Deep Learning approach to Tomato Plant Diseases Recognition and related Sensitivity Analysis DL-IC 2018 Project

Alessandro Erba Mirco Manzoni
Giuseppe Mascellaro
Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milan, Italy
{alessandro2.erba, mirco.manzoni, giuseppe.mascellaro}@mail.polimi.it

Abstract

Most of the proposed approaches to Tomato plant diseases recognition are based either on a small dataset or on PlantVillage dataset which contains leaves images on a table. In this work we use Convolutional Neural Networks trained on PlantVillage dataset reaching an accuracy of 99.85%*, outperforming state-of-the-art results. Then our contribution is to overcome the dataset limitations proposing a reasonable and robust variation of it having in mind a realistic deployment environment.*

1. Introduction

The advances of science and technology in history have given the possibility to produce enough food to meet the demand of more than 7 billion people. However, food provisioning is threatened by a number of factors such as climate change [1], the decline in pollinators [10], plant diseases [20], and others. Plant diseases are not only a threat to food security at the global scale, but can also have disastrous consequences for smallholder farmers whose livelihoods depend on healthy crops. In the developing world, more than 80 percent of the agricultural production is generated by smallholder farmers [26], and reports of yield loss of more than 50% due to pests and diseases are common [4]. Various efforts have been developed to prevent crop loss due to diseases based on pesticide usage. Independent of the approach, identifying a disease correctly when it first appears is a crucial step for effective and efficient disease management [13].

Historically, diseases identification has been supported by agricultural experts directly on the field. In recently times, these efforts have been additionally supported by leveraging the increasing of Internet penetration worldwide with on-line diagnoses and the tools based on mobile phones, taking advantage of the rapid uptake of mobile phones technology in all parts of the world [11]. These factors, together with advances in computer vision and machine learning, lead to a situation where disease diagnosis based on automated image classification, if technically feasible, can be made available at an unprecedented scale and cost-effectiveness.

On this line, our work focuses first on a Deep Learning approach to disease identification task of 10 tomato plant classes (1 healthy and 9 diseases) using PlantVillage [19] dataset (leaves images) and secondly on the targeted sensitivity analysis of the dataset which has been, in fact, used in state-of-the-art related works [2, 23]. The rationale behind the choice of a particular specie of plants (i.e., tomato plants) is that farmers do know what their plantations are about, hence we exploit this fact as prior knowledge.

As first step we reproduce the experiments of [2] improving their performance results. Then, we show how the learned models perform against variations of input images. This analysis is further supported by the usage of visualization techniques. Finally, on that basis, we conduct a sensitivity analysis of the dataset by building ad hoc variations of it. The analysis of these variations gives insights on actual robustness of the dataset. Indeed, the experiments are pursued keeping in mind realistic deployment environments in which the prediction phase will be performed (i.e., images taken from plantations, greenhouses, and so on). On this assumption we show that the dataset has some not negligible limitations. In light of this we propose a reasonable image augmentation choice that lets the dataset be more robust to various deployment environments.

2. Related work

Several works have been proposed in the literature to plant diseases identification. The classical approach given by the expertise support directly on the field has offered diverse solutions that show outstanding performance, nonetheless they do not provide yet a scalable and cost-effective solution [15, 17, 18].

After analysis of their work and investigation presented by the authors of [16, 21], it has been decided to employ the image processing approach among other laboratory-based approaches. Several handcrafted feature-based methods have been widely applied specifically for image processing. These methods are usually combined with classifiers from machine learning (e.g., Support Vector Machines (SVMs) [5], K-Nearest Neighbors [6], Random Forests [3], and so on). In [22], the authors have presented a survey of well-known conventional methods for handcrafted feature extraction.

