Identifying underline criteria of leaf disease classification with explainable AI

Thanuijan ——
dept. Computer Engineering
University of Peradeniya
Peradeniya, Sri Lanka
——@eng.pdn.ac.lk

Imesh Udara Ekanayake
dept. Computer Engineering
University of Peradeniya
Peradeniya, Sri Lanka
imeshuek@eng.pdn.ac.lk

Damayanthi Herath
dept. Computer Engineering
University of Peradeniya
Peradeniya, Sri Lanka
damayanthiherath@eng.pdn.ac.lk

***Abstract*—In an average smartphone there is a reasonable amount of noise in the captured images. Training machine learning model on a state of the art image dataset reduces the performance of real world testing of the trained model. There have been thousands of papers published to detect diseases in plants using image processing and machine learning techniques but the number of applications available for farmers are hand full. To address this issue, this research has been conducted to examine the use of effect of artificial noise on leaf images to check the performance on detecting diseases in tomato plants. The results has showed that there is a higher effect from the noises and training with noised images would increases the performance XX% higher.**

***Index Terms*—Tomato leaves, Machine Learning, Plant Disease, Image Noise**

I. Introduction

The global agriculture market is 11.2 trillion dollars with a compound annual growth rate of 11.1% [1]. There it is required to make sure the plants are healthy and an adequate amount of harvest can be taken to maintain the supply according to the demand. In 2018 38.3 million metric tons of tomato has been consumed globally [2] and 42% post-harvest loss has happened mostly due to pre-harvest practices [3]. Where diseases cause for a major part of the losses.

The ability to capture digital images and mobile computational power have been rapidly increased during the last decade which had led to conduct more researches based on images. Using high quality images to train a machine learning model has led to low the performance in real world applications due to inferior quality in average smart phone cameras.

To examine and overcome the above issue, we have added four different kind of noises to the tomato leaf disease dataset ——CITE—— and tested the performance of the models trained with and without artificially added noise images. For the experiment the mobileNet [4] architecture has been used as the neural network model due to the shallow neural network architecture and less requirement of computational power to run in a mobile device. The four types of noises that has been considered are Gaussian noise, Salt-and-pepper noise, Poisson noise and Speckle noise. The above four different noises are the most commonly occurred noises in digital image capturing.

II. Related works

A. Tomato leaves diseases

Mostly the bacteria survive in the cold weather on tomato plants and on infected plant debris. There are 9 most common types of tomato leaf diseases such as Tomato mosaicvirus, Target Spot,Bacterial spot, Tomato Yellow Leaf Curl Virus, Late blight, Leaf Mold, Early blight, Spider mites Two-spotted spider mite and Septoria leaf spot [5].

B. Image noises

1) Gaussian noise: Gaussian noise denoting a signal noise that has a probability density function equal to that of the normal distribution. The noise has been considered separately for the each red, green and blue colours.

Principal sources of Gaussian noise occurs during image acquisition due to poor illumination and high temperatures or during transmission.

2) Salt and paper noise: Salt and pepper noise is an Impulse Noise which makes noise pixel intensities in extreme values. In general this noise type occurs during the data transmission, failure in memory cell or analog-to-digital conversion in capturing devices.

3) Poisson noise: Photon noise, also known as Poisson noise, is a basic form of uncertainty associated with the measurement of light, inherent to the quantized nature of light and the independence of photon detections [6]. Poisson distribution approaches a normal distribution about its mean, and the elementary events (photons, electrons, etc.) are no longer individually observed, typically making shot noise in actual observations indistinguishable from true Gaussian noise. Since the standard deviation of shot noise is equal to the square root of the average number of events N, the signal-to-noise ratio (SNR) is given by,

C. Transfer learning

Transfer learning is a machine learning technique where a pre-trained model is use as the initial training model of a new problem [7]. There it is possible to transfer the knowledge learnt for the previous problem into the new solution. There are many benefits comes along with this method, saving of resources and improve efficiency when training new models are one primary benefits. Moreover, When data is not enough to train a model from scratch, it tends to over fit or provide unsatisfied results. There transfer learning is one of the best methods to tackle this issue.

