Improving In-field Cassava Whitefly Pest Surveillance with Machine Learning

Jeremy Francis Tusubira, Solomon Nsumba, Flavia Ninsiima, Benjamin Akera, Guy Acellam, Joyce Nakatumba
Artificial Intelligence Lab
Makerere University
[tusubirafrancisjeremy, snsumba, delmira91, akeraben, acellam.guy,jnakatumba]@gmail.com

Ernest Mwebaze, John Quinn
Google Research
[emwebaze, jquinn]@google.com

Tonny Oyana
Geospatial Data and Computational Intelligence Lab
Makerere University
toyana@cit.ac.ug

Abstract

Whiteflies are the major vector responsible for the transmission of cassava related diseases in tropical environments, and knowing the numbers of whiteflies is key in detecting and identifying their spread and prevention. However, the current approach for counting whiteflies is a simple visual inspection, where a cassava leaf is turned upside down to reveal the underside where the whiteflies reside to enable a manual count. Repeated across many cassava farms, this task is quite tedious and time-consuming. In this paper, we propose a method to automatically count whiteflies using computer vision techniques. To implement this approach, we collected images of infested cassava leaves and trained a computer vision detector using Haar Cascade and Deep Learning techniques. The two techniques were used to identify the pest in images and return a count. Our results show that this novel method produces a whitefly count with high precision. This method could be applied to similar object detection scenarios similar to the whitefly problem with minor adjustments.

1. Introduction

The whitefly, *Bemisia tabaci*, is the most common pest and plant-viral vector responsible for transmitting cassava diseases such as Cassava Mosaic Disease and Cassava Brown Streak disease in Africa[11]. Studies have shown a correlation between whitefly populations and the spread of these crop diseases[17]. The most commonly used method for monitoring pest populations in crop health surveys is a simple visual inspection – a method that is arduous, time-consuming and prone to human errors. To overcome these limitations, computer vision techniques can be used to efficiently monitor pests in farms and greenhouses. To accomplish this, cameras are used to capture images of the plants, and then image processing techniques are applied to recognize, track and count the pests . Amongst the current automated pest count methods, none can automatically count whiteflies for cassava leaf images.
Although recent methods have been proposed for automatic detection and count of whiteflies on leaves utilizing image processing and machine learning techniques, there is no robust deployable method. According to Fishpool and Burban [6], there are two main existing methods for assessing populations of adult whiteflies. One is to use attractive or non-attractive traps for the flying adults while the other method is to count adults and/or nymphs in situ. The latter method is commonly used in field surveys. It involves the direct count of adults on representative shoots of individual cassava plants. A leaf is held by the petiole and gently inverted so that the adults present on the underside can be counted[5]. This is the basis of our approach. Images of whitefly-infested leaves are captured in this context and our method returns a whitefly count with minimum bounding boxes around the whiteflies detected.

Other studies use image processing methods that employ features such as colour, shape, size and texture to extract whiteflies from the green leaf background [1, 2]. As [7] states, the disadvantage in these automated methods for pest count is that they are site-specific, highly dependent on ground truth data, and not reusable for other situations. This is especially of great concern because whiteflies have a host range of more than 250 plants including cotton, cassava, sweet potato and tomato, which may grow in different site characteristics. A technique developed for one scenario does not necessarily work in another scenario due to differences in the pest site background. To counter this problem, Boissard et al proposed a general detection system, which

is not site-specific to detect whiteflies using size, texture, shape and colour analyses[2]. Based on their results, it is highly evident that evaluations were done on cropped image samples/patches of leaves. By using cropping as a form of background subtraction, it can be said that their evaluation does not take into consideration cases of infested leaf images having complex backgrounds. More so it is not evident whether a concrete evaluation of their method was done on all types of leaves belonging to various plants; therefore it is hard to conclusively state that this approach works in all situations of automatic whitefly pest detection and count.
Machine learning methods have been used for pest recognition, detection and count. To accomplish this, a classifier is trained on image datasets of the pest to learn its features. The trained model is then used to make predictions on new inputs. If the input meets a certain threshold during prediction, the model classifies it as a pest. Sample studies that propose machine learning include the detection and count of aphids in soybean leaves, identification of fruit fly[18], and recognition of vegetable pests[27]. In this study, we focus on machine learning techniques for the detection and count of whiteflies on cassava leaves. The key challenge is classification and localization of the whiteflies on the leaf. We therefore employ and compare two well-known methods for object detection; Haar Cascade Method and Deep Convolutional Neural Networks.

