Journal of Advanced Research in Applied Sciences and Engineering Technology

Journal homepage:
https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/index
ISSN: 2462-1943

Android-Based App Guava Leaf Diseases Identification using Convolution Neural Network

Choon Fai Foo[1], Kim Gaik Tay[1,*], Osamah Al-Qershi[2], Audrey Huong[1], Chang Choon Chew[1], Shehab Abdulhabib Alzaeemi[1]

[1] Faculty of Electrical & Electronic Engineering, Universiti Tun Hussein Onn Malaysia (UTHM), Parit Raja, 86400 Batu Pahat, Johor, Malaysia
[2] University of Melbourne, Parkville VIC 3052, Australia

ABSTRACT

Guava is an economically significant fruit crop in many regions. Conventional plant disease identification depends on skilled workers visually inspecting samples, which can be laborious and error-prone. Therefore, the goal of this study is to create an Android app that uses InceptionV3 to precisely determine the health of guava plants, with an emphasis on hyperparameter tuning and ideal model selection. Inception V3 was chosen as the base model for this study as it outperformed the 25 pre-trained models from Keras API. The Guava leaves dataset consists of 120 raw images collected from real-time capture and the Google website. The images belong to four categories: Algal Spot, Red Rust, Whitefly, and Healthy leaf. The dataset was split into 60% for training and 40% for validation. To increase the number of images, augmentation was performed to multiply each training by 5 times. The 2048 features were extracted from the second last layer of Inception V3. These features were saved in the Numpy format to facilitate hyperparameter tuning. The best hyperparameter tuning results were achieved with a batch size of 4, a learning rate of 0.001, and the Adamax optimizer for 50 epochs, resulting in a validation accuracy of 0.99. This set of hyperparameters was used to train the model, yielding in a training accuracy of 0.98 and a validation accuracy of 0.9. The trained model was exported in the tflite file format and integrated into an Android app developed using Android Studio. To evaluate the performance evaluation of the app, 15 unseen images per class were used and the app achieved an overall F1-Score of 0.85, a recall of 0.85, a precision of 0.89, and a test accuracy of 0.85.

Keywords:
Guava leaf disease; CNN; Deep learning; Transfer-learning; IncepationV3

1. Introduction

Malaysia's advantageous location makes it a good fit for the agriculture sector [1]. The tropical guava fruit, which grows in East Asia, Central America, South America, and South Africa, is a good source of bioactive substances such ascorbic acid, lycopene, beta-carotene, and dietary fibre. These substances provide defence against ailments like diabetes, heart disease, and cancer [2].

* *Corresponding author.*
E-mail address: tay@uthm.edu.my

https://doi.org/10.37934/araset.57.1.7388

Regretfully, physical labour and antiquated techniques are frequently used in guava planting. Agricultural workers must spend a lot of time and effort using this antiquated method, which involves closely examining the leaves' colour and markings to identify diseases. The performance of the agriculture sector as a whole may suffer as a result.

Convolutional neural networks (CNNs) have been applied to many fields including medical field to detect meningioma tumour by Anita and Kumaran [3] and autism spectrum disorder presented by Neeharika and Riyazuddin [4]. These developments motivate more research into CNNs for the diagnosis of plant diseases, such as those affecting guava leaves.

CNN contains a lot of pre-trained models such as AlexNet [5], GoogLeNet or InceptionV1 [6], DenseNet [7], ResNet [8], VGGNet [9], SqueezeNet [10], MobileNet [11], InceptionV2 [12], InceptionV3 [13]. Since building a new model takes a lot of time and computational resources, using the transfer learning from these pre-trained models for categorizing new applications might save time and energy. All of the pre-trained models should be tested using the same training and test dataset in order to determine which model is best for a certain application. This will help to identify which model achieves the highest accuracy for the given application. By performing this comparative evaluation, one can select the most appropriate pre-trained model to achieve optimal results while minimizing the training time and computational resources required.

Existing guava leaf disease identification studies have shown gaps in utilizing advanced techniques like convolutional neural networks (CNNs), limitations in model selection, hyperparameter optimization, and accessibility to real-life image datasets. Therefore, this study aims to fill these gaps by developing an Android app using InceptionV3, focusing on optimal model selection and hyperparameter tuning for accurate guava plant health identification. The app can identify 4 classes of outputs which are healthy, Algal Spot, disease, red rust disease, and whitely disease of papaya leaf. This app aims to assist workers in easily identifying the diseased plant through the camera on their mobile phones by capturing images of the leaves. With this app, workers will be able to identify plant diseases more efficiently, thereby improving productivity and efficiency in the agricultural sector. This study made several notable contributions:

i. It constructs a dataset encompassing four distinct classes of guava leaf disease.
ii. It determines the most effective pre-trained CNN model by evaluating 25 different models available through the Keras API.
iii. It fine-tunes hyperparameters for InceptionV3 using the GridSearch technique to enhance performance.
iv. It creates an Android application designed for the identification of the three classes of guava leaf diseases.

2. Related Works

Harakannanavar *et al.,* [14] conducted a study to determine whether a tomato leaf is healthy or unhealthy, using 600 samples from the tomato village dataset. They utilized Support Vector Machine (SVM), Convolution Neural Network (CNN), and K-Nearest Neighbor (K-NN) algorithm for classification. The proposed approach achieved an outstanding 99.6%. This demonstrates that CNN-machine learning classification techniques are well-suited for achieving the desired accuracy.

Anitha *et al.,* [15] suggested a customized CNN-based approach to distinguish healthy peppers, potatoes and tomatoes with diseases using leaf images. They utilized a dataset containing 15 classes. The proposed CNN model generated a classifier accuracy of 95%.

Falaschetti *et al.,* [16] developed a classification system using OpenMV Cam H7 Plus, a Python programmable camera for real-time plant disease classification. The training stage employed their customized CNN on the ESCA dataset and the PlantVillage-augmented dataset. The accuracy of the system was reported to be 98.10% for the ESCA dataset and 95.24% for the PlantVillage-augmented dataset.