The main drawback of these methods regards feature engineering, that is a complex and time-consuming process which needs to be revisited every time according to the problem at hand. Thus, the performance of classifiers depends heavily on the underlying features. For these reason, deep learning has allowed researchers to consider and design systems as a unified and automated process with no *handcrafted* features [14]. In particular, Convolutional Neural Networks (CNNs), first introduced in [28], have showed, in fact, how to bind together feature extraction to classification in image recognition by means of LeNet architecture. Tremendous achievements have been made by CNNs in the past few years in image classification and benchmarked, for instance, against ImageNet dataset [12].

The principles of CNNs have spread also to plant diseases identification. The authors of [2], have proposed a comparison between shallow models with combined handcrafted features and deep models for the identification of 9 tomato plant diseases. They have analyzed different models using PlantVillage dataset containing 14,828 images of cropped leaves put on a table. The best shallow model has achieved 95.47% accuracy using Random Forests, while the best deep model has achieved 99.19% accuracy using GoogLeNet architecture (pre-trained on ImageNet). Furthermore, they have used Occlusion [29] as model visualization method and on its basis they claim that backgrounds in images do not influence the prediction of their best model. In other related works [8, 23] and in our, we argue that this is not always true.

The work proposed in [23] have focused on developing deep models for the identification of 38 classes (i.e., crop and disease information) using PlantVillage dataset (54,306 images of leaves). They have compared several models having various hyper-parameters and dataset variations (RGB, gray-scale, segmented). Their best result, 99.34% accuracy, has been achieved by GoogLeNet (pre-trained on ImageNet) employing RGB images. They noticed that their best model, when tested on a set of images (derived from trusted Internet sources) taken under conditions different from the images employed for training, determines a substantial accuracy reduction, to just above 31%. According to the authors, this limitation is caused by homogeneous background in the images.

A more robust approach has been pursued in [8] where the authors proposed a robust deep-learning-based detector for real-time tomato diseases and pests recognition. They have built their own dataset of 5,000 images taken under different conditions and scenarios divided in 9 classes (and the backgrounds class). Then, with the support of experts, they manually annotated the areas of every image containing the disease or pest with a bounding box and class. Finally, they have analyzed several models for object detection and achieving outstanding performances. Interestingly, they criticize PlantVillage dataset since the images it contains have been previously cropped in the field and captured by a camera in the laboratory causing image recognition and object detection on realistic environments images unfeasible. Instead, their work has aimed at dealing with background variations mainly caused by the surrounding areas of plants or the place itself (i.e., images taken from plantations, greenhouses, and so on).

According to the complains that have been made about PlantVillage dataset, our work provides a reasonable heuristic solution to overcome its limitations for what backgrounds are concerned.

3. Proposed approach

Hereby, we present how we tackle the challenges set up in the introduction.

3.1. Workflow overview

First, we inspect the dataset by manually visualizing the images and by counting them for each class. Then, as showed in Figure 1, we split the dataset (70% train and 30% test) so that we can train the model and assess its final performance on independent data.

As second step, we enhance train data by applying augmentation techniques. By doing this we let the model learn also slight modification of available data allowing for better generalization capabilities. Optionally, we can make further transformations according to a previous sensitivity analysis of the dataset. Important to notice is that, at this stage, all data splits are normalized.

After that, we set up the cross-validation framework by randomly dividing the data into train and validation sets (since we have a large number of samples, we can assume that each class has a sufficient number of representative samples, i.e., stratification). Cross-validation method is used to estimate the performance of the model on unseen samples and, hence, its generalization capabilities. At this point, once hyper-parameters are chosen, we can run the training phase. Hyper-parameters choice is of tremendous

