Mobile Platform Implementation of Lightweight Neural Network Model for Plant Disease Detection and Recognition

Anton Louise P. de Ocampo[1] and Elmer P. Dadios[2]

***Abstract*—Due to the increasing world population, agriculture sectors from around the globe are challenged to increase their yield per year. However, harvests suffer from defects due to plant diseases. The current methods for mitigate spreading plant diseases are entirely dependent on the detection and recognition of such. Detection and recognition systems for plant diseases often require huge database for reference and/or computationally expensive systems. In this paper, we present a computationally light neural network model for detection and recognition of plant diseases and implement it to a mobile platform. Here, a two-step training process is used: pre-training on ImageNet data set of wide variety of objects and retrained on data set of specific plant diseases. The model achieved a test accuracy of 89.0 %.**

I. INTRODUCTION

By 2050, it is expected that the world population will explode to 9.6 billion. More to feed requires more agricultural yield. To suffice the demand for food, at least 70 % increase from the present global yield must be produced [1]. The presence of plant diseases imposes great challenges for farmers to meet the demand. The impact of plant diseases on the quality of the yield can be very serious although not all losses on agricultural harvest are accounted to such pathogenic attacks [2]. From around 40 years, methods for pest and disease management had been important in increasing food production. But despite these techniques, plant diseases still claim 10 to 16 % of the total global harvest [3]. The lack of operational forecasting systems for incidences of pests and diseases render management and protections methods difficult to implement [4]. In order to perform a certain protection scheme, it is necessary to first be able to detect and recognize the disease present in crops. Various methods, models and systems using image processing and machine learning techniques were proposed for such purpose. One example is the study of Vakilian et.al, which has used Artificial Neural Network in detecting incidences of fungi-caused defects in cucumber leaves. The model can estimate the HPI or Hour Post Inoculation using leaf image features fed to a pre-trained ANN using back-propagation [5]. In another method, images of diseased plants are sent to a server via a mobile device for image processing and analysis. In [6], the authors take images of plant diseases by phone and send them to a computer for processing. The authors claimed 87.75 % average accuracy for detecting diseases in palm trees. Similarly, the authors in [7] used Android phone to take pictures of crops with suspected symptoms of plant disease and send it to a server for cross-validation. In traditional Machine Learning techniques, multidimensional features of plant images are used to train different classifiers such as K-Nearest Neighbor, Naive Bayes, Decision Trees, Support Vector Machines, Radial Basis Function and Neural Networks. The common difficulties encountered with such approaches are in the manner of feature extraction and selection which include the presence of backgrounds, optimization of features, and automation of continuous extraction under real environmental conditions [8]. Some improvements have been proposed which include the use K-means clustering for segmentation, anisotropic diffusion to remove image noise and other background data removal methods [9]. A much larger improvement has been achieved when the method of feature extraction and selection has been fully automated using Convolutional Neural Networks or CNN. Several models which incorporate CNN have been developed for the purpose of plant disease classification. In [10], the authors developed a Caffe framework-based classification model for plant disease classification which achieved precision in the rage of 91 - 98 % with 96.3 % average. In [11], a classifier based on LeNet architecture is developed for automating detection of diseases in banana leaves. In [12], combination of three meta-architectures of CNN is developed. Faster Region-based CNN, Region-based Fully Convolutional Network and Single Multibox Detector are combined to create a robust CNN-based classifier for plant disease detection in tomatoes. The experiments showed effectiveness of the CNN classifier in detecting nine different tomato diseases. In [13], CNN model with pre-trained network is used in classifying cassava plant diseases including mosaic disease, cassava brown streak, red and green mite damages, and brown leaf spot. Accuracy achieved averages to 96.6 %. In [14], 98.6 % accuracy has been achieved for classifying diseases of olive plants in controlled environment using vision-based classifier incorporating CNN. As described in above examples of CNN applications, the models developed are optimized for a specific plant. In [15], a more diversified classifier has been developed for detection and recognition of diseases of more species of plants. The authors claimed 96.3 % accuracy. However, it is required that their inference system is implemented on a computer server which can perform computationally heavy operations.

