ISSN 2278-3091
S
Volume 9, No.3, May - June 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse53932020.pdf
https://doi.org/10.30534/ijatcse/2020/53932020

A Comparative Study of Pretrained Convolutional Neural Network Model to Identify Plant Diseases on Android Mobile Device

Sembada Denrineksa Bimorogo[1], Gede Putra Kusuma[2]
[1]Computer Science Department, BINUS Graduate Program - Master of Computer Science,
Bina Nusantara University, Jakarta, Indonesia, 11480, sembada.bimorogo@binus.ac.id
[2]Computer Science Department, BINUS Graduate Program - Master of Computer Science,
Bina Nusantara University, Jakarta, Indonesia, 11480, inegara@binus.edu

ABSTRACT

Optimization of mobile devices to be used as early identification tools of plant diseases using an application based on Convolutional Neural Network (CNN), with a high degree of accuracy and low power consumption is the focus of this study. The study was conducted using a dataset consisting of 38 different classes of PlantVillage dataset, which were then expanded using 2 classes of coffee plants and 4 classes of rice plants. The models that are going to be tested and compared consists of MobileNet V2, NasNet Mobile, DenseNet 121 layer, and InceptionV3. In the experiments, it was found that there was a decrease in accuracy when the application was run on a mobile device when compared to when it was run on a PC. Experiments also show that InceptionV3 is the most stable model and reaches the highest level of accuracy, which is 98.45% on mobile devices. However, InceptionV3 consumes a lot of resources when used on mobile devices. Meanwhile, MobileNet V2, NasNet Mobile, dan DenseNet 121, do not consume a lot of resources when tested on mobile devices. In terms of accuracy, NasNet Mobile reached 97.31%, then MobileNet V2 reached 96.55%, and DenseNet 121 reached 96.21%. Based on the research criteria, it can be concluded that the CNN model that is most suitable to be used on mobile devices is NasNet Mobile. Because it has a high degree of accuracy with low resource consumption.

Key words: Plant Disease Identification, PlantVillage Dataset, Deep Learning, Convolutional Neural Networks, Mobile Application.

1. INTRODUCTION

Plants are a classification of living beings that have a very important role, especially for humans. However, dangerous diseases often found on plants that could cause a decrease of quality and quantity from the produced product. Detecting and preventing the disease immediately is very important to be able to solve the problem before it becomes a more severe and widespread stage. A quick and accurate diagnosis of disease severity will help to reduce the loss of yield [1].

There are various parts of plants that attacked by diseases, such as leaves, stems, fruits and others. It would be difficult if the disease qualification was done using the naked eye. Therefore, an understanding of specific and accurate picture patterns is now in demands. And here the role of image processing plays an important role[2].

Though the process of classification and identification of plant diseases can be done using image processing at a small cost, the use of technology as a tool to detect plant diseases has not been used to the fullest. This leads to financial losses and profit reduction to farmers[3].

A mechanism is needed to be able to detect disease on leaf images, which are expected to provide accurate disease information from a plant. The information that referred to is the relevance between the diseases on picture of plant with the actual plant's disease. One mechanism that able to detect diseases in plants quickly and accurately is by using a machine learning algorithm.

The main area of research in machine learning is that a computer program should be learning automatically to recognize complex patterns and makes it's own intelligent decisions based on data [4]. The machine learning method used in this study is image classification. One algorithm that is currently popular to solve image classification problems is Convolutional Neural Network (CNN) [5, 6] which is included in the deep learning algorithm.

There have been a lot of studies that use deep learning to identify diseases in plants, as has been done by Zhang et al. [7] by increasing two deep convolutional neural networks models, each of which achieved an accuracy of 98.9% and 98.8%. In addition, Ferentinos [8] has achieved identification accuracy of 98.53% and 98.49% from the two models of convolutional neural networks (VGG and AlexNetOWTBn) using a database of 87,848 photographs which taken under laboratory conditions and real conditions in cultivation. Then Too et al. [9] used several deep convolutional neural network architectural models including VGG 16, Inception V4, ResNet with 50, 101 and 152 layers and DenseNet with 121 layers for image-based plant disease classification. Brahi,I et al. [10] classify plant diseases and achieve the highest

accuracy of 99.76%, using the Inception V3 model of several different CNN architectures using the public PlantVillage dataset. Then the results of experiments that have been carried out by Jadhav et al. [11], GoogleNet and VGG16 reached an accuracy rate of 96.4%, followed by AlexNet with an accuracy rate of 95%, then DenseNet at 93.6%, and finally ResNet101 at 92.1%.

Based on previous research, there has not been a research that focus on the application of deep learning techniques to be used on mobile devices, especially the research of plant disease classification.

To optimize the use of mobile devices, this study will evaluate CNN models that have a high degree of accuracy and are created specifically to run on mobile devices. The conclusion is to get the most effective model which have a high degree of accuracy, low latency, and relatively small resource consumption.

Based on the problem, the authors used several architectural models such as MobileNet V2, NasNetMobile, Inception V3 and DenseNet with 121 layers to be compared. Then the model will be compared again after being applied to mobile device application with the aim of evaluating and finding a suitable model for mobile devices.

2. RELATED WORKS

In their study [7], they used two models of improved deep convolutional neural networks (GoogLeNet and Cifar10) that achieved high identification accuracy of 98.9% and 98.8% respectively. Experiments in this journal show that it is possible to improve recognition accuracy by increasing the diversity of merge operations, a reasonable addition of the Relu function and dropout operations, and including some adjustments of the model parameters.

In research that has been done by Ferentinos [8], two CNN models (VGG and AlexNetOWTBn) achieved identification accuracy of 98.53% and 98.49% respectively. Model training was carried out using a dataset of 87,848 photographs which taken under laboratory conditions and conditions evident in the field of cultivation. Data in this journal consists of 25 plant species in 58 different classes using combinations [plants, diseases], including several healthy plants. The architecture of the VGG model, a convolutional neural network, achieved a success rate of 99.53% (top error -1 by 0.47%) in the classification of 17.548 plant leaf images which had not been previously trained on the model.

In their research [9], they used several deep convolutional neural network architectural models that were evaluated including VGG 16, Inception V4, ResNet with 50,101 and 152 layers and DenseNet with 121 layers for image-based plant disease classification. DenseNets obtained a test accuracy score of 99.75%. However, despite its good architectural performance, further research needs to be done to improve computing time.

Brahimi et al. [10] in their research focused on the superiority of the deep learning technique of visualization method in providing transparent information, so that an explanation and details of the classification mechanism could be obtained. In their research, performance of various variations and types of CNN learning architecture were compared based on the time required in the learning process. The results of the comparison will be useful for future researchers to choose the best CNN architecture in building a practical system for the detection of plant diseases. The evaluation process was done using the PlantVillage dataset. In process of training and evaluating CNN's state-of-art performance, the open Caffe framework, the deep learning framework developed by Berkley Vision and Learning Center would be used. In this study, the highest accuracy was achieved using the inception_v3 model with a value of 99.76% and a time of 5.64 hours, while the least time was achieved using shallow type learning in the Squeeznet model of 0.85 hours with an accuracy of 96.26%.

In the research conducted by Jadhav et al. [11], several CNN models were implemented using 1,200 data of sick and healthy soybean leaves. Classification is carried out on the proposed model by modifying various hyperparameters, such as minibatch size, max epoch, and learning rate bias. The results of experiments showed that GoogleNet and VGG16 reached an accuracy level of 96.4%, followed by AlexNet with an accuracy rate of 95%, then DenseNet at 93.6%, and finally ResNet101 at 92.1%

3. RESEARCH METHODOLOGY

The methodology in this study mainly involve two stages, which is retraining stage and testing stage. The retraining stage is carried out by retraining the pretrained CNN model. The testing page is carried out by testing the model on both PCs and mobile device. Illustration depict in Figure 1 shows the sequence of work processes in the study.

