amounts of computational space and power. Critical and discerning features are extracted after the images have been augmented.

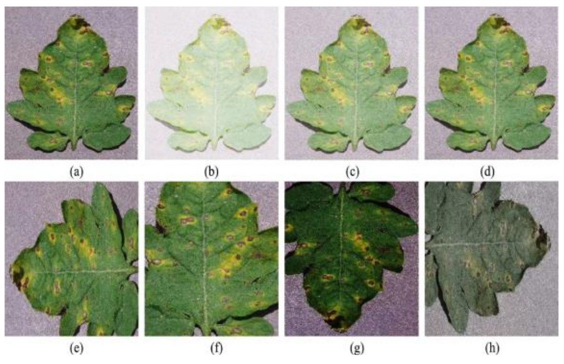


Fig. 2. Original image (a) compared to augmented images (b)-(h).

A. *Training the CNN Model*

The dataset is split between 80/20 for training and validation respectively as it is found that this split ensures that the model will identify pictures that are not part of the original training dataset. Also, this provides the model with a fair comparison for recognising the data and have sufficient new data for validation testing. The learning rate is set within the ranges of 0.01 to 0.0001. A lower learning rate will be used when training the top layers of the classifier and fine-tuning the model for specific results. A higher learning rate is reserved for more general, low-specificity layers. Next, CNN is built around the needs of the images and tested for the best accuracy. To further enhance the model's efficiency, CNN undergoes transfer learning to fine-tune the image categories to obtain better results. After all the data is ready, the model is trained using machine learning algorithms, such as AlexNet, VGG16, and MobileNetV2.

During training, the CNN model is represented by Eq. (1). The features are categorized into four-dimensional tensors and two-dimensional matrices, respectively. The dimensions of the tensors must match, as the image processing workflow in the layers will not function if the matrices in the system are not compatible. At this stage, the output of the convolutional layers serves as the input for subsequent layers.

$$\mathbb{M}_0 : f|(x; \theta_0) \tag{1}$$

, where

$\{(x_i, y_i)\}_{i=1}^{N}, (x, y)$ and $\theta_0 = \{W_0^l, b_0^l\}_{l=1}^{L}$

are the training dataset and original images used for training respectively, x is the input, y is the predicted output, b is the bias and W is the weight. The output h_1 of the 1-th layer is given by Eq. (2).

$$h_1 = \sigma(W_0^l h_{l-1} + b_0^l) \tag{2}$$

The final output after the images is fitted through all the convolutional layers is given by Eq. (3). Softmax represents the activation function that is implemented on CNN. Activation functions are responsible for how well the algorithm can recognize and train the dataset.

$$f(x) = softmax\,(W_0^L h_{l-1} + b_0^L) \tag{3}$$

The training samples represented by Eq. (4) can be duplicated when fed into the CNN via image translation, noise interference, and image mirroring. In addition, data augmentation is also performed before training the data to allow the training data size to be directly modified without changing the training iteration times. Each test image is augmented with images using the same data augmentation parameters. Eq. (5) represents the prediction results of the augmented images.

$$D_t = \{(x_i, y_i)\}_{i=1}^{M} \tag{4}$$

$$f\,(\mathbf{x}) = \frac{M}{\widetilde{M}} \sum_{i=1}^{\widetilde{M}/M} f\,(\widetilde{\mathbf{x}}_i) \tag{5}$$

Once the training is completed, the model is rerun with the validation dataset to ensure the difference in accuracy is not significant. Following that, the completed model will be saved, and the data obtained will be transferred to a cloud network using Zigbee wireless sensor networks (WSNs). The information can be accessed remotely and easily by farmers to ensure that the best solutions can be applied to tackle diseases.