Naik *et al.,* [17] conducted a study to classify 5 types of chili leaf diseases: down curl of a leaf, Geminivirus, Cercospora leaf spot, yellow leaf disease, and up curl disease using a squeeze-and-excitation-based CNN model. They collected images from a self-built chili leaf dataset taken from the surrounding areas of Madipadu Agraharam. In the self-built chili leaf dataset, there were six classes in total, with five classes representing different chili leaf diseases and one class for healthy chili leaves. Each class contained the same ratio of images with 850 images per class. The researchers used 12 pre-trained models to classify the plant leaf diseases using the chili leaf dataset, both with and without augmentation. VGG19 achieved the highest accuracy of 83.54% without augmentation, while DarkNet53 achieved an accuracy of 98.82% with augmentation images outperforming other models. Additionally, they designed and tested a squeeze-and-excitation-based convolutional neural network (SECNN) model with the chili leaf dataset. The SECNN model demonstrated an accuracy of 98.63% without augmented images and an impressive accuracy of 99.12% with augmented images. Lastly, the SECNN model was tested using PlantVillage data including leaves from various plants such as apple, cherry, corn, grape, peach, pepper, potato, strawberry, and tomato. When classifying the 43 different classes of plant leaf datasets, the SECNN model achieved an accuracy of 99.28%.

Russel and Selvaraj [18] designed CNN models to assist plant disease detection. They incorporated a customized filter known as Law's Mask, which served as a learnable filter to facilitate adaptive learning. Law's Mask was specially designed for extracting texture features that accurately characterize various patterns. Three datasets were used: the MepcoTropicLeaf dataset, the Plant Village dataset, and the Leaf Image Data Repository. A quarter of the samples in each dataset were set aside for testing, while the other seventy-five percent were used for training. The Plant Village dataset included 61486 enhanced photos of 14 different species, divided into 38 distinct classifications that included both healthy and diseased leaves. The Data repository leaf images contained 4503 images with 12 species, representing both healthy and diseased conditions. Lastly, the MepcoTropicLeaf dataset encompassed 50 species of herbal plants, with each class having at least 50 images. Using the Plant Village dataset for training, the model had a remarkable accuracy of 99.17% for classifying plant species and 98.61% for classifying plant disease. The model achieved a classification accuracy of 97.16% for identifying plant species and 90.02% for identifying leaf disease when trained on data Repository Leaf Images. The classification accuracy achieved using MepcoTropicLeaf dataset was 90.86%.

Prabu and Chelliah [19] proposed mango leaf disease detection utilizing a CNN enhanced by a crossover-based levy flight distribution technique. For their study, they collected a total of 380 images of both healthy and diseased mango leaves. Out of these images, 300 were mango disease images and 80 images were healthy mango images. The mango disease images were classified into three categories: Mango Anthracnose, Bacterial black spot, and Sooty mould. The dataset was divided into a ratio of 0.7:0.15: 0.15 for training, validation, and testing respectively. To enhance generalization and prevent overfitting, they applied various data augmentation techniques. The learning phase involved the use of MobileNetV2 model. The healthy class accuracy is 99.43%, Mango Anthracnose accuracy is 97.32%, the Bacterial black spot accuracy is 99.13% and Sooty mould accuracy is 99.06%.

Gaikwad *et al.,* [20] proposed fruit leaf disease infected by fungi identification using deep CNN. The study focused on classifying three types of fruit leaves: apple, custard apple and guava. The image

collection, which consists of 14181 images with 10 different classes, both sick and healthy leaves was gathered from a real-time environment. 80% of the images were utilized for training and 20% for testing. They experimented with 3 different versions of input images: colour images, black and white images, and grey images for training. Besides that, they selected AlexNet and SqueezeNet for the training process. The results revealed that the colour image outperformed the black and white images, and grey images. The accuracy obtained using the AlexNet was 86.8%, while the SqueezeNet model achieved 86.6%. Both models demonstrated similar recognition accuracy.

Nandhini and Ashokkumar [21] investigated four types of tomato leaf disease identification: bacterial spot, septoria leaf spot, late blight, and tomato mosaic virus using CNN with an enhanced crossover-based monarch butterfly algorithm. The dataset used in their study comprised 6218 images, which included samples from the four classes of disease and healthy leaves. These images were obtained from the Plant Village database. 90% of the dataset was utilized for training and the remaining 10% was employed for testing. In addition, they employed Vgg16 and InceptionV3 CNN models. Impressive classification accuracy was attained, with Vgg16 getting 99.98% accuracy and InceptionV3 earning 99.94% accuracy.

Mustafa *et al.,* [22] investigated pepper bell leaf disease identification using an enhanced CNN. They gathered a total of 2475 images of pepper bell leaves, both with and without bacteria. To boost the number of dataset images, they employed an augmentation technique using an image generator, producing in a dataset with 20000 images. They divided the images into two groups: training, which received 70% of the images, and testing, which received 30% of the images. In the pre-possessing steps, they performed resizing of the images, edge detection, and converted RGB leaf images to YUV colour space. Then they equalized the intensity value of the YUV leaf images by converting them back to RBG leaf images. These preprocessing steps aimed to enhance the efficiency of the model's classification and training process. The researchers experimented with different types of models, including VGG, ResNet, Inception-ResNet-v2, DenseNet, and Optimized-CNN. Among all the models, the Optimized-CNN model achieved the highest accuracy of 99.99% in identifying plant leaf diseases.