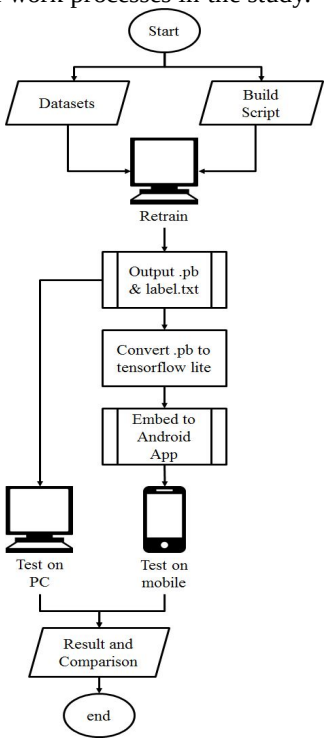

Figure 1: Proposed Method

The retraining stage begin with data preparation, then collecting image data and finally applying an in-depth learning to learning transfer method. The next stage, which is testing stage, consist of testing process of the output file from retraining stage.

3.1 Retrain Stage

The design of system in retraining stage were illustrated in a flowchart which can be seen in Figure 2.

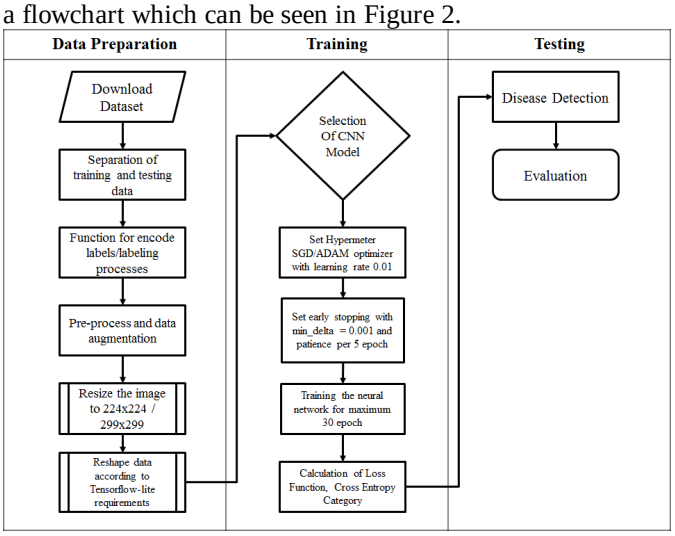

Figure 2: Retrain stage process

The CNN training process is carried out using the TensorFlow framework, the learning model in Python 3.6, TensorFlow-lite, Android Studio, and PyCharm community edition. The experiments were carried out on a PC which has the specifications that illustrated in Table 1.

Table 1: Specifications and Configuration on PC

No	Hardware and software	Characteristics
1	Memory	16 GB
2	Processor (CPU)	Intel Core i3-7100 CPU @ 3.90 GHz (4 CPUs)
3	Graphics (GPU)	NVIDIA GeForce GTX 1050 Ti 12 GB
4	Operating system Linux	Debian GNU/Linux 9.11 (stretch)

3.1.1 Dataset Preparation

There are several sources of datasets used in this study. These include [12], then the coffee plant dataset that used in the study [13] and finally the rice plant leaf dataset [14] which was also combined with the rice leaf dataset by Aldrin Kein G. Francisco [15]. In addition to these sources, researchers also added several pictures of rice leaf leaves obtained by taking portraits of several rice leaf samples as well as downloading from several agricultural sites that list rice leaf diseases. Total images that collected were 57551 images consisting of 18 plant species in 44 different classes with 30 combinations of diseased plants and 14 healthy plants. The dataset will be divided into 3 parts, consisting of 60% of training data, 20% of validation data, and 20% of test data, thereby producing 34531 training data, 11500 validation data, and 11520 test data summarized in Table 2. The system used to carry out the testing process on mobile devices will select images one by one to be tested. Because testing would be done one by one, then in this study 5% of the entire testing dataset, i.e. as many as 580 images will be separated by class for testing on PCs and mobile devices.

Table 2: PlantVillage dataset, which in addition two coffee leaf classes and four rice leaf classes were included.

No	Class	Training	Validation	Testing
1	Apple-Apple-scab	378	126	126
2	Apple-Cedar-apple-rust	165	55	55
3	Apple-Frogeye-Spot	373	124	124
4	Apple-healthy	987	329	329
5	Blueberry-healthy	901	300	301
6	Cherry-including-sour- healthy	512	171	171
7	Cherry-including-sour -Powdery-mildew	631	210	211
8	Coffee-healthy	115	38	39
9	Coffee-Leaf-rust	682	227	228
10	Corn-maize-Cercospora-leaf- spot-Gray-leaf-spot	308	102	103
11	Corn-maize-Common-rust	715	238	239
12	Corn-maize-healthy	697	232	233
13	Corn-maize-Northern-Leaf- Blight	591	197	197
14	Grape-Black-rot	708	236	236
15	Grape-Esca-Black-Measles	830	276	277
16	Grape-healthy	254	84	85
17	Grape-Leaf-blight-Isariopsis-Leaf-Spot	646	215	215
18	Orange-Haunglongbing- Citrus-greening	3304	1101	1102
19	Peach-Bacterial-spot	1378	459	460
20	Peach-healthy	216	72	72
21	Pepper-bell-Bacterial-spot	598	199	200
22	Pepper-bell-healthy	887	295	296
23	Potato-Early-blight	600	200	200
24	Potato-healthy	91	30	31
25	Potato-Late-blight	600	200	200
26	Raspberry-healthy	223	74	74
27	Rice-Bacterial-leaf-blight	74	24	25
28	Rice-Brown-spot	240	80	80
29	Rice-healthy	631	210	210
30	Rice-Leaf-blast	206	68	69
31	Soybean-healthy	3054	1018	1018
32	Squash-Powdery-mildew	1101	367	367
33	Strawberry-healthy	274	91	91
34	Strawberry-Leaf-scorch	665	222	222
35	Tomato-Bacterial-spot	1276	425	426
36	Tomato-Early-blight	600	200	200
37	Tomato-healthy	955	318	318
38	Tomato-Late-blight	1145	382	382
39	Tomato-Leaf-Mold	571	190	191
40	Tomato-Septoria-leaf-spot	1063	354	354
41	Tomato-Spider-mites-Two- spotted-spider-mite	1006	335	335
42	Tomato-Target-Spot	842	281	281
43	Tomato-Tomato-mosaic-virus	224	74	75
44	Tomato-Tomato-Yellow-Leaf-Curl-Virus	3214	1071	1072
Total		**34531**	**11500**	**11520**

3.1.2 CNN Architecture

Generally, CNN model has similar architecture which can be illustrated in Figure 3. The architecture starts with input in images form and then followed by convolutional operations in convolutional layer, pooling operations in pooling layer and fully connected layer

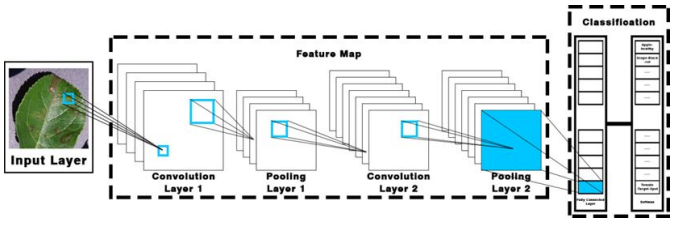

Figure 3: Convolution Neural Network (CNN) architecture

The pre-trained model will be stored in the cloud, this is done so that the model can be reused and downloaded from the framework provider.