B. *Feature Extraction*

Feature extraction is a necessary part of image processing. The primary purpose of feature extraction is to segment and identify the most crucial parts of an image from each pixel and represent that in a resized image fit for the chosen image classification algorithm. This paper focused on HSV (Hue, Saturation, Value) categorization. HSV categorization allows for greater flexibility between different intensities of colour instead of focusing on a variety of colours. HSV is preferred because plant leaf diseases tend to have a limited colour spectrum. Therefore, focusing on the intensity of the brown or dark spots provides a more accurate analysis of the severity of the disease. Once the colour categorization is selected, colour information is extracted using colour moments (CM). CM requires the mean, standard deviation, and variance of colour, which are calculated using Eqs. (6) - (8).

$$\mu_i = \frac{1}{N} \sum_{j=1}^{N} f_{ij} \tag{6}$$

$$\sigma_i = \left(\frac{1}{N} \sum_{j=1}^{N} (f_{ij} - \mu_i)^2 \right)^{\frac{1}{2}} \tag{7}$$

$$\gamma_i = \left(\frac{1}{N} \sum_{j=1}^{N} (f_{ij} - \mu_i)^3 \right)^{\frac{1}{3}} \tag{8}$$

where f_{ij} is the colour value of the colour components of the individual image pixels and μ_i, σ_i, γ_i (i=1,2,3) represent the mean, standard deviation, and variance respectively.

C. Neural Network Architecture

After the images have been pre-processed and segmented, they are fed into a deep convolutional neural network (DCNN), as shown in Fig. 3.

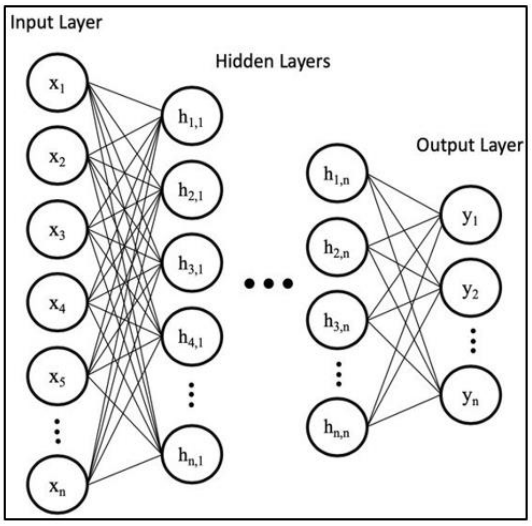


Fig. 3. A DCNN structure with hidden layers and units.

This model consists of input, hidden, and output layers. Each layer is wholly connected to the other and holds individual weight values that add to the output. The weights are then randomly initialized and updated using the Stochastic Gradient Descent (SGD) and a standard instance using Keras libraries hosted in Tensorflow. SGD performs parameter updates with every training epoch, alleviates redundancy by computing gradients at every instance, and refreshes the parameters faster. As the images are passed through the different layers, more features are extracted and become more detailed, as shown in Fig. 4. The hyperparameters tuned for the DCNN, training and validation are provided in the Results section.

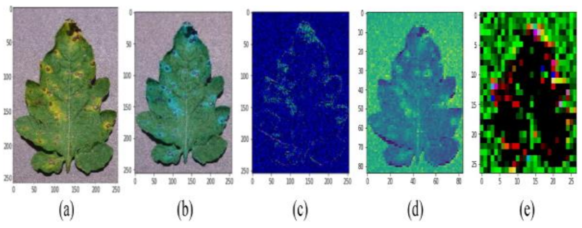


Fig. 4. The stages of the leaf images through a DCNN network.

D. Data Transfer Using Wireless Sensor Networks

Once the model is trained, saved, and all instances have been recorded, the information is transferred to the cloud for easy, remote access. WSNs will be employed to facilitate the intelligent disease detection of this system. WSNs are nodes interconnected by either cellular or Wi-Fi, which will allow users to control and monitor essential parameters such as ambient temperature, soil moisture, humidity, and air pressure. These nodes contain sensors such as temperature and soil moisture sensors to monitor the environment at selected intervals. Once the sensors have detected a change, the data will be updated live on the cloud network.

The wireless network connection uses an XBee Module, which is an embedded solution that utilizes the standard IEEE 802.15.4 networking protocol for fast, real-time data transfer to a cloud IoT platform [12]. The main microcontroller responsible for controlling the sensors in each WSN is an Arduino Mega ATMega2560. An Amazon-based cloud storage solution, AWS Quicksight, is used alongside Tensorflow, which is the primary image processing software. Figure 5 shows the block diagram of the WSNs connected to the Quicksight database, where real-time IoT data is hosted.