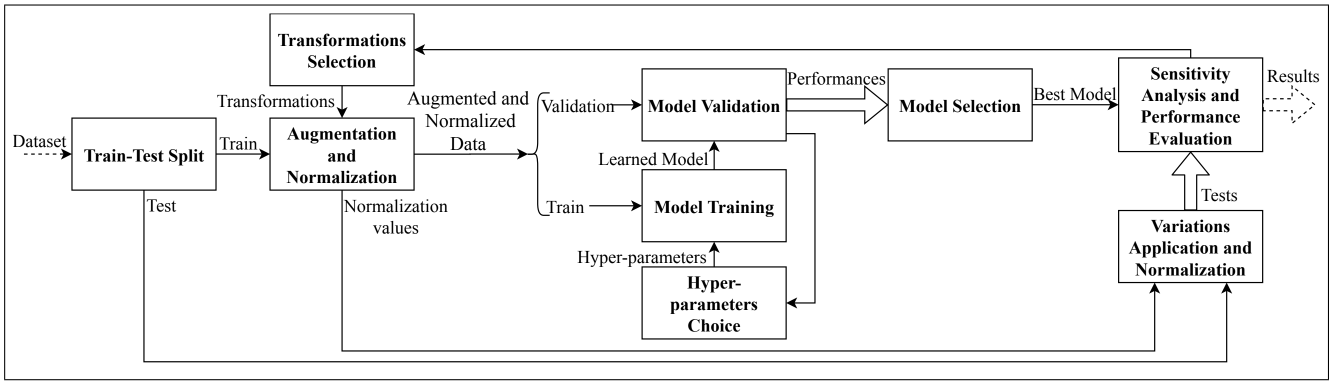

Figure 1. *Workflow overview schema.*

importance as it largely affects the resulting model in both qualitative (the model may learn different representations) and computing time terms. The main ones involve the architecture of the model, the learning rate and its decay speed, the batch size, the optimizer and its arguments choice, regularization type and its value choice, the number of epochs. In our experiments we search the space of hyper-parameters guided by the model validation performances.

Once a satisfactory number of trials is made, we move on to the selection of the best model according to the performances' estimates made by model validation. After that, as a further model performance evaluation, we test it against test data coming from the original dataset and possibly with the application of several variations so to better assess its generalization capabilities. This is further supported by model visualization on test datasets. At this stage, according to sensitivity analysis and performance evaluation, we may decide to make ad hoc modifications of the transformations we apply to the data in earlier stages, and then go through the workflow again.

3.2. The classification task

On the basis of the set up framework, we conduct several classification experiments. For each experiment we take note of the confusion matrices on both validation phase – for model selection – and tests phase – for sensitivity analysis and performance evaluation. In order to assess the quality of the results and to summarize them, we use metrics of performance commonly employed within classification problems [25]. Measures for multi-class classification are based on a generalization of binary classification measures for C classes. M index represent macro-averaging over the C classes.

3.3. Model visualization

Sensitivity analysis and performance evaluation are not only dealt with metrics of performance but also by visualizing the model using the following approaches.

- **Kernels visualization** of the first (it is the most interpretable) layer of a CNN is useful because well-trained models usually display nice and smooth filters without any noisy patterns. Noisy patterns can be an indicator of a network that has not been trained for long enough, or possibly a very low regularization strength that may have led to overfitting.
- **Gradient-weighted Class Activation Mapping (GradCAM)** uses the gradients of any target concept, flowing into the final convolutional layer to produce a coarse localization map highlighting the important regions in the image for predicting the concept. Among the other properties, this method helps in achieving generalization by identifying dataset bias [24].
- **Occlusion method** acts by systematically occluding different portions of the input image, and monitoring the output. Therefore, if the model is correctly localizing the relevant objects, the probability of the correct class drops significantly when the objects are occluded [29].

4. Experiments

Our experiments are about two main tasks: binary and multi-class classification. In this section, first we describe the dataset and the variations created to perform sensitivity analysis. Then, we go trough the experiments setup where we describe how we have conducted the experiments to accomplish the two tasks and finally we show the results.