- **MobileNet V2**

MobileNet V2 is a neural network developed from its predecessor MobileNet V1. MobileNet V2 has a significant increase from MobileNet V1, and encourages the progress of the state-of-the-art visual recognition on mobile devices. In MobileNet V2 there are two types of blocks, one block is a residual block with stride 1, and the other block is a block with stride 2 and used for downsizing. There are three different layers for the two block types.

Overall, the model produced by MobileNet V2 has a faster runtime for the same accuracy on all spectrum when compared to MobileNetV1 [16]. In addition, MobileNetV2 was able to show better accuracy with the use of fewer parameters when compared to MobileNetV1.

- **NasNet Mobile**

NASNet is a machine learning model that proposes to look for architecture of blocks in small datasets, which then said blocks will be transferred to large datasets [17]. In its application to ImageNet image classification, NASNet was able to obtain prediction accuracy of 82.7% in the validation dataset [18]. In that study, the NASNet model obtained results that achieved state-of-the-art, but the results were obtained with a smaller model size and a lighter level of complexity.

The architectural blocks or cells that are sought on NASNet can then be arranged in such a way as to produce a variation from the NASNet architecture, where smaller variations are called NASNetMobile or NASNet-A (4 @ 1056). NASNetMobile has a number of parameters that resemble MobileNet, but is able to provide better accuracy performance. NASNet architecture itself consists of 2 different cells, namely normal cells and reduction cells. Furthermore, the exact layer arrangement for the two layers is searched by using recurent neural network.

- **InceptionV3**

Inception V3 is the third edition of Google's Inception Convolutional Neural Network that was developed to resolve ImageNet large Visual Recognition Challenge. Inception itself is a machine learning model that was first introduced in the GoogLeNet architecture by [19]. The purpose of the inception model is to act as a multi-level feature extractor by counting convolution filters in the same module. The results of the filters are then stacked into the channel dimensions before inserted into the next layer.

- **DenseNet**

DenseNet is a machine learning model architecture which has a condition where every layer will be connected to all layers directly. Layers on DenseNet will get input which is the output of all layers that were passed before. In addition, layers on DenseNet will produce output that will be used on all layers which will then be traversed. This allows the networkk formed to be more streamlined. The process on DenseNet is different from the traditional Convolutional layer, where a layer takes input from the previous layer and provides output for the next layer [20]. From the several variants, one of the smallest DenseNet types to be used on ImageNet is DenseNet121. DenseNet121 is a variation of the DenseNet model that has a small number of parameters which is similar to MobileNet.

3.2 Testing Stage

The process of the testing stage on PC and mobile device illustrated in Figure 4

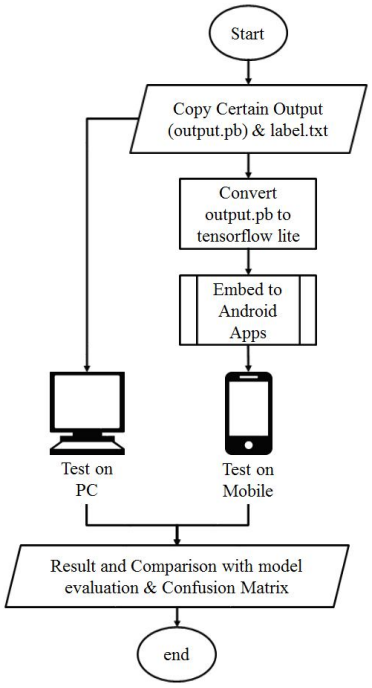

Figure 4: Testing stage process

In the testing stage, a comparison between model testing on a PC device and model testing on a mobile device will be carried out. To conduct testing on a mobile device, it is necessary to carry out a conversion process so that the file can be optimized to run on the mobile device. Several file that produced from the training process, such as "output.pb" was converted to " output.lite". The conversion process was carried out using "TensorFlow Lite Optimizing Converter" program (tflite_convert) which was already included in the installation of TensorFlow.

Serialization format used in TensorFlow Lite (TFLite) is different from the serialization format used in TensorFlow, where FlatBuffers were used in TFLite, meanwhile Buffer protocol were used in Tensorflow. The advantage of using FlatBuffers is that the data can be mapped into memory and the used directly from disk without having to be loaded and parsed before. This results in a much faster startup time, as well as enabled the operating system to avoid running out of memory by giving them the choice which required pages will be loaded and unloaded from the model file. After the conversion is done, the file then copied into the assets folder on the mobile device .

For evaluation process on mobile devices, android based mobile applications were built using application samples from Tensorflow and then modifying it, as illustrated in Figure 5.

Figure 5: Test application on mobile device

The deployment process on mobile devices will be carried out on devices with the specifications described in Table 3.

Table 3: Mobile device specifications

No	Hardware and software	Characteristics
1	Memory	4 GB
2	Processor (CPU)	Octa-core, 4 x ARM Cortex-A73 2 GHz + 4 x ARM Cortex-A53 2 GHz
3	Operating system Linux	Android 9 Pie;ColorOS 6
4	Rear Camera	16 MP, f/1.8, phase detection autofocus, LED flash

4. EXPERIMENT

4.1 Experimental Design

The experiment was separated into two main stages, which is retraining stage and testing stage. In retraining stage, researchers would calculate the evaluation and comparison of time, accuracy and loss used by the pretrained architecture. The output produced by this stage is in the form of a output file (.pb), which then converted into a tflite (.lite) file to be used in the evaluation phase on mobile devices.

The next step is to evaluate testing data using PC and performance testing using mobile device. The two devices used will then be measured based on the resources needed, such as the size of memory used and the use of energy resources (batteries). The results of these measurements will then be used in determining the minimum requirements for hardware specifications on mobile devices, as well as the most effective model to be used on mobile devices.

In this experiment, environment setting used is PC or workstation which equipped with a tool used to do deep learning processes, python 3.6, OpenCV, TensorFlow, Keras and NumPy module. While on a mobile device, an Android-based application will be built using Android Studio. Output generated from the application is an explanation of the time required to process an image in millisecond (ms), predictions of certainty presentation, predicted labels, use of batteries in microampere (µA) and battery usage in kilobytes (KB).

Same separate images were used to conduct comparative testing between the performance on PC and mobile device. This was conducted with the aim to measure the value of accuracy, latency and resource consumption of mobile devices.

The testing process on PC was started by executing the output.pb file using a script that has been built specifically to carry out the testing process. Meanwhile, the testing process on mobile devices is done by using a plant disease classification application by manually checking the images one by one. The results of the testing process will then be sent to an online matrix calculator [21]. Before the training process on a mobile device is carried out, power charging will be carried out on the device until the battery is full and then the device would be restarted. That was done in an effort so the device would have the best initial conditions and to ensure there are no unnecessary applications that run at the same time.

4.2 Experimental Result

4.2.1 Retraining on PC

The first experiment was done by running a script program in Jupyter notebook format that is distinguished on each model. With some special conditions, where it will run with a maximum of 30 epochs, using the early stopping method by applying min_delta of 0.001 and patience per 5 epochs, using the same hyperparameter, using learning rate of 0.01, as well as a dataset that has been separated between train set and validation set. The performance of each model trained is illustrated graphically in Figure 6 for performance of accuracy and Figure 7 for performance of cross entropy.