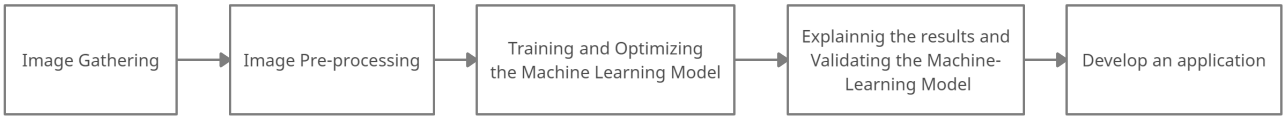

Fig. 1. Proposed workflow.

$$p_G(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-\mu)^2}{2\sigma^2}}$$

Fig. 2. Gaussian Equation

z represents the grey level,μ represent the mean and represents the standard deviation

$$\mathrm{SNR} = \frac{N}{\sqrt{N}} = \sqrt{N}$$

Fig. 3. Gaussian Equation

D. Model generalization with noise

Using noise to generalize machine learning models has been used in many object detection research [8], [9]. The noise can be used as a method of regularisation to avoid overfitting data as well as to generalize the model for the real world data [10].

E. Explainable AI

The current state of machine learning suffers from the obtaining the trust of domain experts to implement the applications in the real world [11].

1) LIME: LIME - Stands for Local interpretable model-agnostic explanation. This explainability method can be used to explain individual predictions made by a black box machine learning model [12]. LIME will generate several samples images that are similar with our input image by turning on and off some of the super-pixels of the image.

2) SHAP: SHAP (SHapley Additive exPlanations) is a game theoretic approach to explain the output of any machine learning model and it connects optimal credit allocation with local explanations using the classic Shapley values from game theory and their related extensions [13]. In the explanation it represent defferent pixels in 2 different colours(Red and Blue), there the red pixels represent positive SHAP values that increase the probability of the class, while blue pixels represent negative SHAP values the reduce the probability of the class.

3) GradCAM: Gradient-weighted Class Activation Mapping,uses the gradients of the targeted class flowing into the final convolution layer to produce a coarse localization map highlighting the important regions in the image.

III. Methodology

A. The workflow

The research as been conducted in 5 main steps: Data gathering, image pre-processing, training and optimizing the machine learning model, explain the result to validate the model with explainable AI and develop and application with an interface (Fig. 3).

B. Data Gathering

A popular data set called PlantVillage from Kaggle online platform was used for this research. This data set has more than 163,000 images. Mainly these images are categorized into three: original RGB, Grayscale version of the raw images, and RGB images with the leaf segmented and color corrected. Only the original RGB category - taken from several plants categorized into several diseases - was considered for this study. The distribution of the frequency of images in each disease class of every plant was varied. The tomato plant showed a relatively high number of disease classes and better variation (Fig. 4). Therefore, in this study, only tomato plant leaf images from four disease classes with a higher frequency: Bacterial spot, Tomato Yellow Leaf Curl Virus, Late blight, and Early blight, and images from the healthy class were selected to train the model. Fig. 5 depicts that all classes except the yellow leaf curl virus are almost equally distributed; Still, the data is imbalanced. Therefore, the imbalance in the data has to be handled.

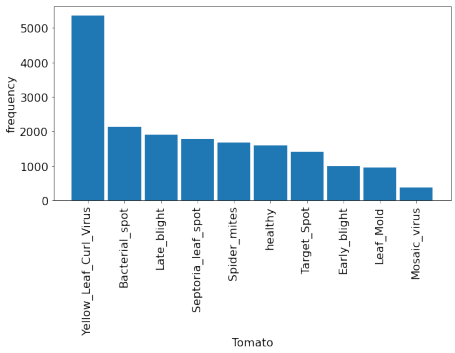

Fig. 4. Distribution of frequency of images in each tomato plant's disease class in plantVillage Data set

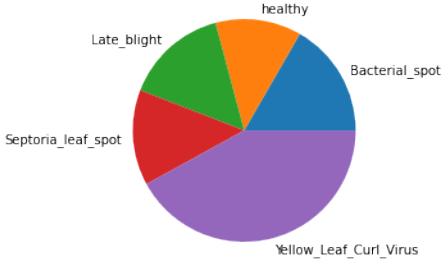

Fig. 5. Distribution of frequency of images in selected disease classes.