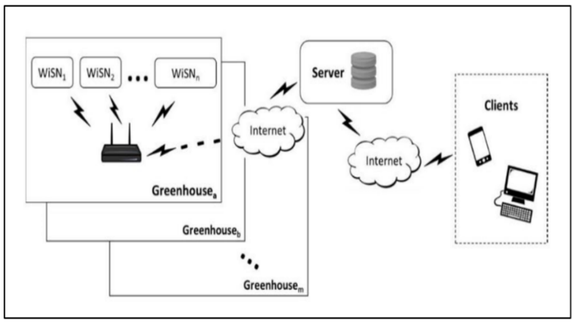


Fig. 5. Online data transmission.

III. RESULTS AND DISCUSSION

A. Model Training

After the images are fed through the neural network, the model's parameters are configured for training. First, the images were fed through a pooling layer containing the image's dimensions [13]. Its primary function is to down-sample a high-dimensional image without changing the depth. Max pooling is used to reduce the size of the processed feature map to allow faster and more compact results. The dropout layer implemented is customary for improving generalization and reducing overfitting, especially in larger datasets, by randomly disabling a proportion of neurons during training, encouraging the network to learn better and produce the most optimal results. For smaller datasets, a higher dropout rate is selected while a lower dropout rate is selected for larger datasets. Different dropout values have been tested to obtain the best results, as shown in Fig. 6. These values were selected based on the choice of the hyperparameters and the end goal of the model. For example, a dropout rate of 0.2 indicates that there is a 20% chance of a neuron being dropped.

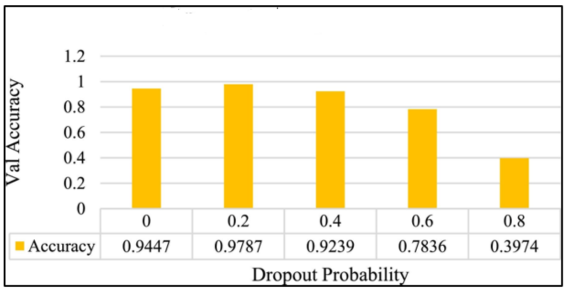


Fig. 6 Comparison of dropout values from 0.2 to 0.8.

It can be observed that a dropout value of 0.2 brings the highest validation test accuracy. Using this dropout value, the images are fed through the final dense layer, connecting all convolutional and pooling layers and merging them into a single output. ReLu was used as the activation function for multi-class operations. The batch size, which determines how many iterations per epoch the model will run, is also determined using trial and error for the best results. Batch sizes affect how quickly or slowly a dataset will converge. Smaller batch sizes suffer from high overfitting rates, but larger batch sizes take longer to converge and require more computational power [14]. Figure 7 displays the batch sizes attempted for this research.

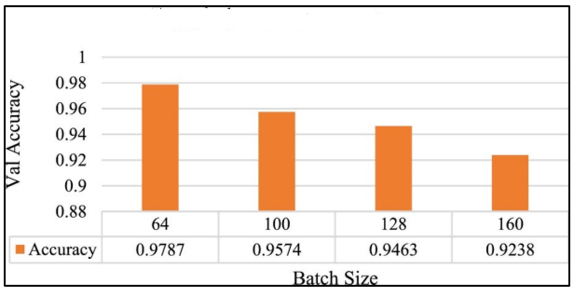


Fig. 7 Batch size comparison from 64 to 160.

A batch size of 64 is ideal for this study, producing up to 97.8 % accuracy on validation tests. The dataset is relatively small compared to other big data analysers. The CNN chosen is more compact than standard DCNN, which means faster convergence can lead to better results. Epochs are defined as the number of times the model will be run through the entire training dataset. Generally, the more iterations a model goes through, the more accurate it gets. From Fig. 8, it can be observed that the trend remains true until around 1000 epochs, when the accuracy rate starts to plateau. The plateauing is caused by the training dataset and validation dataset reaching convergence and learning everything about the dataset.

The dataset was divided between training and validation using an 80/20 split, respectively, to give the model a fair chance of recognizing the data and have enough new data for validation testing. The learning rate was set within the range of 0.01-0.0001. A lower learning rate is used when training the top layers of the classifier and fine-tuning the model for specific results as the neural network goes deeper. A higher learning rate is reserved for more general, low-specificity layers. Table I shows the hyperparameters that have been tuned for this research.