Uguz and Uysal [23] conducted a classification study of olive leaf diseases employing deep CNNs. They collected a total of 3400 olive leaves, which included samples of healthy leaves, Aculus olearius and Olive peacock spot diseases. They conducted two different experiments, one with an augmented dataset and the other without augmentation. For the augmented dataset, they applied various augmentation techniques to increase the diversity of the dataset. In both experiments, the researchers split the image samples into 80% for training and 20% for testing for each class. Besides that, they employed three different models: VGG19, VGG16 and proposed CNN architecture. The experiment results showed that the VGG16 model outperformed for all three classes (healthy, olive peacock spot and aculus olearius) without data augmentation, obtaining an accuracy of 87%. Similarly, with the data augmentation, the VGG 16 model still outperformed the other models, achieving an accuracy of 87% for all three classes.

Thangaraj *et al.,* [24] conducted a comparative study of DenseNet121, DenseNet169, InceptionV3 and Xception on guava leaf disease detection using Kaggle dataset. The dataset comprises 2300 images including classes such as Canker, Dot, Healthy, Mummification and Rust. A split ration of 0.8: 0.2 for training and validation was utilized. They demonstrated that DenseNet169 outperformed the others with an accuracy of 96.12%.

Perumal *et al.,* [25] employed image preprocessing, K-Means clustering for segmentation, GLCM features and then classified guava leaves using SVM. Their dataset comprises 70 guava leaves images, consisting of 3 classes: Anthracnose, bacterial blight and healthy leaves. They achieved an accuracy of 98.17%.

Thilagavathi and Abirami [26] employed image preprocessing and extracted 128 texture features to classify 125 guava leave images across five classes: Algal, rust, curl, powdery mildew and viburnum chindo using both KNN and SVM classifiers. SVM outperformed, achieving the highest accuracy of 98.2% for alga spot, powdery mildew and viburnum chindo.

In summary, only four out of twelve related studies have focused on guava leaf disease identification. However, in a study conducted by researchers [20], they examined four classes of guava leaves alongside four classes of apple leaves and two classes of custard apples leaves using AlexNet and SqueezeNet. Notably, access to their real-life captured images is currently unavailable. It's worth mentioning that the Guava leaf diseases classes used in studies [24-26] differ from this study. Moreover, the studies in [25,26] did not employ CNN. In addition, researchers who did employ transfer learning from CNN did not select the best pre-trained CNN models or optimize the hyperparameters used in the CNN. Therefore, the primary goal of this study is to develop an Android phone app that utilize the best pre-trained CNN model and the best hyperparameter for identifying whether an entire guava plant is afflicted by a disease or remains healthy, employing InceptionV3.

3. Methodology

Figure 1 illustrates the flowchart of the entire project, starting from the selection of the best pre-trained model for guava leaf disease detection, followed by hyperparameters tuning, image acquisition, preprocessing, model training, app development, and app testing stages.

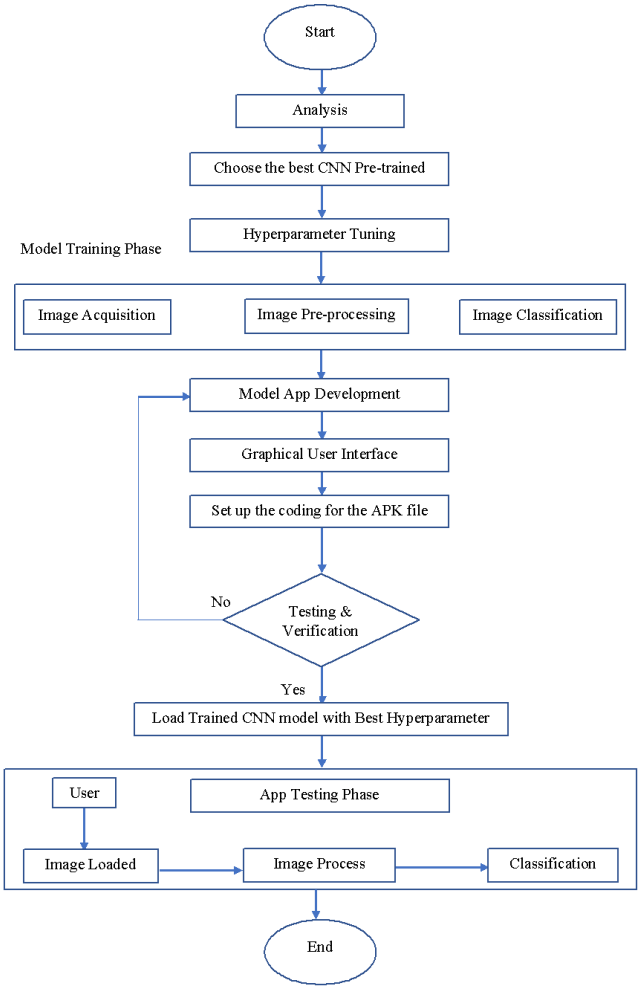

Fig. 1. Project flowchart

3.1 Dataset

The guava leaf diseases, including Algal Spot, Red Rust, Whitefly, and Healthy leaf were collected through real-life capture and Google search engine. The database comprises 120 raw images, which were split into 60% for training and 40% for validation. A 40% validation split was chosen to increase the number of validation images, as the total number of raw images is limited to only 120. Using a lower validation split would result in a smaller validation set, which may not be sufficient for effectively validating the training process. The addition of augmented images can help improve the training dataset further. Additionally, another dataset containing 20 new unseen images with 4 different classes was designated for testing the developed app, as shown in Table 1.

Table 1
Dataset distributions

Distance (m)	Raw Image Data				Saved Augmented Image			
	Algal Spot	Healthy	Whitefly	Red Rust	Algal Spot	Healthy	Whitefly	Red Rust
Training	18	18	18	18	90	90	90	90
Validation	12	12	12	12	12	12	12	12
Testing	5	5	5	5	5	5	5	5

During the training phase, the trained image dataset underwent augmentation techniques such as rescale, random shear, random zoom, and random horizontal flipping by using the image generator. These techniques were applied to increase the number of images for training. Figure 2 illustrates examples of the dataset, presenting both original and augmented images used in this study.