2. Methods and Materials

2.1. Image dataset acquisition and sample preparation

To train the whitefly detectors, 7500 images of whitefly infested leaves were collected from cassava gardens at the National Crop Resources Research Institute (NaCRRI) in Namulonge, Uganda. In whitefly trial fields, cassava plants that are intentionally exposed to whiteflies to conduct different studies of the effect of whiteflies on cassava plants and it is from these fields that pictures of the whitefly infested leaves were taken. During the image capturing process, two human experts randomly selected plants from which the images were taken and each image was taken by turning the fully expanded leaves on a plant to expose the underside where the whiteflies reside and capturing this view. The pictures were taken using a Tecno Spark 3 android smartphone camera with a resolution of 13 megapixels to produce images with a dimension of 4000 x 1920 pixels and all the captured pictures were transferred to a laptop for storage. The images that were taken can be grouped under two major categories; images with a low to moderate abundance (less than 50 whiteflies) and images with moderate to high abundance (more than 50 whiteflies) as shown in Figure 1.

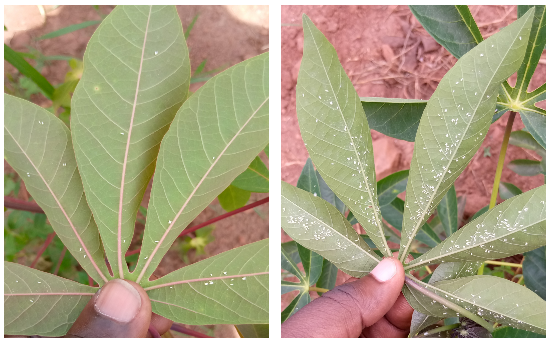

(a) Low—moderate infestation (b) Moderate—high infestation

Figure 1. Images of whitefly infested leaves; (a) Low to Moderate infestation (b) Moderate to high infestation

2.2. Dataset annotation

To train the whitefly detection models, the data was labelled by drawing bounding boxes tightly around the whitefly within an image and a label assigned to the whitefly object. Annotation experiments showed that the time required to annotate an image ranged from 4—15 minutes depending on the abundance of the whiteflies on the image and based on this, annotating the entire dataset of 7500 images would require about 1250 labour hours. Due to a limitation of human resource and time, a set of 2000 images was randomly selected to be annotated as this was deemed to be sufficient dataset for this experiment. Using an opensource tool called LabelImg [1], bounding boxes were drawn around each whitefly in the selected set of images as shown in Figure 2 and annotations were saved in the PASCAL VOC (Visual Object Classes)[4] format. This formed the base data input for both approaches in this experiment.

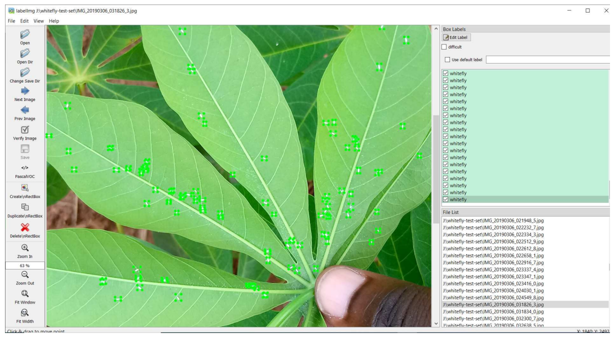


Figure 2. Image annotation using labelImg

2.3. Development of models for automated whitefly count

This section describes our novel methods for detecting the whiteflies on cassava leaf images. The central aim of this method is to accurately detect and localize the whitefly

[1]Tzutalin.LabelImg.Gitcode(2015). https://github.com/tzutalin/labelImg

on a cassava leaf image. We have considered two supervised machine learning techniques due to their high performance in the task of object detection;Haar Cascade classifier and Faster-RCNN ResNet 101 which both require labelled images as training data. Utilizing this labelled data, we adopt these two machine learning techniques to detect the whiteflies on the cassava leaf images. In the next section, we will explain our implementation of the Haar Cascade detector that has shown immense performance in detection of human faces, then we will describe the deep learning approach that was implemented using convolutional neural networks.