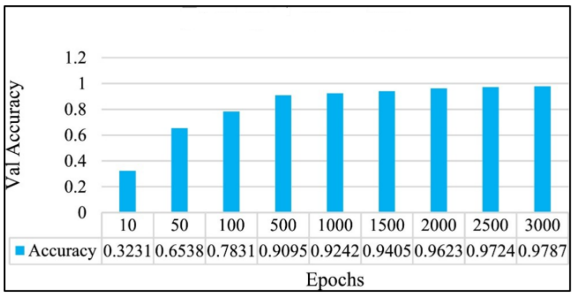


Fig. 8 Epochs iteration comparison from 10 to 3000.

Table I. Hyperparameters tuned for DCNN.

Parameters	Value
Training Epochs	3000
Learning Rate	0.01-0.0001
Batch sizes	64
Training test size	54306
Validation set size	3900
Test set size	1950
Dropout value	0.2

B. *Model Testing*

The PlantVillage dataset, which is available on Kaggle, was used for testing [15]. The training model used is based on the VGG16 structure. It is a custom-built CNN that consumes less memory as compared to a traditionally trained learning model architecture. Figure 9 shows the structure of CNN.

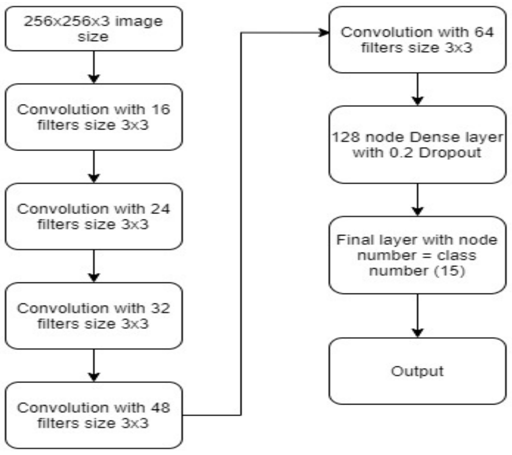

256x256x3 image size
Convolution with 16 filters size 3x3
Convolution with 24 filters size 3x3
Convolution with 32 filters size 3x3
Convolution with 48 filters size 3x3
Convolution with 64 filters size 3x3
128 node Dense layer with 0.2 Dropout
Final layer with node number = class number (15)
Output

Fig. 9. Convolutional neural network layout.

The total number of trainable parameters in this CNN structure amounted to 2,257,984. Compared to traditional models such as AlexNet and GoogLeNet, it is a much more compact solution to image processing. The dataset was resized to fit the batch size of 256×256×3 to facilitate uniform image recognition. The dataset has been divided into 15 different classes, and the final dense layer of the network has 15 nodes. Each CNN layer increased by 16 as the network got more complex than the last fully connected layer. A 3×3 convolutional layer structure was used instead of

5×5 to reduce the computational complexity. Table II shows the comparison between different CNN architectures and their trainable parameters.

Table II. CNN architectures and their trainable parameters.

CNN	Number of Parameters
VGG16	138 million
Inception V3	23.8 million
MobileNetV2	2.3 million
NasNetMobile	4.3 million
Proposed CNN	2.2 million

The dataset has been pre-trained on the extensive ImageNet database with its included weights. Then, the dense layers were trained with random weights initialized during the sequence. ImageNet is an extensive free-access database hosting 3.2 million images in over 5,000 categories. Since the dataset is a multi-class problem, the categorical_crossentropy loss function was used.

The PlantVillage dataset contained 54306 images across 15 different vegetable classes, including healthy tomato, potato, and pepper bell leaves. Some of the leaf images were infected with early and late blight diseases, bacterial spots, and mold. The training samples were augmented to increase the number of samples in the original data set. After a series of pre-processing steps, such as denoising the original leaf disease images, 26 images were randomly selected from each category for 90°, 180°, and 270° rotations, up- and-down swapping, and left-right swapping for data augmentation. The augmented images were used as training samples for the proposed CNN model.