*This work was supported by DOST ERDT, De La Salle University and Batangas State University.

[1]A.L. de Ocampo is an ERDT scholar and with Faculty of Electronics, Instrumentation, and Mechatronics Engineering, Batangas State University, Rizal Ave. Ext., Batangas City, Philippines

[2]E. Dadios is with the Department of Manufacturing Engineering and Manangement, De La Salle University, 2401 Taft Ave, Malate, Manila, 1004 Metro Manila, Philippines

978-1-5386-7767-4/18//$31.00

In this paper, we used a model that is implemented on mobile platform which is computationally lightweight compared to the previous works mentioned earlier. Classification is performed at the bounding capabilities of mobile platform. Also, the accuracy of the system classify diseases among diversified types of plants has been investigated. The paper is arranged as follows: Section 2 describes Depth-wise and Point-wise Convolutions processes that is used in the implemented model. Section 3 presents the method of implementation used. Section 4 shows the results obtained from experimentation, and lastly, Section 5 concludes the paper by summarizing important findings.

II. Depth-wise and Point-wise Convolutions

To simplify the discussion on Depth-wise Convolution, we take an RGB image of a size F x F x M as an example where F is the number of horizontal (or vertical) pixels and M is the number of channels which is 3 for RGB. The filter or kernel convolved onto this image has a dimension of K x K X 1 where K is the number of horizontal (or vertical) pixels of the kernel. There are M filters strode N times onto the F x F image: one filter per channel. In RGB image, there are 3 filters needed. The output of the depth-wise convolution is of size P X P x M where P is number of strides done to cover the whole F x F image. On the other hand, Point-wise convolution uses a 1 x 1 x M filter which is strode onto an F x F x M image N times which yields an output of F x F x M. Combining these 2 convolution processes in a single block of operation yields a layer known as depth-wise separable convolutional layer.

In Table I, the architecture of the classifier model used in this paper is shown. The architecture is known as Mobilenet and is built on depth-wise separable convolutional layers shown in Table I as 'DS Conv.'. It has 27 convolutional layers, a fully-connected layer, an average pooling layer and an activation layer. Although not shown in the table, each depth-wise convolution is followed with a batch norm and a ReLU prior to point-wise convolution. Simply, each 'DS Conv.' layer has 2 sets of batch norm and a ReLU as shown in Fig. 1. At layers where kernel is allowed to stride 2 times means that down-sampling is implemented at that layer. At the terminal of the last convolutional layer, another down-sampling is performed using average pooling which reduce the dimension of the output of the convolution to 1 dimensional vector which is fed to the fully connected layer. The output of this layer is then fed to the activation layer which is using Softmax [16].

TABLE I
Mobilenet Body Architecture

Layer	Type/Stride	Filter Shape	Input Size
1	Std. Conv./2	3x3x3x32	224x224x3
2	DS Conv./1	3x3x32 dw	112x112x32
3	Std. Conv./1	1x1x32x64	112x112x32
4	DS Conv./2	3x3x64 dw	112x112x64
5	Std. Conv. /1	1x1x64x128	56x56x64
6	DS Conv./1	3x3x128 dw	56x56x128
7	Std. Conv. /1	1x1x128x128	56x56x128
8	DS Conv./2	3x3x128 dw	56x56x128
9	Std. Conv. /1	1x1x128x256	28x28x128
10	DS Conv./1	3x3x256 dw	28x28x256
11	Std. Conv. /1	1x1x256x256	28x28x256
12	DS Conv./2	3x3x256 dw	28x28x256
13	Std. Conv. /1	1x1x256x512	14x14x256
14-23	5x DS Conv./1	3x3x512 dw	14x14x256
	5x Std. Conv. /1	1x1x512x512	14x14x256
24	DS Conv./2	3x3x512 dw	14x14x512
25	Std. Conv. /1	1x1x512x1024	7x7x512
26	DS Conv./2	3x3x1024 dw	7x7x1024
27	Std. Conv./1	1x1x1024x1024	7x7x1024
	Avg. Pool /1	Pool 7x7	7x7x1024
	F.C. Layer/1	1024x1000	1x1x1024
	Softmax/1	Classifier	1x1x1000