C. Model selection and training

Under various pre-trained image classification models such as VGG16, AlexNet, DenseNet121, ResNet50, InceptionV3, and MobileNetV1; MobileNetV1 was proposed and obtained from Keras (Keras is an online platform that provides a Python interface for deep learning models). MobileNetV1 is purpose-built for basic mobile and embedded system applications. MobileNetV1's compact nature is shown in Table I. MobileNetV1, which can optionally be loaded weights pre-trained with ImageNet data set, showed 99% accuracy when fine-tuned in the previous study (Demonstrated in Table I) [14]. Therefore, to detect a broader range of plant diseases with high accuracy in the factual environment, MobileNetV1 was customized and fine-tuned for this particular study.

MobileNetV1 architecture is built based on depth-wise separable convolutions. The core idea of the depth-wise separable convolutions is that rather than using a standard 3x3 convolution filter (Fig. 6), the function is split into depth-wise convolution filters (Fig. 7) followed by 1x1 convolution, called a point-wise convolution (Fig. 8). This convolution factorization vastly reduced the number of parameters to 4.3 million.

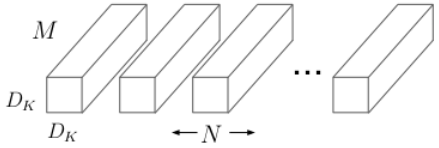

Fig. 6. Standard Convolution

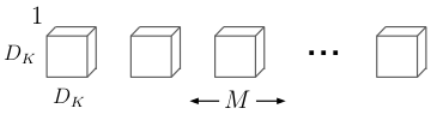

Fig. 7. Depth-wise Convolutional filters

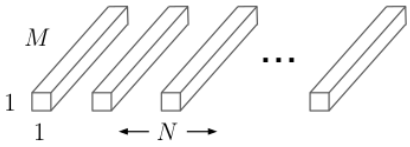

Fig. 8. Point-wise Convolutional filters

Originally MobileNetV1 had 28 layers, counting depth-wise and point-wise convolution separately. The basic building block of MobileNetV1 consists of 3x3 depth-wise convolution followed by 1x1 point-wise convolution in each layer; A batch normalization and Rectified Linear activation Unit are applied after each convolution layer as shown in Fig. 9. Additionally, the first layer and the stridden convolution in the depth-wise convolution handle the downsampling. A Global average pooling is applied at the final stage before the fully connected layer to reduce the spatial resolution. Then, the output from the pooling layer is flattened and fed through the fully connected layer to alter the weights accordingly. Then the output from the fully connected layer is fed through the final layer, which uses the softmax activation function, which determines the probabilities of an input being in a particular class [15].

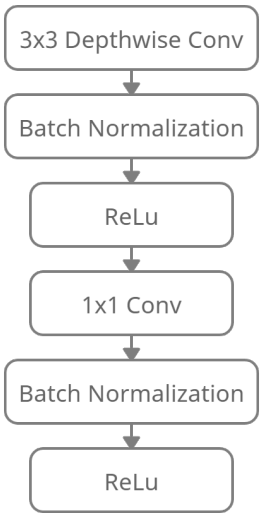

Fig. 9. Depthwise separable convolution

Keras applications expect specific kind of preprocessing. Hence, before feeding the images to the fine-tuned MobileNetV1 application, the image data should be preprocessed with the preprocess function defined for MobileNetV1.