4.1. Datasets

We analyze 18,158 images of tomato plant leaves, which are split in 10 classes, 9 diseases and 1 healthy. The scope is to predict the correct class of a leaf image. All the images are RGB and sized 256x256 pixels. The classes are: (A) Bacterial Spot $2,127$ samples, (B) Early Blight $1,000$ samples, (C) Late Blight $1,909$ samples, (D) Leaf Mold 952 samples, (E) Septoria Leaf Spot $1,771$ samples,

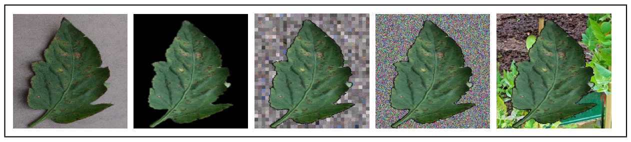

Figure 2. *Samples of dataset images, from left to right:* ***Original****,* ***Segmented****,* ***Random Background Crop****,* ***Random Noise Background*** *and* ***Random Background Image****.*

(F) Two-spotted Spider Mites 1, 676 samples, (G) Target Spot 1, 404 samples, (H) Tomato Yellow Leaf Curl Virus 5, 356 samples, (I) Tomato Mosaic Virus 373 samples, (J) Healthy 1, 590 samples.

Given the original dataset we conduct the training phase on different variations of the original dataset, we work on the backgrounds in order to find a way to overcome the limits highlighted in [8, 23] and confirmed by us. Especially, we have found out a dependence between the original backgrounds and their belonging class. In Figure 2 we can see all the images and the different variations of the dataset we have tested.

- **Original dataset.** It consists of images of tomato leaves laying over a gray table.
- **Segmented dataset.** It consists of images of tomato leaves with no background, this variation has been used also in [23]. In our work we have re-segmented the images since in the available segmented dataset they were incomplete.
- **Random Background Crop dataset.** It consists of the leaf images coming from PlantVillage with a background composed of crops of background coming from the Original dataset.
- **Random Noise Background dataset.** It consists of leaf images coming from PlantVillage with a background composed of random colored pixels.
- **Random Background Image dataset.** It consists of leaf images coming from PlantVillage dataset, each with a background taken from a set of 4,176 images we have built by looking for tomato plantation pictures on the Internet. The objective of this dataset is not only to decouple the background influence on the classifier but also to try to bring the leaf to a realistic situation, where the leaf image is taken directly from the plantation.

4.2. Experiments setup

4.2.1 Data augmentation and normalization

We perform data augmentation applying to each original sample two randomly selected transformations from this pool: Flip Top Bottom, Flip Left Right, Rotate 90°, Rotate 180°, Rotate 270°, Flip Top Bottom and Rotate 90°, Flip Top Bottom and Rotate 270°.

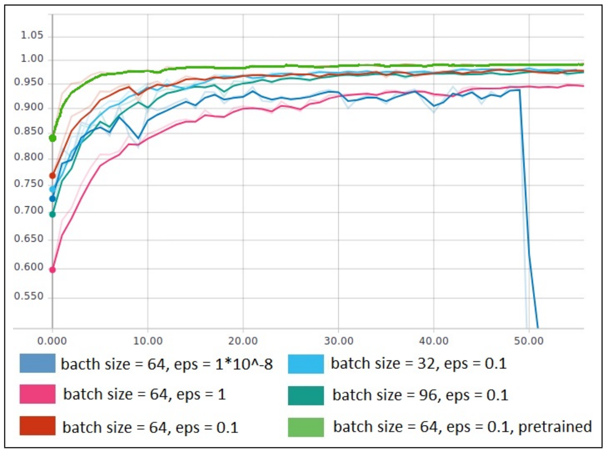

Figure 3. *Validation accuracy over epochs.*

Another key aspect of our setup is the normalization of the input images, indeed, the chosen normalization changes dramatically the model performance. In our work we consider the following normalization approaches:

Dataset Normalization. Given the chosen variation of the dataset we compute the mean and the standard deviation of the train dataset.

Per-leaf Normalization. Starting from the segmented train dataset we compute the mean and standard deviation of the non-black pixels.