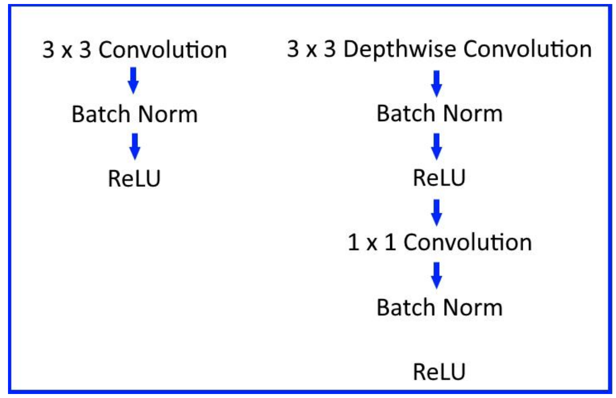

Fig. 1. Modules for Std. Conv. (left) and DS Conv. (right)

TABLE II
Statistics of Plant Disease Data Set

Plant Disease	Retraining	Validation	Testing	Total
Black Mold	333	84	111	528
Blight	837	212	279	1328
Mildew	378	96	126	600
Rusts	287	73	96	455
Bacterial Wilt	515	131	172	817
Healthy	2042	519	681	3242

III. Method of Implementation

A. Plant Disease Data Set

To diversify the data set for plant diseases, several sources were considered both online and actual sources. The collected samples reached to 6,970 images of various plant diseases. These images were manually annotated according to 6 different classes: 5 classes for major plant diseases and 1 class categorized as 'healthy'. Images were subjected to random rotation and augmented to create different views of the same sample. Each image containing the defected parts was re-sized to 224 x 224 pixels. In Fig. 2, randomly chosen images representing each type of diseases are shown. The data set is referred as "Plant Diseases Data Set" or "PDDS" in this paper.

PDDS was divided into 3 segments: 63 % for training, 16 % for validation, and 21 % for testing. In Table II, the statistics of the data set is presented.

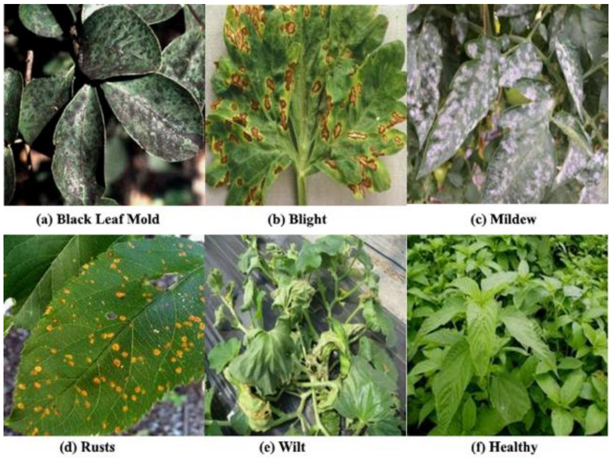

Fig. 2. Randomly selected representative for each plant disease

B. Transfer Learning and Optimization

The model used in this paper is already pre-trained on 60,000 images of different objects including things, persons, plants and animals. In order to direct its inferential capabilities to that of classifying diseases in plants, the final classifying layer is retrained to the data set of plant diseases prepared in this research. The width used for the classifier model is 50 % of the largest width for Mobilenets. For processing images of input dimension of 224 x 224, the total process parameters generated sums up to 1.3 million only which is relatively smaller compared to other CNN models. The model used is retrained 4,000 times using Tensorflow libraries. During the initial phase of retraining, the classifier model is allowed to train on PDDS except for the last layer. Prior to the last last, a bottleneck is generated for each image. Bottleneck is a set of values of neural activation from the second-to-the-last layer of the classifier model. Bottlenecks represent the feature map for each image as an input to the last layer which performs the classification. Only the last layer of the classifier is retrained 4,000 times and received adjustment of weights while the other layers are trained only once. This approach in retraining made it faster and feasible for less powerful machines. After retraining, optimization is conducted to remove training operations that are redundant or have very little effect on the classifier weights. The model weights are quantized to reduce the total memory footprint of the model once implemented in mobile phone. Once optimized, validation and test are conducted on the model. Tensorboard is used to monitor the processes from training to validation to testing.