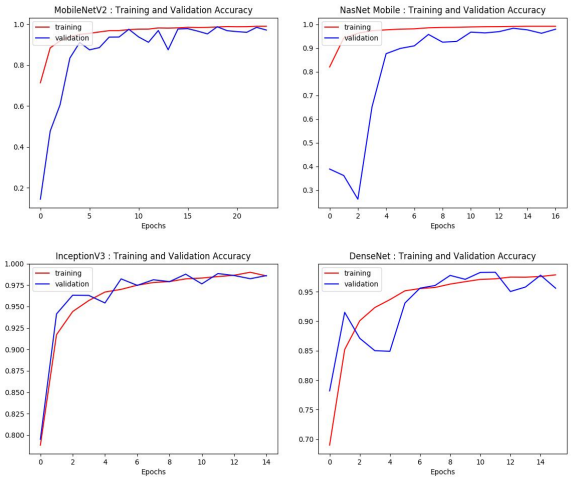

Figure 6: Accuracy performance on each model

Figure 6 shows that the retention process of MobileNet V2 stops at epoch 24 with an accuracy value of 0.9890 or 98.90% and a validation accuracy value of 0.9706 or 97.06%. While the retaining process of NasNet Mobile stops at epoch 17 with

an accuracy value of 0. 9920 or 99.20% with the validation accuracy value is 97.98 or 97.98%.

Then the retaining process in the Inception V3 model stops at the epoch 15 with an accuracy value of 0.9857 or 98.57% and the validation accuracy value of 0.9860 or 98.6%. The end of the DenseNet 121 retaining process stops at epoch 16 with an accuracy value of 0.9784 or 97.84%, with a validation accuracy of 0.9558 or 95.58%.

While in Figure 7, the NasNet Mobile Loss value is the lowest with a loss accuracy in the epoch 17 is 0.0277, with a validation loss value of 0.0011 compared to NasNet Mobile and MobileNet V2.

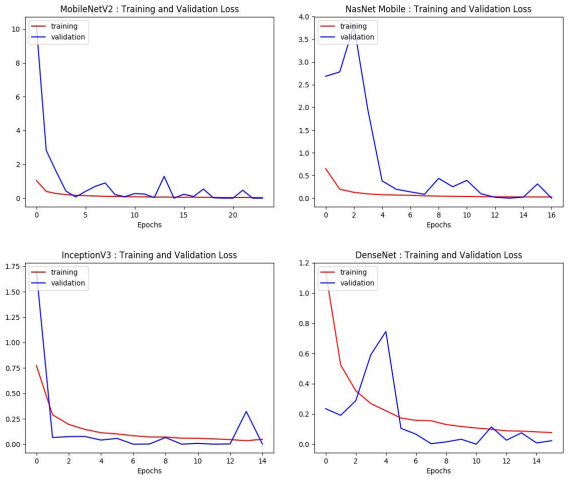

Figure 7: Categorical-cross entropy loss on each model

The second experiment tested empirically using native python scripts to produce accuracy values, a measure of the size of the output file in the form of a protobuf file and also the time required for a cycle of retraining or runtime retraining. The experimental results are summarized in Table 4.

The final evaluation on the PC retraining experiment shows that although the Inception V3 model runs only up to 15 epochs, the test results on Inception V3 get the highest accuracy level of 98.68% followed by NasNet Mobile with 98.23%, then MobileNet V2 with 97.31% and finally DenseNet 121 that reaches the level accuracy of 96.94%. While validation losses were recorded on NasNet Mobile, MobileNet V2, Inception V3 and DenseNet 121 respectively from lowest to highest. In the .pb file output size, MobileNet V2 generated 21.2 MB, while NasNet Mobile generated 39.9 MB, then Inception V3 generated 171 MB and finally DenseNet 121 generated 59.5MB. Training time on the DenseNet 121 model took longer than other models by taking as many as 15177 second per epoch.

4.2.2 Comparison test PC and mobile device

- **Accuracy and Latency on PC and mobile device**

An experimental comparison of performance between a PC and a mobile device was carried out using 580 images shown in Table 5 with overall accuracy and average latency.

Table 5: Comparison report of accuracy and latency on PC and mobile device

Model	Accuracy (%)		Average Latency (ms)	
	PC	Mobile	PC	Mobile
MobileNet V2	97.07%	96.55%	47	104
NasNet Mobile	98.28%	97.31%	64	243
InceptionV3	98.62%	98.45%	105	773
DenseNet 121	96.55%	96.21%	121	474

The numbers in Table 5 show that the accuracy on mobile devices hit some loss, while Inception had a high latency rate. However, Inception V3 experienced the lowest loss with 0.17%, followed by DenseNet 121 with 0.34%, then MobileNet V2 with 0.52%, and finally Nasnet Mobile with 0.97%.

- **Confusion Matrix on PC**

Figure 8 through figure 11 are illustrating performance of classification output for each model and class that has been tested on PC. The performance illustrated using confusion matrix where Y axis represents the correct label and X axis represents the prediction label.

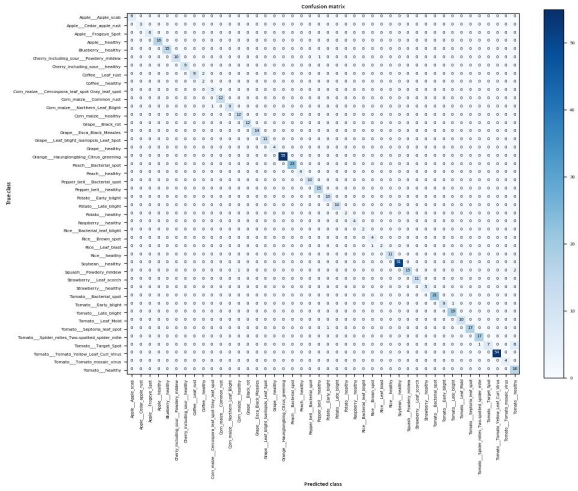

Figure 8: Confusion matrix for MobileNet V2 on PC

Table 4: Retraining result accuracy, loss, output, and time

Model	Epoch	Accuracy (%)			Loss		Output File (MB)		Time /Epoch (Second)
		Train	Valid	Test	Train	Valid	.pb	.lite	
MobileNet V2	24	98.91%	97.06%	97.31%	0.0378	0.0011	21.2	10.2	4971
NasNet Mobile	17	99.2%	97.98%	98.23%	0.0277	0.0011	39.9	18.17	9361
Inception V3	15	98.57%	98.61%	98.68%	0.0486	0.0012	171	89.25	10179
DenseNet 121	16	97.84%	95.58%	96.94%	0.0766	0.0232	59.5	28.96	15177

Figure 8 shows good performance in most classes, while in the Tomato Target Spot class, it shows a significant decrease in performance.

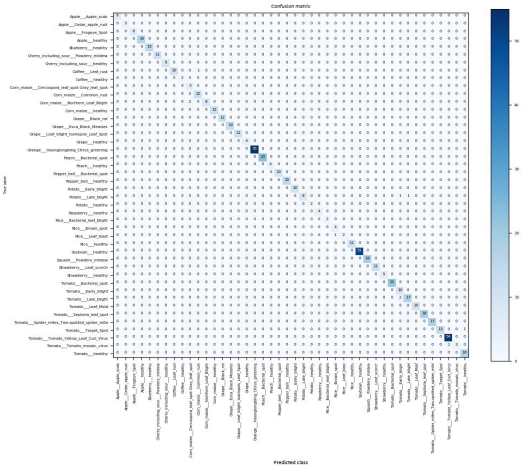

Figure 9: Confusion matrix for NasNet Mobile on PC

Figure 9 shows the Squash Powdery Mildew class and Tomato Target Spot class getting better performance on NasNet Mobile.