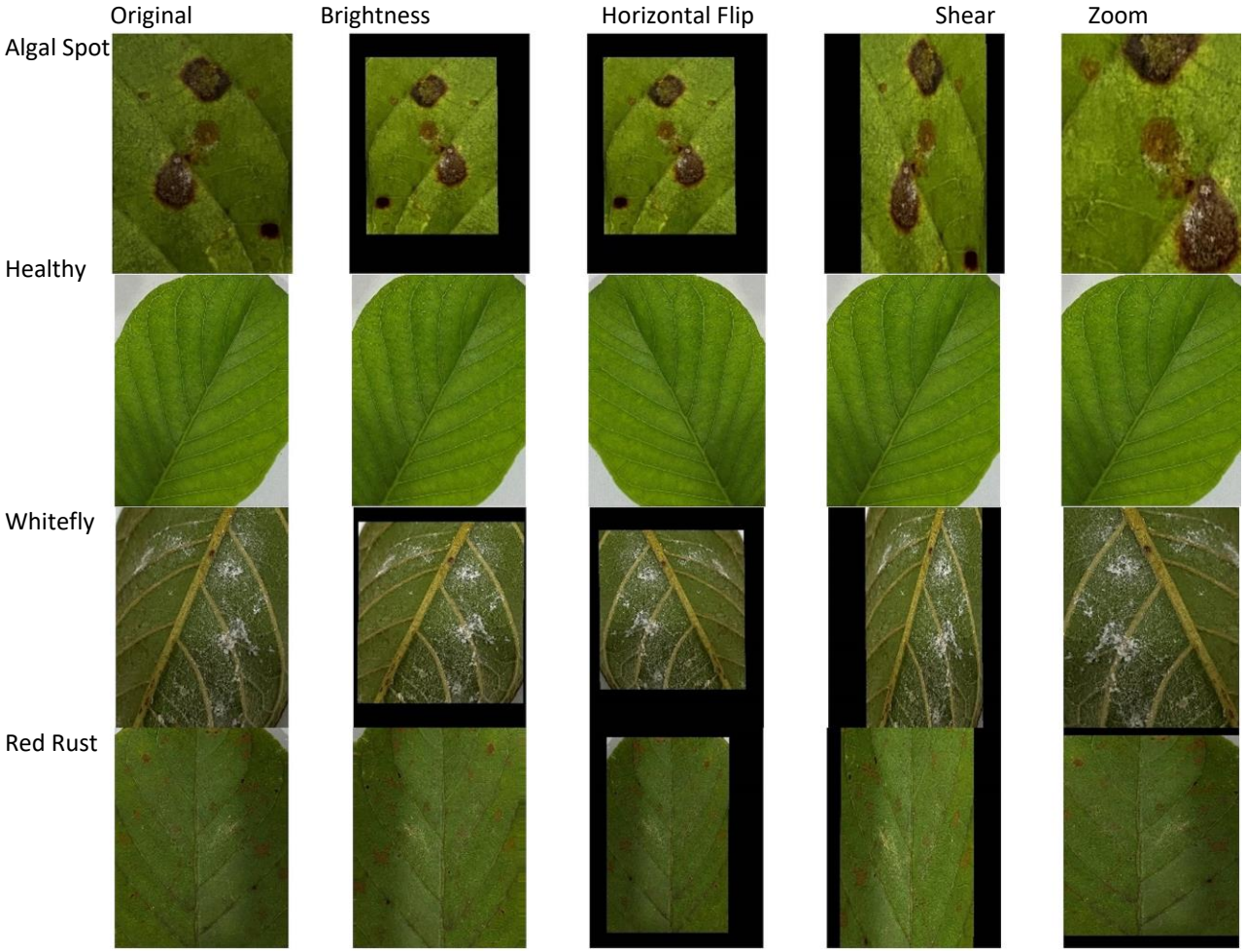

Fig. 2. Samples of original images and the augmented images

3.2 The Best Pre-Trained Model

A Python script was written to load all 25 pre-trained models from Keras library and evaluate their validation accuracy using the guava leave dataset. The script was run for 10 epochs.

3.3 Hyperparameter Tuning

In the hyperparameter tuning step, the images were input at the input layer of InceptionV3 and processed until the second last (average pooling) layer to extract 2048 features. These extracted features were saved in NumPy format to facilitate the tuning of the hyperparameters pipeline of a Keras-based deep learning model using Grid Search with 3 cross-validation as shown in Table 2. The choice of the batch size, learning rate and epochs range is based on studies by Lai [27] and Waheed [28] which concluded that using a small batch size, a small learning rate and large number of epochs can lead to increased accuracy.

Table 2
Hyperparameters range

Hyperparameters	Range
Batch size	[4, 8, 16],
Optimizer	['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam'
Learning rate	[0.01, 0.001, 0.0001]
epochs	[30, 40. 50]

3.4 Model Training

The model started training with the best hyperparameters obtained in Section 3.3. Once the training model was completed, the data was used to plot the graphs for training and validation accuracies and losses. Afterward, the trained model was saved and converted to tflite format. This conversion process enables the model to be imported and utilized within the Android Studio environment during the development stage.

3.5 Performance Evaluation

The performance of the developed App was evaluated using accuracy, precision, recall and F1-score as follows researchers [29,30]:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

$$\text{f1-score} = 2 * \frac{(\text{Recall} * \text{Precision})}{(\text{Recall} + \text{Precision})} \tag{3}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FN + FP} \tag{4}$$

Where TP is true positive, TN is true negative, FP is false positive, and FN is false negative.