2.3.1 Haar Feature-based Cascade Classifier for Object Detection

Object Detection using Haar feature-based cascade classifiers was proposed by Paul Viola and Michael Jones[26]. It is an effective object detection method that has been applied in the detection of faces, road signs, and other objects [14, 22]. A classifier, namely a cascade of boosted classifiers, working with Haar-like features, is trained with a few hundred sample views of a particular object of interest (positive examples) that are scaled to the same resolution e.g. 20 x 20, and negative examples - arbitrary images of the same size. The term ”cascade” is derived from the fact that the resultant classifier will consist of several simpler classifiers (stages) that are applied subsequently to a region of interest until the candidate is rejected or all the stages are passed. On the other hand, ”boosted” refers to classifiers at every stage of the cascade that are complex themselves and are built out of basic classifiers using one of four different boosting techniques: Discrete Adaboost, Real Adaboost, Gentle Adaboost and Logitboost. The basic classifiers are decision-tree classifiers with at least two leaves and the inputs are Haar-like features, such as edges and lines.

2.3.2 Haar Cascade Data Preparation

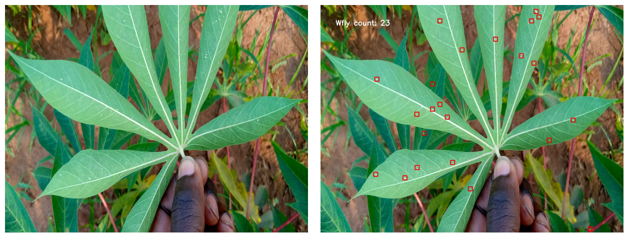
To train the Haar Cascade, positive and negative image samples are required. Positive samples are images that contain the object of interest(whitefly) whereas negative samples are images that do not have any whiteflies. The positive samples were obtained from the base dataset that was annotated in PASCAL VOC with 1800 images and 200 images forming the training and test set respectively. Negative samples were generated by creating 200×200 pixel image patches that did not contain whiteflies from the annotated images. A total of 17250 negative samples were created and the training and test set sizes were 12436 and 4814 images respectively. To increase variation and size of the positive sample dataset, new samples were created by rotating the positive samples 10 times at an interval of 36 degrees.

2.3.3 Configuration and implementation of the Haar Cascade Classifier

To train a Haar Classifier Rejection Cascade, we used opencv [3, 16] an open-source library used to perform image analysis tasks which provides a utility to implement the Haar Classifier. Opencv defines several parameters that should be specified for training a Haar Cascade classifier and for this experiment the parameters that were used are specified in Table 2.3.4. The training was done on a Linux 64bit computer with 16GB of memory and an Intel i7 processor running at a maximum clock speed of 3.2GHz and the output of the training is a cascade file saved in XML (Extensible Markup Language) format which defines different attributes describing the object of interest.

2.3.4 Inference with the Haar Classifier

The Haar Cascade classifier works by analyzing a region of interest from an input test image and returns 1 if the region of interest is likely to contain a whitefly or 0 otherwise. The region of interest is the same dimension as that defined by the width and height parameters during the cascade training. To search for whiteflies in a whole image, a search window is moved over the entire image and each region is checked by the classifier for the presence of a whitefly. The total number of regions classified as having a whitefly are counted in order to determine the total count of whiteflies on the image. For the bounding boxes to be drawn around the whiteflies, the pixel-co-ordinates of the whitefly regions are extracted using the classifier and a bounding box is drawn around the whitefly on the image as shown in Figure 3



(a) Input test image (b) Detected whiteflies

Figure 3. Inference by Haar Cascade classifier; (a) Input test image and (b) red bounding boxes drawn around detected whiteflies.