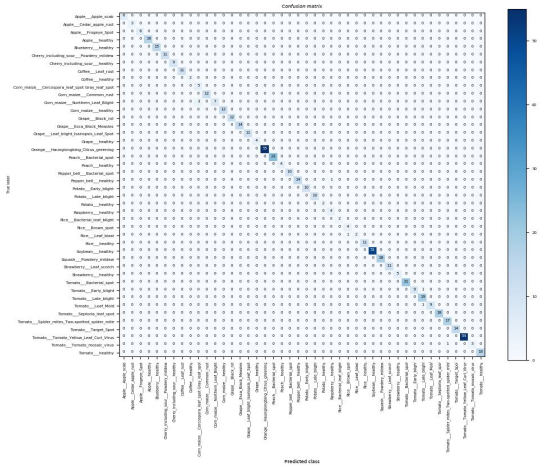

Figure 10: Confusion matrix for Inception V3 on PC

Figure 10 shows that the Tomato Target Spot and Potato Late Blight classes get better performance in Inception V3 than two previous models. However, the Corn Maize Northern Leaf Blight class, Pepper Bell Healthy class and Tomato Leaf Mold class show a slight decrease in performance.

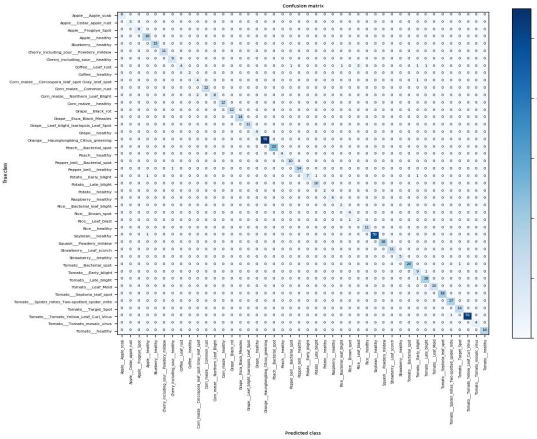

Figure 11: Confusion matrix for DenseNet 121 on PC

Figure 11 shows the underperformance of the other 3 models, such in the Coffee Leaf Rust class, the Potato Early blight spot class and the Tomato Yellow Leaf Curl Virus class.

- **Classification Performance Comparison**

Table 6 through table 9 are showing the result of experiments which carried out by testing and comparing of output files that generated from retrain process.

Table 6: Classification comparison report on Mobilenet V2

No	Class	Mobile			PC			Support
		Precision	Recall	F1 Score	Precision	Recall	F1 Score	
1	Apple___Apple_scab	1	1	1	1	1	1	6
2	Apple___Cedar_apple_rust	1	1	1	1	1	1	3
3	Apple___Frogeye_Spot	1	1	1	1	1	1	6
4	Apple___healthy	1	1	1	1	1	1	16
5	Blueberry___healthy	1	1	1	1	1	1	15
6	Cherry_including_sour___healthy	1	1	1	1	1	1	9
7	Cherry_including_sour___Powdery_mildew	1	0.91	0.95	1	0.91	0.95	11
8	Coffee___healthy	0.50	1	0.67	0.50	1	0.67	2
9	Coffee___Leaf_rust	1	0.82	0.9	1	0.82	0.9	11
10	Corn_maize___Cercospora_leaf_spot_Gray_leaf_spot	0.71	1	0.83	0.83	1	0.91	5
11	Corn_maize___Common_rust	1	1	1	1	1	1	12
12	Corn_maize___healthy	1	1	1	0.92	1	0.96	12
13	Corn_maize___Northern_Leaf_Blight	0.89	0.8	0.84	1	0.9	0.95	10
14	Grape___Black_rot	1	1	1	1	1	1	12
15	Grape___Esca_Black_Measles	1	1	1	1	1	1	14
16	Grape___healthy	1	1	1	1	1	1	4
17	Grape___Leaf_blight_Isariopsis_Leaf_Spot	1	1	1	1	1	1	11
18	Orange___Haunglongbing_Citrus_greening	1	1	1	1	1	1	55
19	Peach___Bacterial_spot	1	1	1	1	1	1	23
20	Peach___healthy	1	1	1	1	1	1	4
21	Pepper_bell___Bacterial_spot	1	1	1	1	1	1	10
22	Pepper_bell___healthy	0.94	1	0.97	0.94	1	0.97	15
23	Potato___Early_blight	0.91	1	0.95	0.91	1	0.95	10
24	Potato___healthy	1	1	1	1	1	1	2
25	Potato___Late_blight	1	0.9	0.95	1	1	1	10
26	Raspberry___healthy	1	1	1	1	1	1	4
27	Rice___Bacterial_leaf_blight	1	1	1	1	1	1	2
28	Rice___Brown_spot	0.8	1	0.89	0.8	1	0.89	4
29	Rice___healthy	1	1	1	1	1	1	11
30	Rice___Leaf_blast	1	0.67	0.8	1	0.67	0.8	3
31	Soybean___healthy	1	1	1	1	1	1	51
32	Squash___Powdery_mildew	1	0.83	0.91	1	0.83	0.91	18
33	Strawberry___healthy	1	1	1	1	1	1	5
34	Strawberry___Leaf_scorch	1	1	1	1	1	1	11
35	Tomato___Bacterial_spot	1	1	1	1	1	1	21
36	Tomato___Early_blight	1	0.9	0.95	1	0.9	0.95	10
37	Tomato___healthy	0.7	1	0.82	0.67	1	0.8	16
38	Tomato___Late_blight	0.9	0.95	0.92	0.95	1	0.97	19
39	Tomato___Leaf_Mold	0.91	1	0.95	1	1	1	10
40	Tomato___Septoria_leaf_spot	1	0.94	0.97	1	0.94	0.97	18
41	Tomato___Spider_mites_Two-spotted_spider_mite	0.94	1	0.97	0.94	1	0.97	17
42	Tomato___Target_Spot	1	0.50	0.67	1	0.50	0.67	14
43	Tomato___Tomato_mosaic_virus	1	1	1	1	1	1	4
44	Tomato___Tomato_Yellow_Leaf_Curl_Virus	0.98	1	0.99	1	1	1	54

From Table 6 it is know that some classes have decreased accuracy on mobile device, such as potatoes, some diseased corn and some diseased tomatoes. While healthy corn and healthy tomatoes experience better performance on mobile device.

Table 7: Classification comparison report on NasNet Mobile

No	Class	Mobile			PC			Support
		Precision	Recall	F1 Score	Precision	Recall	F1 Score	
1	Apple___Apple_scab	1	1	1	1	1	1	6
2	Apple___Cedar_apple_rust	1	1	1	1	1	1	3
3	Apple___Frogeye_Spot	1	1	1	1	1	1	6
4	Apple___healthy	1	1	1	1	1	1	16
5	Blueberry___healthy	1	1	1	1	1	1	15
6	Cherry_including_sour___healthy	1	1	1	1	1	1	9
7	Cherry_including_sour___Powdery_mildew	1	0.91	0.95	1	1	1	11
8	Coffee___healthy	1	1	1	1	1	1	2
9	Coffee___Leaf_rust	1	0.91	0.95	1	0.91	0.95	11
10	Corn_maize___Cercospora_leaf_spot_Gray_leaf_spot	0.71	1	0.83	0.71	1	0.83	5
11	Corn_maize___Common_rust	0.92	1	0.96	0.92	1	0.96	12
12	Corn_maize___healthy	1	1	1	1	1	1	12
13	Corn_maize___Northern_Leaf_Blight	1	0.8	0.89	1	0.8	0.89	10
14	Grape___Black_rot	1	1	1	1	1	1	12
15	Grape___Esca_Black_Measles	1	1	1	1	1	1	14
16	Grape___healthy	1	1	1	1	1	1	4
17	Grape___Leaf_blight_Isariopsis_Leaf_Spot	1	1	1	1	1	1	11
18	Orange___Haunglongbing_Citrus_greening	1	1	1	1	1	1	55
19	Peach___Bacterial_spot	1	1	1	1	1	1	23
20	Peach___healthy	1	1	1	1	1	1	4
21	Pepper_bell___Bacterial_spot	1	1	1	1	1	1	10
22	Pepper_bell___healthy	0.94	1	0.97	1	1	1	15
23	Potato___Early_blight	1	1	1	1	1	1	10