Figure 10 shows the accuracy of the trained model. It can be observed that it grows exponentially in just 300 training steps. The pre-trained weights from ImageNet were frozen in the convolutional layer, allowing the top layers of the classifier to be trained, which allowed for a quick training time and high accuracy. Other image processing techniques, such as VGG16 and InceptionV3, were tested using the same dataset as well. These algorithms are more complex, and different hyperparameters had to be adjusted before they could be used. Training from scratch for these complex algorithms achieved at least 80 % accuracy, but it was less satisfactory than the proposed CNN network. Fine-tuning improved all training networks' accuracy and even outperformed some baseline training models. Fine-tuning helped improve training times by using pre-trained weights from established datasets like ImageNet.

Table III shows the performance obtained by different training methods. Although the proposed CNN does not have the best performance, the training time is the shortest and requires the least computational power among the CNN models tested, using 3GB of RAM throughout the training and classification periods. Although VGG16 yielded the highest accuracy, it is a large, sequential-type model that requires complex 5×5 convolutional filters and many fully connected layers. The proposed CNN uses a simple network and lower-order convolutional filters, keeping accessibility in mind.

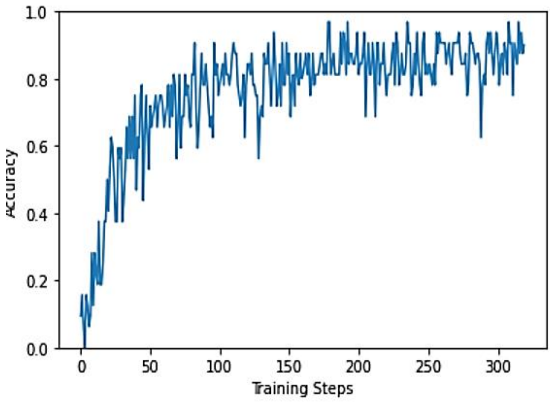


Fig. 10. Tensorflow trained model.

Table III. Comparison between learning methods and CNN Models

CNN Model	Training Methods	Validation Accuracy	Training time (Second)
VGG16	Train from scratch	89.19%	385.728
	Transfer learning	86.52%	301.902
	Fine tuning	97.12%	251.011
Inception V3	Train from scratch	91.17%	385.661
	Transfer learning	72.09%	311.735
	Fine tuning	96.37%	261.872
MobileNetV2	Train from scratch	78.84%	390.025
	Transfer learning	77.52%	334.976
	Fine tuning	96.12%	285.431
NasNetMobile	Train from scratch	79.98%	392.726
	Transfer learning	78.21%	347.561
	Fine tuning	96.95%	299.165
Proposed CNN	Trained using ImageNet	86.52%	126.716

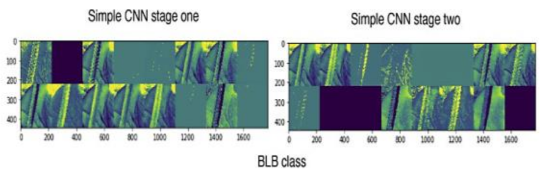


Fig. 11. First convolutional neural layer.

Figure 11 shows the first convolutional layer of the proposed CNN network. The bacteria light class contains 16 two-dimensional images of size 256×256, resized in the hyperparameters as mentioned earlier. The first layer of any convolutional networktends to be the general layer, saving regional features of an image instead of segmented details. Activations retain all previous input from the images in the previous layer. Some of the filters remain blank because they were not activated. Figure 12 demonstrates that the last layer uses all learned features and activations, leaving no blank spaces.

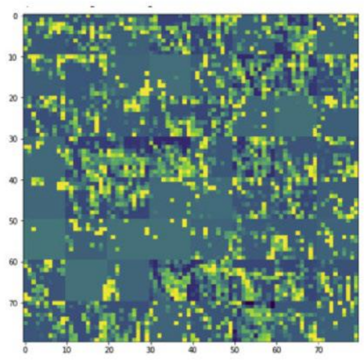


Fig. 12. Final convolutional layer with all activated filters.