2.3.5 Deep Learning approach to whitefly detection

Deep Learning is an approach that has significantly advanced the field of machine learning. In particular, Convolutional Neural Networks, which are deep learning algorithms optimized for image processing, have greatly improved the field of computer vision to the extent that some previously complex image recognition tasks can now be

Parameter	Value
Width of training samples	20
Height of training samples	20
Number of cascade stages	20
Minimal hit rate per classifier stage	0.95
Maximal false alarm rate	0.5
Number of positive samples	9000
Number of negative samples	12436

Table 1. Training parameters used for the Haar Cascade classifier.

achieved with human-level accuracy[8, 9]. The whiteflies, being small white patches on green backgrounds, posed as a good problem on which to use deep learning techniques. Specifically, we formulate this task as an object-detection problem for which we are trying to detect, localize and count the number of whiteflies on a cassava leaf. We use a convolutional neural network architecture called Faster Regional proposal Convolutional Neural Network (Faster R-CNN)[19] with the ResNet 101 [10] backbone as our feature extractor.

2.3.6 The Faster-CNN Architecture

We employed a regional proposal convolutional neural network – Faster R-CNN[19] for the task of localization and detection of the whitefly on a cassava leaf. This model has shown good performance in many detection problems [9, 24, 20, 28]. Its detection pipeline consists of two stages:

1. The regional proposal network: A set of category-independent object proposals are generated using selective search. The output of the region proposal network is several boxes (proposals) which form feature maps that are later used by the neural network to check the occurrence of objects. A process called Region of Interest pooling (ROI Pooling) is performed to ensure that the generated feature maps are all of the same sizes. ROI pooling does this by splitting the input feature map into a fixed number of roughly equal regions. Max Pooling - a technique to reduce the dimensions of the feature map is then applied to every region.

2. The second part of the model's architecture consists of a convolutional neural network that uses these proposals to perform classification. In our approach, we adapt the fully convolutional Residual Network (ResNet 101) [10] as the backbone network. Particularly, we exploit transfer learning by using the ResNet model pre-trained on the COCO (Common Objects in Context) dataset [15] which helps us achieve higher accuracy levels and faster training time as compared to training a model from scratch.

Parameter	Value
Momentum optimizer	0.9
Learning rate	0.0003
Batch Size	1
Image size	1024×768

Table 2. Training parameters specified for the Faster-RCNN model.

2.3.7 Data preparation for Faster-RCNN

Input data for this method was involved converting the images and their corresponding annotations stored as XML files from the base dataset into TF (Tensorflow) Record files. This specific data format is required as input for training when using the Tensorflow Object detection API (Application Programming Interface) utility which was utilised for this experiment. To evaluate both models on the same set, a test TF record was first created from the same 200 images used as the test set for the Haar Cascade. The rest of the images were used to create train and validation TF Records with the training set having 1600 images and the validation set having 200 images.

2.3.8 Configuration and Implementation of Faster-RCNN Model

For this experiment, the Faster R-CNN implementation provided by the Tensorflow Object detection API, a Deep Learning framework by Google was used to train and evaluate the model. In particular, we utilized a machine learning technique called transfer learning which enables utilization of CNN's previously used for different imaging task to be adopted for use in new sometimes unrelated tasks [23]. In this case, the Faster-RCNN model used was trained on the Microsoft COCO dataset. Fine-tuning is an important step and a best practice when using a transfer learned model. It gives the ability to share the connection weights between a previously learned model to the new model, retraining it to be adapted to our task. However, even though several studies have reported the effectiveness of fine-tuning in deep learning[13, 12, 25], it is not always clear which layers of CNN should be retrained. In this study, all layers except the fully connected layer were re-trained without freezing and the weights of the pre-trained model were used as initial weights. The Tensorflow object detection API provides a number of parameters that can be defined for training a model and Table2.3.8 shows the parameters that were specified for this experiment.

To increase the sample size and variation, random horizontal flipping of images was applied to augment the input data and the model was trained for 10276 time-steps using the Google cloud platform Machine Learning Engine and tensorboard was used to monitor the training process.

The output of the the training process using the Tensorflow object detection API is a set of checkpoint files containing learned features from the input dataset. To create a model, the checkpoint files a frozen into protobuff file which is a standard representation of machine learning models when using Tensorflow.