24	Potato___healthy	1	1	1	1	1	1	2
25	Potato___Late_blight	1	0.8	0.89	1	0.8	0.89	10
26	Raspberry___healthy	1	1	1	1	1	1	4
27	Rice___Bacterial_leaf_blight	1	1	1	1	1	1	2
28	Rice___Brown_spot	0.8	1	0.89	0.8	1	0.89	4
29	Rice___healthy	1	1	1	1	1	1	11
30	Rice___Leaf_blast	1	0.67	0.8	1	0.67	0.8	3
31	Soybean___healthy	1	1	1	1	1	1	51
32	Squash___Powdery_mildew	1	1	1	1	1	1	18
33	Strawberry___healthy	1	1	1	1	1	1	5
34	Strawberry___Leaf_scorch	1	1	1	1	1	1	11
35	Tomato___Bacterial_spot	1	1	1	1	1	1	21
36	Tomato___Early_blight	0.77	1	0.87	0.77	1	0.87	10
37	Tomato___healthy	0.89	1	0.94	0.94	1	0.97	16
38	Tomato___Late_blight	0.94	0.89	0.92	0.94	0.89	0.92	19
39	Tomato___Leaf_Mold	1	1	1	1	1	1	10
40	Tomato___Septoria_leaf_spot	1	1	1	1	1	1	18
41	Tomato___Spider_mites_Two-spotted_spider_mite	1	1	1	1	1	1	17
42	Tomato___Target_Spot	1	0.86	0.92	1	0.93	0.96	14
43	Tomato___Tomato_mosaic_virus	1	0.75	0.86	1	0.75	0.86	4
44	Tomato___Tomato_Yellow_Leaf_Curl_Virus	0.98	1	0.99	0.98	1	0.99	54

Table 7 shows that the results on NasNet Mobile experienced a slight decrease in accuracy in the Cherry class, including sour Powdery mildew, Pepper Bell Healthy, Tomato Healthy and Tomato Target Spot.

Table 8: Classification comparison report on Inception V3

No	Class	Mobile			PC			Support
		Precision	Recall	F1 Score	Precision	Recall	F1 Score	
1	Apple___Apple_scab	1	1	1	1	1	1	6
2	Apple___Cedar_apple_rust	1	1	1	1	1	1	3
3	Apple___Frogeye_Spot	1	1	1	1	1	1	6
4	Apple___healthy	1	1	1	1	1	1	16
5	Blueberry___healthy	1	1	1	1	1	1	15
6	Cherry_including_sour___healthy	1	1	1	1	1	1	9
7	Cherry_including_sour___Powdery_mildew	1	1	1	1	1	1	11
8	Coffee___healthy	1	1	1	1	1	1	2
9	Coffee___Leaf_rust	1	1	1	1	1	1	11
10	Corn_maize___Cercospora_leaf_spot_Gray_leaf_spot	0.63	1	0.77	0.62	1	0.77	5
11	Corn_maize___Common_rust	1	1	1	1	1	1	12
12	Corn_maize___healthy	1	1	1	1	1	1	12
13	Corn_maize___Northern_Leaf_Blight	1	0.7	0.82	1	0.7	0.82	10
14	Grape___Black_rot	1	1	1	1	1	1	12
15	Grape___Esca_Black_Measles	1	1	1	1	1	1	14
16	Grape___healthy	1	1	1	1	1	1	4
17	Grape___Leaf_blight_Isariopsis_Leaf_Spot	1	1	1	1	1	1	11
18	Orange___Haunglongbing_Citrus_greening	1	1	1	1	1	1	55
19	Peach___Bacterial_spot	1	1	1	1	1	1	23
20	Peach___healthy	1	1	1	1	1	1	4
21	Pepper_bell___Bacterial_spot	1	1	1	1	1	1	10
22	Pepper_bell___healthy	1	0.93	0.97	1	0.93	0.97	15
23	Potato___Early_blight	1	1	1	1	1	1	10
24	Potato___healthy	1	1	1	1	1	1	2
25	Potato___Late_blight	1	1	1	1	1	1	10
26	Raspberry___healthy	0.8	1	0.89	0.8	1	0.89	4
27	Rice___Bacterial_leaf_blight	1	1	1	1	1	1	2
28	Rice___Brown_spot	0.8	1	0.89	0.8	1	0.89	4
29	Rice___healthy	1	1	1	1	1	1	11
30	Rice___Leaf_blast	1	0.67	0.8	1	0.67	0.8	3
31	Soybean___healthy	1	1	1	1	1	1	51
32	Squash___Powdery_mildew	1	0.94	0.97	1	1	1	18
33	Strawberry___healthy	1	1	1	1	1	1	5
34	Strawberry___Leaf_scorch	1	1	1	1	1	1	11
35	Tomato___Bacterial_spot	1	1	1	1	1	1	21
36	Tomato___Early_blight	1	0.9	0.95	1	0.9	0.95	10
37	Tomato___healthy	1	1	1	1	1	1	16
38	Tomato___Late_blight	0.9	1	0.95	0.9	1	0.95	19
39	Tomato___Leaf_Mold	1	0.9	0.95	1	0.9	0.95	10
40	Tomato___Septoria_leaf_spot	1	1	1	1	1	1	18
41	Tomato___Spider_mites_Two-spotted_spider_mite	0.94	1	0.97	0.94	1	0.97	17
42	Tomato___Target_Spot	1	1	1	1	1	1	14
43	Tomato___Tomato_mosaic_virus	1	0.75	0.86	1	0.75	0.86	4
44	Tomato___Tomato_Yellow_Leaf_Curl_Virus	0.98	1	0.99	1	1	1	54

Table 8 shows that Inception V3 has decreased accuracy in Squash Powdery mildew and Tomato Yellow Leaf Curl Virus. However, the average performance of InceptionV3 on mobile and PC devices has the same performance in each class.