3.6 Android-Based App Development with Android Studio

The trained model was then saved and integrated into the mobile app development process. A mobile app was developed using the Android platform and Android Studio Electric Eel. The app includes various elements, including an image view region, a classified result region, a confidence level region, a ‘Take Picture’ button, and a ‘Launch Gallery’ button. Users are provided with the flexibility to choose between capturing a real-time image or selecting an image from the gallery. Figure 3 illustrates the design of the developed Android App User Interface (UI). During the testing phase, the CNN model was loaded in the mobile app to classify whether a leaf has Algal Spot disease, Red Rust disease, Whitefly disease, or is Healthy.

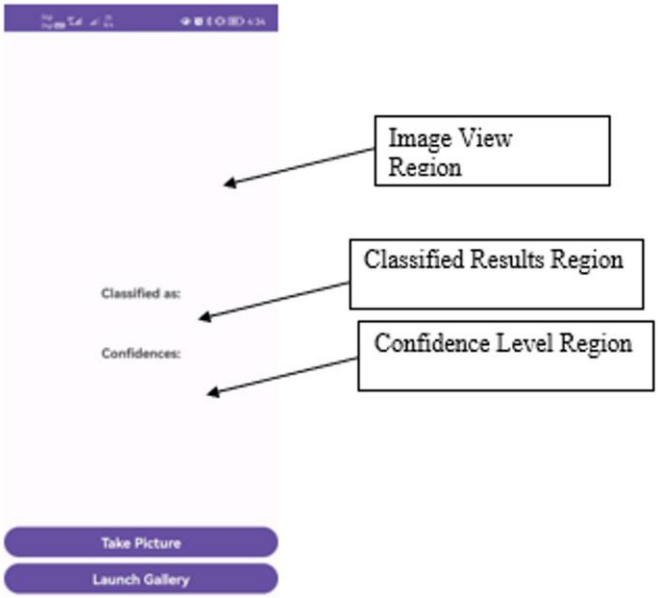

Fig. 3. Developed android app user interface (UI)

4. Results and Discussion

This section discusses the best pre-trained model, hyperparameter tuning, model training and app testing.

4.1 The Best Pre-Trained Model

InceptionV3 was chosen as the trained model for the training process because, among all 25 pre-trained models, it exhibited the highest accuracy of 45.8% compared to the other pre-trained models using 10 epochs as seen in Table 3.

Table 3
All pre-trained models' validation accuracy

model_name	num_model_params	validation_accuracy
InceptionV3	21802784	0.458333343
MobileNetV2	2257984	0.416666657
EfficientNetV2B2	8769374	0.416666657
EfficientNetB4	17673823	0.416666657
EfficientNetB5	28513527	0.416666657
DenseNet121	7037504	0.375
EfficientNetV2S	20331360	0.375
InceptionResNetV2	54336736	0.375
MobileNetV3Small	939120	0.333333343
MobileNetV3Large	2996352	0.333333343
EfficientNetB0	4049571	0.333333343
EfficientNetV2B1	6931124	0.333333343
EfficientNetB2	7768569	0.333333343
DenseNet169	12642880	0.333333343
EfficientNetV2B3	12930622	0.333333343
EfficientNetB6	40960143	0.333333343
EfficientNetV2M	53150388	0.333333343
DenseNet201	18321984	0.291666657

MobileNet	3228864	0.25
EfficientNetV2B0	5919312	0.25
EfficientNetB1	6575239	0.25
EfficientNetB3	10783535	0.25
EfficientNetB7	64097687	0.25
EfficientNetV2L	117746848	0.25

4.2 Hyperparameters Tuning

Table 4 shows that the best hyperparameters were obtained as a batch size of 4, a learning rate of 0.001, and 50 epochs using the Adamax optimizer, which resulted in a validation accuracy of 99%.

Table 4
Tuned Hyperparameter Obtained

Hyperparameters	Best Value
Batch_size	4
Optimizer	Adamax
Learning rate'	0.001
Epochs'	50
Validation accuracy	99%

4.3 Model Training

Figures 4 and 5 depict the training and validation accuracies achieved at the end of the training using the best hyperparameters. The training accuracy reached 98%, while the validation accuracy stood at 90%. When compared to the accuracy gained during the tuning session, the validation accuracy obtained during the training session differed by 0.09. This difference can be attributed to the variation in both scripts using successive layers and different feature extraction techniques. During the hyperparameter tuning session, the second last layer of the InceptionV3 model's extracted features was used to increase tuning effectiveness. On the other hand, in the model training session, the image dataset was inputted throughout the model, starting from the input layer of InceptionV3 to the final dense layer, which included 4 output nodes for each class. The slight difference in validation accuracy may arise because the model training session did not explore multiple permutations of hyperparameters, unlike the hyperparameter tuning session, which exhaustively searched for the best combination.

Classification Report

	precision	recall	f1-score	support
Algal Spot	1.00	1.00	1.00	90
Healthy	0.94	0.99	0.96	90
Red Rust	0.99	0.94	0.97	90
Whitefly	1.00	0.99	0.99	90
accuracy			0.98	360
macro avg	0.98	0.98	0.98	360
weighted avg	0.98	0.98	0.98	360

Fig. 4. Training classification report of model training using the best hyperparameters

Classification Report

	precision	recall	f1-score	support
Algal Spot	1.00	0.92	0.96	12
Healthy	0.92	1.00	0.96	12
Red Rust	0.73	0.92	0.81	12
Whitefly	1.00	0.75	0.86	12
accuracy			0.90	48
macro avg	0.91	0.90	0.90	48
weighted avg	0.91	0.90	0.90	48

Fig. 5. Validation classification report of model training using the best hyperparameters

4.4 Android-Based App for Guava Leaf Disease Classification

The trained model was exported as a tflite file format and utilized in the Android Studio app development process. Users have the option to select the desired image for classification by clicking on either the "Take Picture" button or “Launch Gallery” button. Once the image is imported, it will automatically appear in the image view region of the app. The app will then display the classified result in the classified results region and provide the corresponding confidence level in the confidence level region. Figure 6 illustrates samples of each class’s output result from the developed Android App User Interface (UI).