IV. Results and Discussion

A. Accuracy and Cross-entropy

The generated bottleneck for each image is reused every training epoch. Fig. 3 shows that the increased training epoch improves the training and validation accuracy of the classifier model. The final test accuracy has reached 89 % during testing

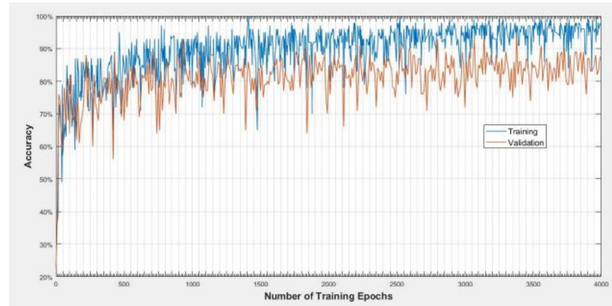

Fig. 3. Accuracy achieved during training and validation. Additional training epochs increased accuracy.

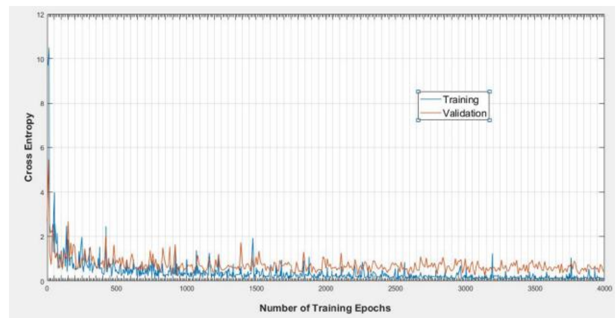

Fig. 4. Cross-entropy variations during training and validation process

with very little loss of accuracy as described by cross-entropy values shown in Fig. 4.

1) Optimization, Quantization and Compression: Deployment of classifier model to a mobile platform requires the model to be optimized to reduce the multiplication-addition operations. Nodes that are not needed by Tensorflow libraries for mobile applications are removed during optimization which reduces the complexity of calculations by merging batch norm operations into the convolutional weights. Quantization is the process of weight mapping into smaller number without changing the architecture or structure of the classifier model. Table III shows the classifier model sizes after optimization, quantization and compression.

B. Testing the Classifier

The classifier model which is Mobilenet-based architecture developed on this research can classify 5 major diseases whose symptoms are appearing on leaves of wide variety of plants. In Fig. 5, an example of actual image and the classification

TABLE III
Percent Compression (PC).

Model	Uncompressed (in MB)	Compressed (in MB)	PC
Retrained	5.492751 MB	5.035895 MB	8.32
Optimized	5.464665 MB	5.031979 MB	7.92
Quantized	5.464684 MB	1.635099 MB	70.08

Fig. 5. Evaluation using an actual image of a plant having blight.

TABLE IV
EVALUATION USING IMAGE BLIGHT.JPG.

	Retrained	Optimized	Quantized
Evaluation Time	0.547s	0.422s	0.406s
Inferential Accuracy			
blight	0.999742	0.999742	0.999985
rusts	0.000257497	0.000257497	1.54E-05
wilt	4.06E-09	4.06E-09	1.87E-14
mildew	2.56E-12	2.56E-12	1.83E-14
black mold	2.00E-12	2.00E-12	2.66E-14

result for the plant disease are shown. Table IV shows the probability distribution for the classification process. In terms of probabilities, the optimized, quantized, and compressed models are not significantly different. This means that doing quantization prior to compression reduces size but does not degrade the performance of the classifier.