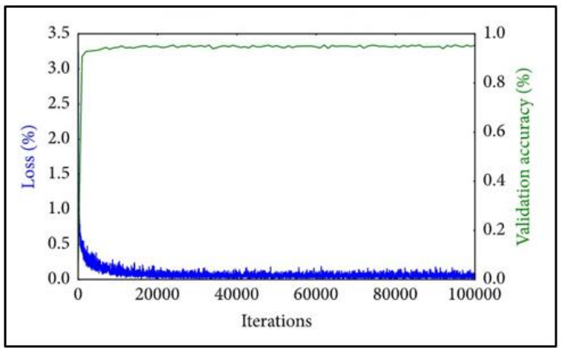


Fig. 13. Accuracy and loss comparisons after 100 epochs.

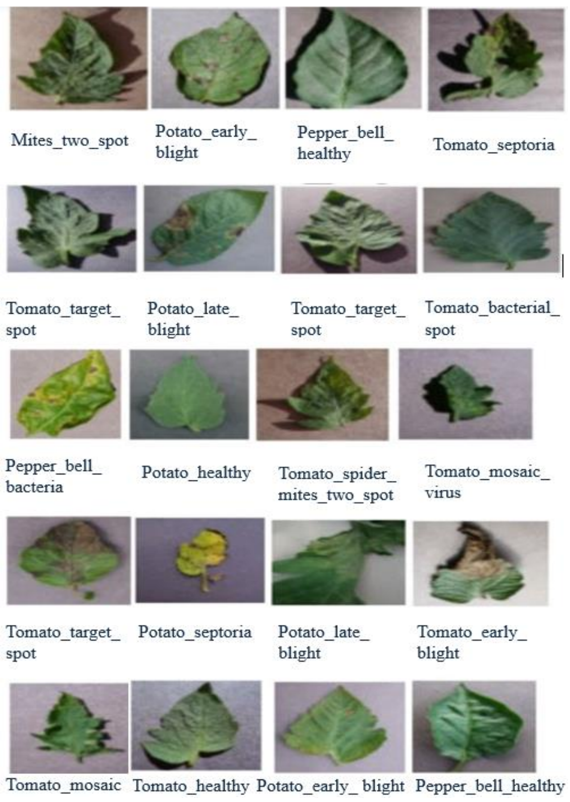


Fig. 14. Model predictions for the plant village dataset.

To enhance the accuracy rate of the proposed CNN, more training parameters were fine-tuned to achieve better results. Figure 13 shows the results of the proposed CNN after 100 epochs of training sessions. The overall accuracy of the proposedCNN increased from 86.52% to 96.87% after fine-tuning and training for over 100 epochs. However, after the 20th epoch, accuracy started plateauing, as did the loss.

Figure 14 shows the predicted leaves from each class in a grid of 30. The model works as expected and can predict multiple classes and be displayed in grid form. Tensorflow was used to train a machine learning model to predict the dataset. Overfitting is also a glaring issue when it comes to enormous datasets. One way to optimize overfitting would be to work with smaller training datasets of about 1000–1200. Data augmentation is also an effective way to alleviate overfitting. It involves extracting more information from the initial dataset to further classify each image from each other to ensure that the learning model performs well, even in challenging data situations. Some of the methods considered include geometric transformations, cropping, flipping, and rotation.

IV. CONCLUSION

This paper sets out to reinvigorate image classification methods for insect-borne diseases in plants. A comparison of some of the most common image processing techniques was pitted against the proposed CNN, with better depth performance and feature-packed layer networks from higher computational algorithms. It is also shown that a simple CNN, pre-trained on ImageNet and fine-tuned, can also provide similar accuracy results at 96.87% as rival other algorithms. In addition, the CNN network has a small trainable parameter number, which keeps memory usage low and makes the solution easily accessible. Furthermore, the simple and user-friendly interface allows new users unfamiliar with agrotechnology to handle image processing technology without the steep learning curve usually associated with it. It also reduces the amount of manual work required to be done by farmers, provides real-time updates on the condition of the crops, and gives them complete control over their field from anywhere in the world.