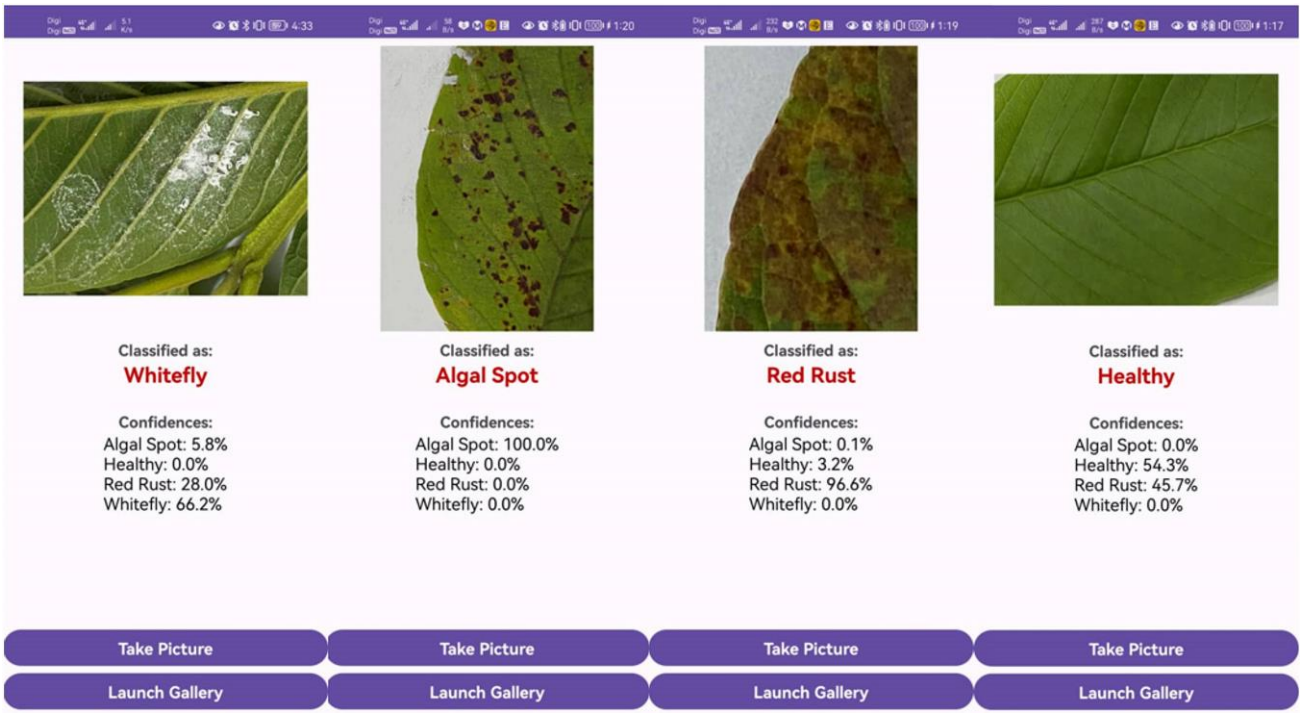

Fig. 6. Developed android app user interface (UI)

The app’s performance was evaluated using a set of 5 unseen images from each class, which were loaded into the image classification app. The results of the categorization process were combined. Table 5 shows the confusion matrix while Table 6 shows the classification report.

Table 5
Confusion Matrix of Designed Android App Test

	Predicted Class			
Actual Class	Algal Spot	Healthy	Red Rust	Whitefly
Algal Spot	5	0	0	0
Healthy	0	4	0	1
Red Rust	1	0	3	1
Whitefly	0	0	0	5

Table 6
Classification report of Designed Android App Test

Class	Precision	Recall	F1-score	Test Accuracy
Algal Spot	0.83	1.00	0.91	
Healthy	1.00	0.80	0.89	
Red Rust	1.00	0.60	0.75	0.85
Whitefly	0.71	1.00	0.83	
Average	0.89	0.85	0.85	

The total performance metrics reveal an average F1-Score of 0.85, a recall of 0.85, a precision of 0.89, and an accuracy of 0.85. The obtained results demonstrate that the proposed Android-Based App can diagnose plant diseases in guava leaves with a relatively high level of accuracy.

Table 7 shows that the healthy guava class in this study outperformed the study by [20] with an accuracy of 89% and is comparable to results of the study by [24]. For the rust class, the existing study achieved a comparable accuracy of 75% to the study by [20]. The overall accuracy of 85% in this study is also comparable to the study [20]. However, it is worth noting that SVM [25,26] and KNN [26] classifiers outperformed CNN techniques in both existing and previous studies. Nevertheless, it should be noted this comparison is based on different datasets and different numbers of test images, which may introduce unfair biases. We believe that increasing the number of images in future studies could lead to improved accuracy. Therefore, in the future, we plan to explore additional sources to increase the number of images used for both training and testing purposes.

Table 7
Comparison of Test Accuracy

Researchers	Dataset	Number of Test Images	Diseases	Methods	Accuracy	
[20]	Own dataset			AlexNet and SqueezeNet	AlexNet	SqueezeNet
		915	Apple healthy		98.9%	95.5%
		450	Apple scab		90.1%	90.9%
		356	Apple black rot		91.5%	87.4%
		108	Apple rust		89.9%	84.4%
		126	Custard apple healthy		96.1%	95.3 %
		82	Custard apple leaf spot		88%	88%
		411	Guava healthy		86.6%	84.2 %
		65	Guava insect eaten		44.6%	56.9 %
		172	Guava rust		75.1%	76.7 %
		145	Guava leaf spot		26.2%	39.3%
					86.8% (overall)	86.6% (overall)
Thangaraj [24]	Kaggle	57	Canker	DenseNet121	89.89%	
		37	Dot	DenseNet169	96.12%	
		259	Healthy	InceptionV3	89.46% (overall)	
		55	Mummification	Xception	90.54%	
		57	Rust			
Perumal [25]	Open store	30	Anthracnose	SVM	96.77%	
		30	Bacterial Blight		98.38%	
		10	Healthy Leaf		99.38%	
Thilagavathi [26]	Own dataset			KNN	KNN	SVM
		25	Algal	SVM	88%	98.2%
		25	Rust		96%	98.2%