2.3.9 Inference with Faster-RCNN model

The Tensorflow object detection API provides a set of python scripts which can be used to detect and count the number of whiteflies from a given set of input images and a frozen model. The script loads the image into the model and the model returns co-ordinates of regions in the test image where whiteflies have been detected and the respective probability of the presence of a whitefly in each region. A probability threshold was set at 0.5 and bounding boxes were drawn for the co-ordinates of regions that met the threshold as shown in Figure 4 and the corresponding whitefly count was determined by the total number of such regions.

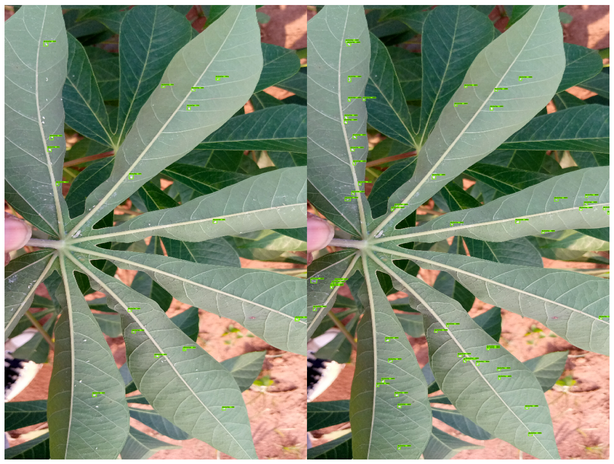


Figure 4. Inference by Faster-RCNN model (right) ground truth annotations (left) detected boxes.

2.4. Evaluation of Models

The whitefly counting task can be approximated to a binary classification task where the goal is to classify an object as a whitefly or not. Following from notion, one of precision and recall are a good measure of how well a classifier is performing[21].
Precision is a measure of the ratio of the true positives detected by the model to the total number of detections made and can be summarized by Equation 1. Recall measures how many true detections were identified by the model as shown in Equation 2. Precision and Recall can be combined to generate the F1 score Equation 3 which gives a better picture of the overall performance of the model.

$$\text{Precision} = \frac{Number\,of\,true\,detections}{Total\,number\,of\,detections} \tag{1}$$

$$\text{Recall} = \frac{True\,detected\,objects}{True\,detected\,objects + True\,undetected\,objects} \tag{2}$$

$$\text{F1 Score} = \frac{2 \times (Precision \times Recall)}{Precision\ +\ Recall} \tag{3}$$

$$\text{IOU} = \frac{Area\,of\,overlap}{Area\,of\,intersection} \tag{4}$$

The Tensorflow object detection API also caters for evaluation of model training jobs using the standard COCO evaluation metrics [15] and MAP (Mean Average Precision) is a major measure of how well an object detection task is performing. The precision of a single detection can be measured using IOU (Intersection over Union) shown in Equation 4, which is the ratio of the area covered by annotation the bounding box to the area of the union between the annotation bounding box and the models detected bounding box for an object. mAP is calculated by getting the average IOU for all detections made and in this case we considered an IOU threshold ranging from 0.5—0.95.

3. Results

3.1. Performance of Faster-RCNN model and Haar Cascade Classifier

The models were evaluated against each of their respective test datasets containing 200 images and their corresponding predictions were obtained. The Precision was used to determine how many actual whiteflies our models predicted and the Recall to determine how the models wrongly classified objects as whiteflies. Using a python script, the number of detections that overlap with annotated bounding boxes from the base dataset was calculated to determine the TP (true positives) for the Haar Cascade detector and the number of non-overlapping detections formed the FP (false positives). Annotated bounding boxes that do not have an overlapping detection box make up the FN (false negative) count. The true positive count, false positive count and false negative counts for the Faster-RCNN model were generated with the help of the Tensorflow Object detection API and results are presented in Table 3.1. During training, the Faster-RCNN model achieved an optimum mAP value of 0.62 (@0.5IOU) on the validation set.

The average time taken by each of the models to analyze an image was recorded with the Haar Cascade classifier taking about **2.1** seconds compared to **4.7** seconds by the Faster-RCNN model.