4.2.2 Binary classification setup

LeNet architecture. This network, proposed in [28], is the first and simplest deep model available. We have adapted the first and the last convolutional layers to fit our images size and the last fully connected layer to perform our binary task.

Weights initialization. For this network we start by using training from scratch approach. The weights in the network are initialized using Xavier initialization [27]. This helps the training since we start from a normally distributed set of weights and we avoid that certain weighs vanish through epochs.

Activation function. The layers activation function is the rectified linear unit (ReLU) function since it diminishes the likelihood of a gradient to vanish.

Loss function. The chosen loss function for the binary classification is the binary cross-entropy since it is properly designed for binary tasks.

Optimizer. We have chosen Adam with its default parameters. It is based on adaptive estimates of lower-order moments and it is computationally efficient.

Validation. For this task we perform 10-fold cross validation. This kind of validation is used to estimate the model performance and therefore to select the hyper-parameters.

4.2.3 Multi-class classification setup

AlexNet architecture. For this task we choose a more complex model since we have first tested LeNet architecture achieving results far from the state-of-the-art [2].
Weights initialization. For this network we try both training from scratch approach and transfer learning. The weights in the first case are initialized using Xavier initialization ([27]). In the transfer learning approach, instead, we use the weights coming from the training on ImageNet [9].
Activation function. Layers activation function is the rectified linear unit (ReLU) function.
Loss function. In this case we use cross-entropy as loss function since it is largely used for multi-class classification tasks.
Optimizer. As optimizer we have chosen Adam [7]. An important annotation should be done about this optimizer. Indeed, through the training process we have noticed that the accuracy after a certain number of epochs largely drops as we can see in Figure 3. This has been caused by the ε parameter. It is used to avoid division by zero when the gradient is almost zero during Adam parameters update. By default it is set to 10^{-8}, however in some cases it causes large network weights updates. Therefore, we set it to 0.1 in order to avoid this problem. All the other parameters are set to their default values.
Validation. In the case of multi-class classification we opt to proceed with cross-validation. We have chosen a different approach from binary classification. Indeed, k-fold cross validation applied to AlexNet requires massive computational time that is not justified since we have checked that there is a small amount of variance across different validation folds.

4.3. Results and discussion

4.3.1 Binary classification

We set up a binary dataset composed of two classes: healthy (J) and infected (A, B, C, D, E, F, G, H, I). After a training phase using 10-fold cross-validation, we select as best hyper-parameters for our binary classification problem, the batch size equal to 32 and the total number of epochs equal to 15. With this configuration, after re-training the model on the whole train set, we obtain an average accuracy of 99.92%. As expected we obtained the healthy class accuracy lower than the infected one because, since the dataset is unbalanced, the model sees more infected samples. As a further analysis, we have to consider that for our unbalanced dataset the baseline accuracy is 90%, therefore just looking at the accuracy is not enough in this case. The F1-score (99.50%) can better assess the performance of the model, which are satisfactory.