Table 9: Classification comparison report on DenseNet 121

No	Class	Mobile			PC			Support
		Precision	Recall	F1 Score	Precision	Recall	F1 Score	
1	Apple___Apple_scab	1	1	1	1	1	1	6
2	Apple___Cedar_apple_rust	1	1	1	1	1	1	3
3	Apple___Frogeye_Spot	1	1	1	1	1	1	6
4	Apple___healthy	0.84	1	0.91	0.89	1	0.94	16
5	Blueberry___healthy	1	1	1	1	1	1	15
6	Cherry_including_sour___healthy	0.9	1	0.95	0.9	1	0.95	9
7	Cherry_including_sour___Powdery_mildew	1	0.91	0.95	1	1	1	11
8	Coffee___healthy	1	1	1	1	1	1	2
9	Coffee___Leaf_rust	1	0.36	0.53	1	0.36	0.53	11
10	Corn_maize___Cercospora_leaf_spot_Gray_leaf_spot	0.67	0.8	0.73	0.67	0.8	0.73	5
11	Corn_maize___Common_rust	1	1	1	1	1	1	12
12	Corn_maize___healthy	1	1	1	1	1	1	12
13	Corn_maize___Northern_Leaf_Blight	1	0.8	0.89	1	0.8	0.89	10
14	Grape___Black_rot	1	1	1	1	1	1	12
15	Grape___Esca_Black_Measles	1	1	1	1	1	1	14
16	Grape___healthy	1	1	1	1	1	1	4
17	Grape___Leaf_blight_Isariopsis_Leaf_Spot	1	1	1	1	1	1	11
18	Orange___Haunglongbing_Citrus_greening	1	1	1	1	1	1	55
19	Peach___Bacterial_spot	1	1	1	1	1	1	23
20	Peach___healthy	1	1	1	1	1	1	4
21	Pepper_bell___Bacterial_spot	0.91	1	0.95	0.91	1	0.95	10
22	Pepper_bell___healthy	1	0.93	0.97	1	0.93	0.97	15
23	Potato___Early_blight	1	0.7	0.82	1	0.7	0.82	10
24	Potato___healthy	1	1	1	1	1	1	2
25	Potato___Late_blight	0.91	1	0.95	0.91	1	0.95	10
26	Raspberry___healthy	1	1	1	1	1	1	4
27	Rice___Bacterial_leaf_blight	0.67	1	0.8	0.67	1	0.8	2
28	Rice___Brown_spot	0.8	1	0.89	0.8	1	0.89	4
29	Rice___healthy	1	1	1	1	1	1	11
30	Rice___Leaf_blast	0.40	0.67	0.50	0.4	0.67	0.5	3
31	Soybean___healthy	1	0.98	0.99	1	0.98	0.99	51
32	Squash___Powdery_mildew	1	1	1	1	1	1	18
33	Strawberry___healthy	1	1	1	1	1	1	5
34	Strawberry___Leaf_scorch	1	1	1	1	1	1	11
35	Tomato___Bacterial_spot	1	0.95	0.98	1	0.95	0.98	21
36	Tomato___Early_blight	0.69	0.9	0.78	0.69	0.9	0.78	10
37	Tomato___healthy	1	1	1	1	1	1	16
38	Tomato___Late_blight	0.9	0.95	0.92	0.9	0.95	0.92	19
39	Tomato___Leaf_Mold	1	1	1	1	1	1	10
40	Tomato___Septoria_leaf_spot	1	1	1	1	1	1	18
41	Tomato___Spider_mites_Two-spotted_spider_mite	0.89	1	0.94	0.94	1	0.97	17
42	Tomato___Target_Spot	0.93	1	0.97	0.93	1	0.97	14
43	Tomato___Tomato_mosaic_virus	1	1	1	1	1	1	4
44	Tomato___Tomato_Yellow_Leaf_Curl_Virus	1	1	1	1	0.98	0.99	54

Table 9 shows that the results on DenseNet 121 have decreased accuracy in the Apple Healthy, Cherry Including Sour Powdery Mildew and Tomato Spider Mites Two Spotted Spider Mite.

- **Mobile Device Resource Consumption**

The experiments result of the using of battery resource from the testing process on mobile are summarized in Table 10. Whereas the use of memory resource is summarized in Table 12.

Table 10: Battery Usage performance each class

Class	MobileNet V2	NasNet Mobile	Inception V3	DenseNet 121
Apple___Apple_scab	2505	3478	3375	4004
Apple___Cedar_apple_rust	1636	2462	1731	2234
Apple___Frogeye_Spot	3734	3514	3296	4680
Apple___healthy	9386	10631	12292	10942
Blueberry___healthy	7707	8718	11892	9175
Cherry_including_sour___healthy	5245	5109	7055	5917
Cherry_including_sour___Powdery_mildew	7337	6340	8828	7580
Coffee___healthy	926	1780	1325	1051
Coffee___Leaf_rust	6248	5982	8996	7119
Corn_maize___Cercospora_leaf_spot_Gray_leaf_spot	2600	3787	4085	2775
Corn_maize___Common_rust	5930	7978	9977	7592
Corn_maize___healthy	6243	8633	10014	8076
Corn_maize___Northern_Leaf_Blight	5590	5964	8116	6037
Grape___Black_rot	5757	7164	9810	8024
Grape___Esca_Black_Measles	7411	9043	10939	7414
Grape___healthy	2711	2921	3304	2403
Grape___Leaf_blight_Isariopsis_Leaf_Spot	8212	7105	9148	6579
Orange___Haunglongbing_Citrus_greening	31824	26532	42316	31375
Peach___Bacterial_spot	13666	14014	19854	17297
Peach___healthy	2717	3035	3140	3205
Pepper_bell___Bacterial_spot	6040	6473	7926	6964
Pepper_bell___healthy	7993	10496	11695	9314
Potato___Early_blight	5593	5461	8857	5663
Potato___healthy	852	1204	1610	1258
Potato___Late_blight	6113	7525	7749	6989
Raspberry___healthy	3393	2653	2975	2155
Rice___Bacterial_leaf_blight	1555	1139	1463	1330
Rice___Brown_spot	3029	2599	3295	2722
Rice___healthy	5990	7142	9129	7281
Rice___Leaf_blast	2690	1592	2203	2056
Soybean___healthy	34565	26606	38231	30359
Squash___Powdery_mildew	10837	10765	14082	9649
Strawberry___healthy	2417	3698	4182	2947
Strawberry___Leaf_scorch	5489	7755	9056	8451
Tomato___Bacterial_spot	11801	15678	17491	12799
Tomato___Early_blight	5095	6908	8346	6986
Tomato___healthy	9526	11033	13680	10793
Tomato___Late_blight	14504	14566	15169	12533
Tomato___Leaf_Mold	7148	6292	8467	6589
Tomato___Septoria_leaf_spot	10024	11577	15867	12254
Tomato___Spider_mites_Two-spotted_spider_mite	9333	12553	14985	10055
Tomato___Target_Spot	9843	8517	12570	8627
Tomato___Tomato_mosaic_virus	2479	3542	3565	2578
Tomato___Tomato_Yellow_Leaf_Curl_Virus	32584	34992	49111	37061
Total	**346278**	**364956**	**471197**	**370892**

From the overall results, from the four models tested the total battery usage on MobileNet V2 is very low while the total battery usage on Inception V3 is very high. To strengthen the

analysis in calculating battery usage, an ANOVA analysis method will be used, with $\alpha = 0.05$ by obtaining the F and P-values of battery usage, which are summarized in Table 11.

Table 11: ANOVA Oneway Analysis of Battery Usage

Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	16354720.88	3	5451573.627	163.0281328	7.11066E-96	2.608745145
Within Groups	77445802.15	2316	33439.46552			
Total	93800523.03	2319				

Analysis of Table 11 shows the P-value 7.11066E-96 is lower than 0.05, so it can be concluded statistically there are significant differences between the four models tested.