The alpha value will control the width of the network. Shrinking the model with the multiplier alpha shows accuracy, computation, and size trade-offs. The alpha value is set as one to use the default number of filters from the research paper used at each layer. Like-wise all parameters when modeling the mobileNet model were set to default values used in the research paper [15].

The first 23 layers of the mobileNet model, which wanted to be fine-tuned, were set non-trainable to keep the weights unaltered, and then the bottom five layers were removed. Finally, the output from the first 23 layers was fed through a Global Average pool layer (dimension is 1024) and a Dense layer (dimension is 5) with Soft Max Activation function, which are defined to predict the disease among five disease classes of the tomato plant.

In the end, the MobileNet had twenty-five layers in total; the first twenty-three layers were non-trainable, and the last two layers were trainable. After setting the layers, the model is compiled with the parameters: 0.0001 as learning rate, Adam optimizer as an optimizer, categorical cross entropy as loss function, and accuracy as metrics (compilation is a method, in Keras, used to freeze the behavior of the model until the compile method is called again). Then MobileNetV1 model was trained with the parameters: the batch size of 32; epochs of 17, where each epoch contained 199 steps. When the fine-tuned MobileNet model was compiled and trained using this parameter with the same data set, the model revealed an accuracy of 99% in a previous study [14]. Thus, those exact parameters are used for this work as well.

TABLE I
COMPARISON OF MODALS ON KERAS

Model	Training Accuracy	Validation Accuracy	Testing F1 Score	Training time per epoch/s	Size	Parameters	Depth	Time(ms) per inference step
VGG16	0.9874	0.9673	0.97	1700	528MB	138.4M	81	109.4
DenseNet121	0.9755	0.9962	0.99	700	33MB	8.1M	242	77.1
ResNet50	0.9998	0.9975	0.99	680	98MB	25.6M	107	58.2
InceptionV3	0.9914	0.9874	189	440	92MB	23.9M	189	42.2
MobileNet	0.9998	0.9962	0.99	380	16MB	4.3M	55	22.6

Since images could be inaccurate in the real world, the model could inaccurately behave when it sees noisy images. Therefore, this study compared the model's performance in different circumstances.

- The model was trained with the original data set, and the model performance was analyzed when testing with accurate and noised images.
- The model was trained with the noisy data set, and the model performance was analyzed when testing with accurate and noised images.

Thus, the model's performance in each circumstance was evaluated and identified as the best scenario suitable for real-world application.

Since the data set is imbalanced, five criteria were used to evaluate the performance: accuracy, precision, recall, F1 score, and confusion matrix.

The accuracy is the number of correctly predicted observations out of all the predictions (Equation **??**). It is an excellent technique to optimize the final machine learning model until the data set has well-balanced classes. When the data set is class imbalanced, the accuracy does not handle the disparity between the number of positive and negative labels. Therefore, in addition to the accuracy, other evaluation criteria are also used in this work.

$$Accuracy = \frac{CorrectlyPredictedObservations}{TotalObservations} \quad (1)$$

The precision evaluates the ratio of correctly predicted observations among the once classified as predicted positives: it could be correctly identified positive or incorrectly identified positives (Equation **??**). Thus precision is beneficial when the cost for the incorrectly identified positives is high.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \quad (2)$$

The recall evaluates the ratio of correctly predicted observations among the actual positives: it could be correctly identified positives or correctly identified negatives (Equation 3). Thus recall is beneficial when the cost for the correctly identified negative is high.

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \quad (3)$$

The F1-Score is the harmonic average of accuracy and recall and allows better measurement of misclassifications than the accuracy metric (Equation 4). When F1-Score is 1, the model is said to be perfect, while when the F1-score is 0, the model is a total failure.

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (4)$$

The confusion matrix is used here since it illustrates how the classification model is confused while making predictions. It gives better insight into whether the model works correctly and what types of errors it produces. This matrix is a square matrix in which the row represents the model's predicted value, and the column represents the actual value of the data and vice versa.