	Haar Cascade	Faster-RCNN
TP	3950	5617
FP	831	140
FN	2997	1330
Precision	0.83	0.98
Recall	0.57	0.81
F1 Score	0.68	0.89

Table 3. TP, FP, FN, Precision, Recall and F1 Score for Haar Cascade classifier and Faster-RCNN model.

3.2. Discussion of Results

This experiment compares two machine learning approaches for whitefly counting on cassava leaf images. The Precision values calculated show that the Faster-RCNN model (0.98) performs better than the Haar Cascade classifier (0.83) and it can correctly detect a higher number of whiteflies as illustrated by the true positive count. The Faster-RCNN model also records a much lower count of false positives compared to the Haar Cascade which implies that the model can extract more robust features that are representative of whiteflies than the Haar classifier which is characteristic of convolutional neural networks. Despite the difference in performance, the scores show that both models still detect whiteflies in the test images with a high precision.
Both the Faster-RCNN model and Haar Cascade classifier record lower recall values with the Haar Cascade classifier registering an average recall of just 0.57. The Faster-RCNN model on the other hand scored a recall of 0.81 which implies that it misses out fewer whiteflies than the Haar classifier as shown by the false negative values in Table 3.1.
Considering the F1 score, the results show that the Faster-RCNN model overall performs better than the Haar Cascade classifier at the task of detecting whiteflies when compared using our test set of 200 images.
Although the Faster-RCNN model achieved a slightly above mAP value of 0.61(@0.5IOU), this performance was considered to be very good given the fact that whiteflies are very tiny objects which therefore have small bounding boxes and this makes it difficult to achieve very high IOU values when compared to object detection tasks involving bigger objects.

4. Conclusion and Future work.

In this paper we presented a method that can be adopted to automate the task of counting whiteflies on cassava leaves by utilizing machine learning image analysis techniques. Results show that both the Haar Cascade classifier and and Faster-RCNN can accomplish this task with high Precision and therefore the use of machine learning to count whiteflies is a feasible approach. The Faster-RCNN model particularly performed exceptionally well with an F1 score of 0.89 which shows that convolutional neural networks are capable of learning the characteristic features of whiteflies very well.
These models can be deployed for in field use on portable devices such as smart phones which would significantly increase the output of experts as they would be able to count whiteflies from more leaf samples compared to how many can be analyzed using the manual protocol. Although the models may not perform with human level accuracy of counting whiteflies, the high Precision shows that they can be adopted as a semi-automated solution involving both machine and human where the human expert would only focus on counting the false negatives (whiteflies missed out by the models).
Future work shall involve conducting experiments to evaluate the use of a custom feature extractor for the Faster-RCNN model instead of transfer-learning with ResNet 101. Although we scored a high precision with transfer learning, we believe that by training a a custom classifier to detect whether an image patch contains a whitefly or not, we can achieve a higher precision with this classifier as a back bone for Faster-RCNN. The models shall also be deployed as usable solutions native applications that can be used on smart phones and as native desktop applications. API's shall also be created to serve the models for others that may want to integrate whitefly counting in their tools.

5. Acknowledgement

We thank the National Science Foundation and the Bill and Melinda Gates foundation for funding this project. We also thank the National Agriculture Resources Research Institute (NaCRRI) whose experts have continued to provide support to this project.

References

[1] C Bauch and T Rath. Prototype of a vision based system for measurements of white fly infestation. In *International Conference on Sustainable Greenhouse Systems-Greensys2004 691*, pages 773–780, 2004. 1

[2] Paul Boissard, Vincent Martin, and Sabine Moisan. A cognitive vision approach to early pest detection in greenhouse crops. *computers and electronics in agriculture*, 62(2):81–93, 2008. 1, 2

[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 3

[4] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, Jan. 2015. 2

[5] Denis Fargette, Claude Fauquet, and J-C Thouvenel. Field studies on the spread of african cassava mosaic. *Annals of Applied Biology*, 106(2):285–294, 1985. 1

[6] LDC Fishpool, C Burban, et al. Bemisia tabaci: the whitefly vector of african cassava mosaic geminivirus. *Tropical Science*, 34(1):55–72, 1994. 1