Table 12: Memory Usage performance each class

Class	MobileNet V2	NasNet Mobile	Inception V3	DenseNet 121
Apple___Apple_scab	1204	1208	2094	1200
Apple___Cedar_apple_rust	628	588	1066	627
Apple___Frogeye_Spot	1240	1192	2189	1254
Apple___healthy	3160	3172	5739	3351
Blueberry___healthy	3096	3112	5466	3297
Cherry_including_sour___healthy	1852	1904	3239	1908
Cherry_including_sour___Powdery_mildew	2184	2308	3918	2383
Coffee___healthy	396	416	701	400
Coffee___Leaf_rust	2192	2284	3905	2351
Corn_maize___Cercospora_leaf_spot_Gray_leaf_spot	1020	1024	1748	1081
Corn_maize___Common_rust	2576	2496	4308	2648
Corn_maize___healthy	2480	2460	4276	2609
Corn_maize___Northern_Leaf_Blight	2036	2144	3537	2139
Grape___Black_rot	2404	2384	4276	2587
Grape___Esca_Black_Measles	2860	2796	4977	3008
Grape___healthy	828	792	1418	859
Grape___Leaf_blight_Isariopsis_Leaf_Spot	2188	2288	3921	2360
Orange___Haunglongbing_Citrus_greening	11008	10896	19432	11534
Peach___Bacterial_spot	4844	4724	8168	4947
Peach___healthy	816	796	1421	848
Pepper_bell___Bacterial_spot	2032	2052	3572	2221
Pepper_bell___healthy	3052	3160	5342	3313
Potato___Early_blight	2052	2060	3556	2000
Potato___healthy	396	412	701	400
Potato___Late_blight	2112	2096	3591	2000
Raspberry___healthy	808	844	1437	800
Rice___Bacterial_leaf_blight	396	412	701	400
Rice___Brown_spot	824	848	1437	848
Rice___healthy	2196	2292	3959	2387
Rice___Leaf_blast	592	600	1056	670
Soybean___healthy	10248	10232	18038	10675
Squash___Powdery_mildew	3660	3760	6430	3867
Strawberry___healthy	1012	1004	1770	1080
Strawberry___Leaf_scorch	2220	2172	3943	2344
Tomato___Bacterial_spot	4316	4268	7429	4243
Tomato___Early_blight	2124	2000	3572	2048
Tomato___healthy	3324	3256	5726	3403
Tomato___Late_blight	3800	3956	6738	4008
Tomato___Leaf_Mold	1972	2072	3540	2144
Tomato___Septoria_leaf_spot	3656	3748	6452	3659
Tomato___Spider_mites_Two-spotted_spider_mite	3364	3460	6046	3539
Tomato___Target_Spot	2820	2840	5028	2971
Tomato___Tomato_mosaic_virus	804	816	1434	832
Tomato___Tomato_Yellow_Leaf_Curl_Virus	11372	10940	19372	11392
Total	**118164**	**118284**	**206669**	**122635**

Overall, the total memory usage on MobileNet V2 is very low followed by NasNet Mobile then DenseNet 121 and finally Inception V3 with the highest total memory usage. ANOVA oneway analysis will be used to strengthen the analysis in calculating memory usage, with $\alpha = 0.05$ by getting the values of F and P-value of memory usage, which is summarized in Table 13.

Table 13: ANOVA Oneway Analysis of Memory Usage

Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	9804177.521	3	3268059.174	40741.21097	0	2.608745
Within Groups	185778.1069	2316	80.21507206			
Total	9989955.628	2319				

Analysis of Table 13 shows the P-value is equal to 0, so that it can be concluded statistically that there is no significant effect between the four models tested.

5. CONCLUSION

The study evaluates the transfer learning method using pretrained model that allows the retraining process to be carried out using workstations with moderate specifications and more reasonable time due to lighter computing costs.

Analysis of the results of the retraining concluded that although the Inception V3 model runs only up to 15 epochs, the size of output file produced is three times larger than the other three models, and in the testing phase Inception V3 is superior to other models with an accuracy value reaching 98.68%.

The results of comparative experiments between PCs and mobile devices showed that certain classes experienced a slight decrease in the value of overall accuracy. The level of accuracy in InceptionV3 is considered quite reliable compared to other models, but with the use of resources on mobile devices that are very high, so Inception V3 is considered inappropriate to detect classification of plant diseases on mobile devices.

Meanwhile on NasNet Mobile, the use of resources on mobile devices is not very high and with a high level of accuracy. This accuracy does not have significant difference when run and tested on either PC or mobile device.

Based on the methodology and the results of the experiment, it was concluded that the NasNet Mobile model is the most suitable model to be used on mobile device as plant disease detection classifications. That is because the model has high accuracy and detection speed, with low resource requirements

REFERENCES

1. C. H. Bock, G. H. Poole, P. E. Parker and T. R. Gottwald, **Plant disease severity estimated visually, by digital photography and image analysis, and by hyperspectral imaging**, *CRC. Crit. Rev. Plant Sci.* 29, (2010). https://doi.org/10.1080/07352681003617285
2. H. D. Gadade and D. K. Kirange, **Machine Learning Approach towards Tomato Leaf Disease Classification**, *Int. J. Adv. Trends Comput. Sci. Eng.* 9, (2020). https://doi.org/10.30534/ijatcse/2020/67912020
3. R. Saha and S. Neware, **Orange Fruit Disease Classification using**, *Int. J. Adv. Trends Comput. Sci. Eng.* 9, (2020). https://doi.org/10.30534/ijatcse/2020/211922020
4. J. Han, M. Kamber and J. Pei, **Data Mining Concepts and Techniques**, 3rd ed, Elsevier Inc., Waltham, (2012).
5. A. Krizhevsky, I. Sutskever and G. E. Hinton, **ImageNet classification with deep convolutional neural networks**, *Adv. Neural Inf. Process. Syst.* 2,

(2012) 1097.
6. J. Patterson and A. Gibson, **Deep Learning A Practitioner's Approach**, 1st ed, oreilly, boston, (2016).
7. X. Zhang, Y. Qiao, F. Meng, C. Fan and M. Zhang, **Identification of Maize Leaf Diseases Using Improved Deep Convolutional Neural Networks**, *IEEE Access* 6, (2018). https://doi.org/10.1109/ACCESS.2018.2844405
8. K. P. Ferentinos, **Deep learning models for plant disease detection and diagnosis**, *Comput. Electron. Agric.* 145, Elsevier, (2018).
9. E. C. Too, L. Yujian, S. Njuki and L. Yingchun, **A comparative study of fine-tuning deep learning models for plant disease identification**, *Comput. Electron. Agric.*, Elsevier, (2018).
10. M. Brahimi, M. Arsenovic, S. Laraba and S. Sladojevic, **Deep Learning for Plant Diseases : Detection and Saliency Map Visualisation Deep Learning For Plant Diseases : Detection and Saliency map Visualization**, (2018). https://doi.org/10.1007/978-3-319-90403-0_6
11. S. B. Jadhav, V. R. Udupi, S. B. Patil and D. Cnn, **Convolutional neural networks for leaf image-based plant disease classification**, *International Journal of Artificial Intelligence*.8, (2019).
12. D. Hughes and Marcel Salathe, **An open access repository of images on plant health to enable the development of mobile disease diagnostics**, (2015).
13. B. Syamsuri and G. P. Kusuma, **Plant Disease Classification Using Lite Pretrained Deep Convolutional Neural Network on Android Mobile Device**, *International Journal of Innovative Technology and Exploring Engineering*. 9 (2019). https://doi.org/10.35940/ijitee.B6647.129219
14. H. Do, **Rice Diseases Image Dataset: An image dataset for rice and its diseases**, (2019). [Online]. Available: https://www.kaggle.com/minhhuy2810/rice-diseases-image-dataset. [Accessed: 27-Jan-2020].
15. A. K. G. Francisco, **Rice Diseases DataSet**, (2019). [Online].Available: https://github.com/aldrin233/RiceDiseases-DataSet. [Accessed: 27-Jan-2020].
16. M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov and L. C. Chen, **MobileNetV2: Inverted Residuals and Linear Bottlenecks**, *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, (2018).
17. B. Zoph, V. Vasudevan, J. Shlens and Q. V. Le, **Learning Transferable Architectures for Scalable Image Recognition**, *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, (2018) 8697.
18. B. Zoph, V. Vasudevan, J. Shlens and Q. V Le, **AutoML for large scale image classification and object detection**, (2017). [Online].Available: https://ai.googleblog.com/2017/11/automl-for-large-scale-image.html. [Accessed: 12-Dec-2019].
19. C. Szegedy, V. Vanhoucke and J. Shlens, **Rethinking the Inception Architecture for Computer Vision**, (2014).
20. G. Huang, Z. Liu, L. Maaten van der and K. . Weinberger, **Densely Connected Convolutional Networks**, (2016).
21. M. Vanetti, **Confusion Matrix Online Calculator**, Confusion Matrix Online Calculator, (2007) . [Online]. Available: http://www.marcovanetti.com/pages/cfmatrix/?noc=44. [Accessed: 13-March-2020].