Future work will involve extending the use of the model by training it for plant disease recognition on more expansive land areas, combining aerial photos of orchards and vineyards captured by drones with CNN for object detection. Image data from a smartphone may be supplemented with location and time information for additional improvements in terms of accuracy. In addition, with the increasing number and quality of sensors on mobile devices, accurate diagnoses via the smartphone can be performed. Combined with the excellent prospects of IoT technology and its flexible array of neural convolutional networks, these steps will improve the technology’s practicality and penetration in practice.

ACKNOWLEDGEMENT

The authors appreciate the support from Heriot-Watt University Malaysia for providing the facilities, and financial support for this research.

REFERENCES

[1] S. Chakraborty and A. C. Newton, "Climate Change, Plant Diseases and Foodsecurity: An Overview," *Plant Pathol.*, vol. 60, no. 1, pp. 2-14, 2011.

[2] Al-Sadi and M. Abdullah, "Impact of Plant Diseases on Human Health," *Int. J. Nutrit., Pharmacol., Neurol. Disea.*, vol. 7, no. 4, pp. 21-22, 2017.

[3] H. Dun-chun, Z. Jia-sui and X. Lian-hui, "Problems, Challenges and Future of PlantDisease Management: from an Ecological Point of View," *J. Integrat. Agricul.,* vol. 5, no. 4, pp. 705-715, 2016.

[4] M. B. I. Reaz, F. Choong, M. S. Sulaiman, F. Mohd-Yasin and M. Kamada, "Expert System for Power Quality Disturbance Classifier," *IEEE Trans. Power Deliver.*, vol. *22*, no. 3, pp. 1979-1988, 2007.

[5] A. Imtiaz and P. K. Yadav, "A Systematic Analysis of Machine Learning and Deep Learning Based Approaches for Identifying and Diagnosing Plant Diseases," *Sustain, Operat. and Comput.*, vol. 4, pp. 96-104, 2023.

[6] B. Punam and G. Pushkar, "Plant Disease Detection Using Hybrid Model Based on Convolutional Neural Network." *Artif. Intellig. Agricul.,* vol. 5, pp. 90-101, 2021.

[7] H. Wei-Jian, J. Fan, D. Yong-Xing, L. Bao-Shan, X. Naixue and E. Bekkering, "MDFC-ResNet: An Agricultural IoT System to Accurately Recognise Crop Diseases," *Special Sect. Data Mining for Internet of Things*, vol. 8, no. 6, pp. 115287- 115298, 2020.

[8] P. K. Sethy, N. K. Barpanda, A. K. Rath and S. K. Behera, "Deep Feature-based Rice Leaf Disease Identification using Support Vector Machine," *Comput. andElectron. Agricul.*, vol. 175, no. 8, pp. 1-9, 2020.

[9] K. Usha, S. J. Prasad and G. Mounika, "Leaf Disease Detection: Feature Extraction with K-means clustering and Classification with ANN," in *Third Int. Conf. Comput. Methodol. and Commun.*, pp. 1095-1098, 2019.

[10] K. Thenmozhi and S. Reddy, "Image Processing Techniques for Insect Shape Detection in Field Crops," in *Proc. Int. Conf. Invent. Comput. and Informat.*, pp. 699-704, 2017.

[11] Q. Zheng, M. Yang, X. Tian, N. Jiang and D. Wang, "A Full Stage Data Augmentation Method in Deep Convolutional Neural Network for Natural ImageClassification," *Discr. Dyna. Nat. and Soc.*, vol. 2020, pp. 1-11, 2020.

[12] D. H. G. Shweta and B. Saraf, "IoT Based Smart Irrigation Monitoring and Controlling System," in *2nd IEEE Int. Conf. Recent Trends in Electron. Informat. & Commun. Technol.*, pp. 815- 819, 2017.

[13] P. J. Arun and G. Geetharamani, "Identification of Plant Leaf Diseases using a Nine- layer Deep Convolutional Neural Network," *Computers & Electrical Engineering*, vol. 76, no. 4, pp. 323-338, 2019.

[14] I. Kandel and M. Castelli, "The Effect of Batch Size on the Generalizability of the Convolutional Neural Networks on a Histopathology Dataset," *ICT Express*, vol. 6, no. 4, pp. 312-315, 2020.

[15] H. Xiao, K. Rasul and R. Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms,", *arXiv*, pp. 1-6, 2017.