D. Explaining the results

High accuracy can be obtained with the MobileNetV1 model; the "back box (un-interpretability)" problem of the model is a serious issue when one needs clarification of why the model is predicted as a particular class of disease. Therefore it is crucial to increase the explainability of this model. Thus, some popular explainable models, such as GradCam, Lime, and SHAP are proposed to explain the model.

1) SHAP: SHAP

2) Lime:

3) GradCam:

E. User application

IV. RESULTS

A. Model outcome

The model outcome is consist of 5 classes which represents the tomato bacterial spot(TBS), tomato late blight(TLB), tomato septoria leaf spot(TSL), tomato yellow leaf curl virus(TYLCV) and healthy tomato leaf(HTL). The overall performance of the model is presented in the (Table II) and the performance of each class is presented in the (Table III).

TABLE II
OVERALL MODEL PERFORMANCE

Matrix	Value
Accuracy	val%
Precision	val%
Recall	val%
F1 score	val%

TABLE III
CLASS PERFORMANCE

Matrix	HTL	TLB	TSL	TYLCV	TBS
Accuracy	value%	value%	value%	value%	value%
Precision	value%	value%	value%	value%	value%
Recall	value%	value%	value%	value%	value%
F1 score	value%	value%	value%	value%	value%

B. Model explanation

Once the model is trained and analysed the results, 3 different instant based explanation methods has been applied.

First the gradcam explanation of an instance has considered. There it can be seen that

C. User application

V. Discussion

VI. Conclusion

In conclusion, a benchmark dataset which has been used to train a machine learning model is not fully compatible to generate testing performance on real world data. To over come this issue from a one particular angle is to use noise as a generalising technique.

Acknowledgment

References

[1] "Agriculture market analysis, size and trends global forecast to 2022-2030," Apr 2022. [Online]. Available: https://www.thebusinessresearchcompany.com/report/agriculture-global-market-report

[2] "Www.aptrc.asn.au." The World Processing Tomato Council, Jan 2020. [Online]. Available: https://www.aptrc.asn.au/wp-content/themes/Divi/includes/builder/frontend-builder/assets/vendors/

[3] I. K. Arah, H. Amaglo, E. K. Kumah, and H. Ofori, "Preharvest and postharvest factors affecting the quality and shelf life of harvested tomatoes: A mini review," *International Journal of Agronomy*, vol. 2015, p. 1–6, 2015.

[4] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv.org*, Apr 2017. [Online]. Available: https://arxiv.org/abs/1704.04861v1

[5] M. Agarwal, A. Singh, S. Arjaria, A. Sinha, and S. Gupta, "Toled: Tomato leaf disease detection using convolution neural network," *Procedia Computer Science*, vol. 167, pp. 293–301, 2020.

[6] S. W. Hasinoff, "Photon, poisson noise." *Computer Vision, A Reference Guide*, vol. 4, 2014.

[7] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010, pp. 242–264.

[8] Q. Ma, L. Jiang, W. Yu, R. Jin, Z. Wu, and F. Xu, "Training with noise adversarial network: A generalization method for object detection on sonar image," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 729–738.

[9] E. E. Kuruoglu and J. Zerubia, "Modeling sar images with a generalization of the rayleigh distribution," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 527–533, 2004.

[10] R. M. Zur, Y. Jiang, L. L. Pesce, and K. Drukker, "Noise injection for training artificial neural networks: A comparison with weight decay and early stopping," *Medical physics*, vol. 36, no. 10, pp. 4810–4818, 2009.

[11] "Agriculture market analysis, size and trends global forecast to 2022-2030," in *The Business Research Company*, Apr-2022, pp. 15–22.

[12] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[13] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[14] L. Chamli Deshan, M. Hans Thisanke, and D. Herath, "Transfer learning for accurate and efficient tomato plant disease classification using leaf images," in *2021 IEEE 16th International Conference on Industrial and Information Systems (ICIIS)*, 2021, pp. 168–173.

[15] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.