[7] Sara Ghods and Vahhab Shojaeddini. A novel automated image analysis method for counting the population of whiteflies on leaves of crops. *Journal of Crop Protection*, 5(1):59–73, 2016. 1

[8] Chris Gibbons, Suzanne Richards, Jose Maria Valderas, and John Campbell. Supervised machine learning algorithms can classify open-text feedback of doctor performance with human-level accuracy. *Journal of medical Internet research*, 19(3):e65, 2017. 4

[9] Seung Seog Han, Gyeong Hun Park, Woohyung Lim, Myoung Shin Kim, Jung Im Na, Ilwoo Park, and Sung Eun Chang. Deep neural networks show an equivalent and often superior performance to dermatologists in onychomycosis diagnosis: Automatic construction of onychomycosis datasets by region-based convolutional deep neural network. *PloS one*, 13(1), 2018. 4

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4

[11] Alana Lynn Jacobson, Siobain Duffy, and Peter Sseruwagi. Whitefly-transmitted viruses threatening cassava production in africa. *Current opinion in virology*, 33:167–176, 2018. 1

[12] Christoph Käding, Erik Rodner, Alexander Freytag, and Joachim Denzler. Fine-tuning deep neural networks in continuous learning scenarios. In *Asian Conference on Computer Vision*, pages 588–605. Springer, 2016. 4

[13] Shin Kamada and Takumi Ichimura. Fine tuning of adaptive learning of deep belief network for misclassification and its knowledge acquisition. *International Journal of Computational Intelligence Studies*, 6(4):333–348, 2017. 4

[14] Yunyang Li, Xin Xu, Nan Mu, and Li Chen. Eye-gaze tracking system by haar cascade classifier. In *2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA)*, pages 564–567. IEEE, 2016. 3

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 4, 5

[16] OpenCV. *The OpenCV Reference Manual*, 2.4.13.7 edition, April 2014. 3

[17] DSO Osiru, WS Sserubombwe, P Sseruwagi, M Thresh, and GW Otim-Nape. Effects of cassava mosaic virus disease on the growth and yield of cassava-some highlights from makerere experiments. *African Crop Science Journal*, 7(4):511–522, 1999. 1

[18] Thainan B Remboski, William D de Souza, Marilton S de Aguiar, and Paulo R Ferreira Jr. Identification of fruit fly in intelligent traps using techniques of digital image processing and machine learning. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, pages 260–267, 2018. 2

[19] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 4

[20] Ruhan Sa, William Owens, Raymond Wiegand, Mark Studin, Donald Capoferri, Kenneth Barooha, Alexander Greaux, Robert Rattray, Adam Hutton, John Cintineo, et al. Intervertebral disc detection in x-ray images using faster r-cnn. In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 564–567. IEEE, 2017. 4

[21] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3), 2015. 5

[22] Burcu Kır Savaş, Sümeyya İlkin, and Yaşar Becerikli. The realization of face detection and fullness detection in medium by using haar cascade classifiers. In *2016 24th Signal Processing and Communication Application Conference (SIU)*, pages 2217–2220. IEEE, 2016. 3

[23] Hoo-Chang Shin, Holger R Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging*, 35(5):1285–1298, 2016. 4

[24] Xudong Sun, Pengcheng Wu, and Steven CH Hoi. Face detection using deep learning: An improved faster rcnn approach. *Neurocomputing*, 299:42–50, 2018. 4

[25] Edna Chebet Too, Li Yujian, Sam Njuki, and Liu Yingchun. A comparative study of fine-tuning deep learning models for plant disease identification. *Computers and Electronics in Agriculture*, 161:272–279, 2019. 4

[26] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, 2001. 3

[27] Deqin Xiao, Jianzhao Feng, Tanyu Lin, Chunhua Pang, and Yaowen Ye. Classification and recognition scheme for vegetable pests based on the bof-svm model. *International Journal of Agricultural and Biological Engineering*, 11(3):190–196, 2018. 2

[28] Liqin Yang, Nong Sang, and Changxin Gao. Vehicle parts detection based on faster-rcnn with location constraints of vehicle parts feature point. In *MIPPR 2017: Pattern Recognition and Computer Vision*, volume 10609, page 106091J. International Society for Optics and Photonics, 2018. 4