V. CONCLUSIONS

It was presented that even with relatively smaller data set, classifier based on Mobilenet architecture can effectively classify plant diseases. The classifier model achieved a relatively fair accuracy even when it is implemented on mobile platform only. All inferential processes can be autonomously done in mobile devices opening more opportunities for other applications. To improve the classifier more, expanding the data set to other diseases can be done. For future development, we are testing the model on smaller machines such as Raspberry pi. We are also developing a system that uses the same model for processing video feeds of the farm for detection of possible incidents of crop defects.

ACKNOWLEDGMENT

The authors thank De La Salle University - Gokongwei College of Engineering, Batangas State University, and DOST Engineering Research and Development for Technology (ERDT) for their support on this work.

REFERENCES

[1] Beecham Research Ltd., "Towards Smart Farming: Agriculture Embracing the IoT Vision," London, 2014.
[2] S. Savary, A. Ficke, J.N. Aubertot, and C. Hollier, "Crop losses due to diseases and their implications for global food production losses and food security," Food Security, 2012.
[3] S Chakraborty and A.C. Newton, "Climate change, plant diseases and food security: an overview," Plant Pathology, pp. 2-14, 2011.
[4] N Chattopadhyay, R.P. Samui, and L.S. Rathore, "Strategies for Minimizing Crop Loss due to Pest and Disease Incidences by Adoption of Weather-Based Plant Protection Techniques," Challenges and Opportunities in Agrometeorology, pp. 235-243, 2011.
[5] K.A. Vakilian and J. Massah, "An artificial neural network approach to identify fungal diseases of cucumber (Cucumis sativus L.) plants using digital image processing," Archives of Phytopathology and Plant Protection, vol. 46, no. 13, pp. 15801588, 2013.
[6] A.F. Aji et al., "Detection of Palm Oil Leaf Disease with Image Processing and Neural Network Classification on Mobile Device," International Journal of Computer Theory and Engineering, pp. 528-532, 2013.
[7] R. Yakkundimath, G. Konnurmath, and J.D. Pujari, "Android Based Detection of Plant Diseases Affecting Leaves," in NCCCI, National Conference, India, 2017, pp. 303-306.
[8] N.R. Kakade and D.D. Ahire, "A Review of Grape Plant Disease Detection," International Research Journal of Engineering and Technology , pp. 673-678, 2015.
[9] S.S. Sannakki, V.S. Rajpurohit, V.B. Nargund, and P Kulkarni, "Diagnosis and Classification of Grape Leaf Diseases using Neural Networks," in International Conference on Computing, Communication and Networking Technologies, Tiruchengode, India, 2013.
[10] S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, and D. Stefanovic, "Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification," Computational Intelligence and Neuroscience, 2016.
[11] J. Amara, B. Bouaziz, and A. Algergawy, "A Deep Learning-based Approach for Banana Leaf Diseases Classification," Bonn, Germany, 2017.
[12] A. Fuentes, S. Yoon, S.C. Kim, and D.S. Park, "A Robust Deep-Learning-Based Detector for Real-Time Tomato Plant Diseases and Pests Recognition," Sensors, vol. 17, no. 9, 2017.
[13] A. Ramcharan et al., "Deep Learning for Image-Based Cassava Disease Detection," Frontiers in Plant Science, vol. 8, 2017.
[14] A. Cruz, A. Luvisi, L. De Bellis, and Y. Ampatzidis, "Vision-Based Plant Disease Detection System Using Transfer and Deep Learning," in ASABE 2017 Annual International Meeting, MI, USA, 2017.
[15] A.M.G.J. Hanson, A. Joy, and J. Francis, "Plant Leaf Disease Detection using Deep Learning and Convolutional Neural Network," International Journal of Engineering Science and Computing, vol. 7, no. 3, pp. 5324-5328, 2017.
[16] A.G Howard et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv:1704.04861, 2017.