		25	Curl		92%	95.45%
		25	Powdery mildew		92%	95.45%
		25	Viburnum chindo		92%	98.2%
Proposed	Own	5	Algal Spot	Optimized	91%	
System	dataset	5	Healthy	hyperparameter	89%	
		5	Red Rust	s of	75%	
		5	Whitefly	InceptionV3	83%	
					85% (overall)	

5. Discussion and Future Studies

The best pretrained was trained for only 10 epochs during the last session due to RAM and GPU limitation in free Colab usage. The 2048 features were extracted from Inception V3 model before proceeding with hyperparameters tuning. This approach helped alleviate the memory usage problem when compared to directly loading images for intensive tuning, which often caused the free version of Colab RAM to run out.

After obtaining the best hyperparameters through the tuning session, we employed them in another model training session to create a saved trained model, which could then be converted to tflite format. The tflite file was subsequently integrated into Android Studio for App development. There explains the variation in training and validation accuracies in both tuning and model training sessions.

The reason we couldn't directly save the best model from the tuning session into tflite format was due to our inability to utilize Python within Android Studio. If the tflite format was generated from the tuning session, we would need to load InceptionV3 model to extract the 2048 features again for new images captured on the phone before classifying them using saved tflite model obtained from the tuning session.

To boost the variety of the guava leaf disease recognition, we will acquire more photos of leaf disease from various species in the future. With the use of this extended dataset, CNN algorithm can categorise a wider range of leaf diseases and species. To reach the highest level of accuracy, the CNN model will undergo tuning, involving the modification of hyperparameters such as the learning rate, number of epochs, and optimizer types using optimization techniques like genetic algorithm or particle swarm optimization. We will select different pretrained models for performance comparison with Inception V3 in future. In future studies, an ensemble of different pretrained models will be incorporated.

6. Conclusions

In In this study, a total of 120 Guava leaf images with 4 different classes were collected and used to train the CNN model. The dataset was split into 60% for training and 40% for validation. To increase the training dataset, image augmented techniques were applied, such as rescale, random shear, random zoom, random horizontal flipping, and increased brightness.

The CNN base model used in this study is InceptionV3 because it achieved the best accuracy among the other 25 pre-trained models. The optimal learning rates, batch sizes, optimizer, and epoch counts were determined using an optimization script developed with InceptionV3 as a base model. To increase tuning efficiency, the images dataset was processed to extract 2048 output features from the second last layer of InceptionV3 and these features were saved in Numpy format.

The tuning results show the training data obtained a validation accuracy of 0.99 utilizing a batch size of 4, a learning rate of 0.001, and the Adamax optimizer for 50 epochs. When the model was

trained using these hyperparameters, a training accuracy of 0.98 and a validation accuracy of 0.9 were achieved. The difference in validation accuracy of 0.09 between the model training and tuning session can be attributed to discrepancies in the feature extraction technique and the use of sequential layers in both scripts. During the hyperparameter tuning session, feature extraction was done up to the second last layer of InceptionV3 to improve tuning effectiveness. On the other hand, in the model training session, the image dataset was inputted throughout the model, starting from the input layer of the base model to the final dense layer. These variations might lead to slightly different validation accuracies between the two sessions.

The trained model was integrated into the development of an Android app using Android Studio. The completed app was then exported as an APK file and installed on a smartphone device. The app performs excellently in terms of classifying guava leaf diseases and achieved an overall F1-Score of 0.85, a recall of 0.85, a precision of 0.89, and an accuracy of 0.85.

Acknowledgement
This research was supported by Universiti Tun Hussein Onn Malaysia (UTHM) through Tier 1 (Vote Q489).

References

[1] Neo, P. “Palm oil dilemma: Why Malaysia’s second-largest commodity export continues to face uncertain future,” *FoodNavigator Asia*, (2020). https://www.foodnavigator-asia.com/Article/2020/02/05/Palm-oil-dilemma-Why-Malaysia-s-second-largest-commodity-export-continues-to-face-uncertain-future

[2] Kamal, Md Mostafa, Md Akhtaruzzaman, Tajnuba Sharmin, Mahfuzur Rahman, and Shakti Chandra Mondal. "Optimization of extraction parameters for pectin from guava pomace using response surface methodology." *Journal of Agriculture and Food Research* 11 (2023): 100530. https://doi.org/10.1016/j.jafr.2023.100530

[3] Anita, John Nisha, and Sujatha Kumaran. "Detection and Segmentation of Meningioma Tumors Using the Proposed MENCNN Model." *Journal of Advanced Research in Applied Sciences and Engineering Technology* 32, no. 2 (2023): 1-13. https://doi.org/10.37934/araset.32.2.113

[4] Neeharika, Chitta Hrudaya, and Yeklur Mohammed Riyazuddin. "Developing an Artificial Intelligence Based Model for Autism Spectrum Disorder Detection in Children." *Journal of Advanced Research in Applied Sciences and Engineering Technology* 32, no. 1 (2023): 57-72. https://doi.org/10.37934/araset.32.1.5772

[5] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks." *Communications of the ACM* 60, no. 6 (2017): 84-90. https://doi.org/10.1145/3065386

[6] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1-9. 2015. https://doi.org/10.1109/CVPR.2015.7298594

[7] Huang, Gao, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. "Densely connected convolutional networks." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700-4708. 2017. https://doi.org/10.1109/CVPR.2017.243

[8] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778. 2016. https://doi.org/10.1109/CVPR.2016.90

[9] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

[10] Iandola, Forrest N., Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size." *arXiv preprint arXiv:1602.07360* (2016).

[11] Howard, Andrew G., Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." *arXiv preprint arXiv:1704.04861* (2017).

[12] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." In *International conference on machine learning*, pp. 448-456. pmlr, 2015.

[13] Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the inception architecture for computer vision." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818-2826. 2016. https://doi.org/10.1109/CVPR.2016.308
[14] Harakannanavar, Sunil S., Jayashri M. Rudagi, Veena I. Puranikmath, Ayesha Siddiqua, and R. Pramodhini. "Plant leaf disease detection using computer vision and machine learning algorithms." *Global Transitions Proceedings* 3, no. 1 (2022): 305-310. https://doi.org/10.1016/j.gltp.2022.03.016
[15] Anitha, P., C. Dhanushya, and Carol Henna KJ. "Cnn based early identification of plant disease using leaf images." In *2021 5th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pp. 780-785. IEEE, 2021.
[16] Falaschetti, Laura, Lorenzo Manoni, Denis Di Leo, Danilo Pau, Valeria Tomaselli, and Claudio Turchetti. "A CNN-based image detector for plant leaf diseases classification." *HardwareX* 12 (2022): e00363. https://doi.org/10.1016/j.ohx.2022.e00363
[17] Naik, B. Nageswararao, R. Malmathanraj, and Ponnusamy Palanisamy. "Detection and classification of chilli leaf disease using a squeeze-and-excitation-based CNN model." *Ecological Informatics* 69 (2022): 101663. https://doi.org/10.1016/j.ecoinf.2022.101663
[18] Russel, Newlin Shebiah, and Arivazhagan Selvaraj. "Leaf species and disease classification using multiscale parallel deep CNN architecture." *Neural Computing and Applications* 34, no. 21 (2022): 19217-19237. https://doi.org/10.1007/s00521-022-07521-w
[19] Prabu, M., and Balika J. Chelliah. "Mango leaf disease identification and classification using a CNN architecture optimized by crossover-based levy flight distribution algorithm." *Neural Computing and Applications* 34, no. 9 (2022): 7311-7324. https://doi.org/10.1007/s00521-021-06726-9
[20] Gaikwad, Sukanya S., Shivanand S. Rumma, and Mallikarjun Hangarge. "Fungi affected fruit leaf disease classification using deep CNN architecture." *International Journal of Information Technology* 14, no. 7 (2022): 3815-3824. https://doi.org/10.1007/s41870-022-00860-w
[21] Nandhini, S., and K. Ashokkumar. "Improved crossover based monarch butterfly optimization for tomato leaf disease classification using convolutional neural network." *Multimedia Tools and Applications* 80, no. 12 (2021): 18583-18610. https://doi.org/10.1007/s11042-021-10599-4
[22] Mustafa, Hassan, Muhammad Umer, Umair Hafeez, Ahmad Hameed, Ahmed Sohaib, Saleem Ullah, and Hamza Ahmad Madni. "Pepper bell leaf disease detection and classification using optimized convolutional neural network." *Multimedia Tools and Applications* 82, no. 8 (2023): 12065-12080. https://doi.org/10.1007/s11042-022-13737-8
[23] Uğuz, Sinan, and Nese Uysal. "Classification of olive leaf diseases using deep convolutional neural networks." *Neural computing and applications* 33, no. 9 (2021): 4133-4149. https://doi.org/10.1007/s00521-020-05235-5
[24] Thangaraj, Rajasekaran, M. Mohan, M. Moulik, A. Logeshwari, T. Loganathan, and P. Koushika. "A Comparative Study of Deep Learning Models for Guava Leaf Disease Detection." In *2023 Third International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, pp. 1-5. IEEE, 2023. https://doi.org/10.1109/ICAECT57570.2023.10117860
[25] Perumal, P. "Guava leaf disease classification using support vector machine." *Turkish Journal of Computer and Mathematics Education (TURCOMAT)* 12, no. 7 (2021): 1177-1183.
[26] Thilagavathi, M., and S. Abirami. "Application of image processing in diagnosing guava leaf diseases." *International Journal of Scientific Research and Management (IJSRM)* 5, no. 07 (2017): 5927-5933. https://doi.org/10.18535/ijsrm/v5i7.19
[27] Lai, Ren Yu, Kim Gaik Tay, Audrey Huong, Chang Choon Chew, and Shuhaida Ismail. "Dorsal hand Vein Authentication System Using Convolution Neural Network."
[28] Laghari, Waheed Ali, Tay Kim Gaik, Audrey Huong, Yaan Yee Choy, and Chang Choon Chew. "Dorsal hand vein identification using transfer learning from AlexNet." *International Journal of Integrated Engineering* 14, no. 3 (2022): 111-119. https://doi.org/10.30880/ijie.2022.14.03.012
[29] Lim, Tze Sheng, Kim Gaik Tay, Audrey Huong, and Xiang Yang Lim. "Breast cancer diagnosis system using hybrid support vector machine-artificial neural network." *Int. J. Electr. Comput. Eng.(IJECE)* 11, no. 4 (2021): 3059. https://doi.org/10.11591/ijece.v11i4.pp3059-3069
[30] Ooi, Min Ning, Kim Gaik Tay, and Chang Choon Chew. "A Safe Box Safety System with Facial Authentication for Multiple Users." In *International Conference on Computational Science and Technology*, pp. 181-191. Singapore: Springer Nature Singapore, 2022. https://doi.org/10.1007/978-981-19-8406-8_13

Name of Author	Email
Choon Fai Foo	foochoonfai1888@gmail.com

Kim Gaik Tay	tay@uthm.edu.my
Osamah Al-qershi[2]	o.alqersgi@unimelb.edu.my
Audrey Huong[1]	audrey@uthm.edu.my
Chang Choon Chew	chewcc@uthm.edu.my
Shehab Abdulhabib Alzaeemi	shehab@uthm.edu.my