Train	Test	Performance
Original	Original	$AverageAccuracy = \mathbf{0.99856}$ $Precision_M = \mathbf{0.99834}$ $Recall_M = \mathbf{0.99817}$ $F_1score_M = \mathbf{0.99825}$
	Segmented	$AverageAccuracy = 0.39017$ $Precision_M = NaN$ $Recall_M = 0.28588$ $F_1score_M = NaN$
Segmented	Original	$AverageAccuracy = 0.61793$ $Precision_M = 0.73020$ $Recall_M = 0.49413$ $F_1score_M = 0.48694$
	Segmented	$AverageAccuracy = 0.96651$ $Precision_M = 0.95952$ $Recall_M = 0.95689$ $F_1score_M = 0.95802$
Random Background Crop	Original	$AverageAccuracy = 0.76918$ $Precision_M = 0.79434$ $Recall_M = 0.673089$ $F_1score_M = 0.67152$
	Segmented	$AverageAccuracy = 0.84804$ $Precision_M = 0.88554$ $Recall_M = 0.73743$ $F_1score_M = 0.73997$
Random Noise Background	Original	$AverageAccuracy = 0.57886$ $Precision_M = 0.66675$ $Recall_M = 0.49010$ $F_1score_M = 0.49930$
	Segmented	$AverageAccuracy = 0.74847$ $Precision_M = 0.82094$ $Recall_M = 0.70987$ $F_1score_M = 0.69742$
Random Background Images	Original	$AverageAccuracy = \mathbf{0.93230}$ $Precision_M = \mathbf{0.92596}$ $Recall_M = \mathbf{0.91333}$ $F_1score_M = \mathbf{0.91639}$
	Segmented	$AverageAccuracy = \mathbf{0.94977}$ $Precision_M = \mathbf{0.94297}$ $Recall_M = \mathbf{0.92846}$ $F_1score_M = \mathbf{0.93450}$

Table 1. *Performance of multi-class classification.* NaN *identifies the presence of never predicted classes.*

4.3.2 Multi-class classification

As we have done for binary classification, we try several configurations by using different hyper-parameters and we test our results on the validation set. As figure 3 shows, the best configuration found is AlexNet pre-trained, with batch size equal to 64, ε equal to 0.1 trained for 57 epochs. We have also tried some configurations using weight decay regularization and changing the initial learning rate. The resulting performances were not as good as the one on the chosen configuration. We think that weight decay did not help to improve performances because we have a fairly large dataset and AlexNet is not enough complex to reach overfitting. Furthermore, in AlexNet there are dropout layers that limit this problem. For what concerns the learning rate, Adam optimizer [7] automatically adapts it after some epochs. Hence, most of the job is done by the optimizer and our selected learning rates involves the first

few epochs of training and not the overall training.
The results obtained in the chosen configuration are shown in the first row of Table 1. AlexNet pre-trained has a very good performance in all the classes and has obtained also better results than the state-of-the-art [2]. We expected that classes with less samples, e.g. (I) Tomato Mosaic Virus, would have been wrongly classified compared to classes with an higher number of samples, e.g. (H) Tomato Yellow Leaf Curl. This is not our case, since our classifier wrongly classifies only some samples that are not related to the size of the predicted class. ===additional material conf matrix?===

4.3.3 Sensitivity analysis

After a good level of accuracy for our multi-class classification task has been reached, we move our attention to the analysis of robustness of our classifier trained on the Original dataset. Hence, we test the model against the Segmented dataset. In particular, in Table 1 we can notice that the accuracy drops and some classes are never predicted causing Precision and F1-score to have NaN value. From this result and from visualization methods we deduced that the model somehow learned patterns in the background. Table 1 describes all the different settings we have tried and the related results. For each approach, we test the model against both Normal and Segmented dataset using Dataset Normalization. First, we use the Segmented dataset for the training phase. Also in this case the trained model has not been able to satisfactorily identify the leaves when tested against the Normal dataset. We have obtained better results using Random Background Crop and Random Noise Background datasets. As final result, we propose the usage of Random Background Images dataset. In this case we use Per-leaf Normalization in order to further make each leaf image independent from its background. The results are shown in Figure 4. This model is the only with an high value on precision and recall, sign of robustness. It is possible to see in Figure 5 the differences on the activation of GradCAM: the first is the outcome of our first model and the activations are distributed both on the leaf and on the background, the latter is the outcome of our chosen model and the activations are located only on the leaf.

5. Conclusion and Future Works

In this study we have proposed deep learning approach to build classifiers for tomato plant diseases recognition and the related sensitivity analysis on PlantVillage dataset. Our results confirms what claimed in [2] that deep models perform well on this task. Furthermore we provide an improved dataset to overcome the problem of homogeneous backgrounds using the sensitivity analysis as highlighted during the work.

A final remark should be made about the results obtained

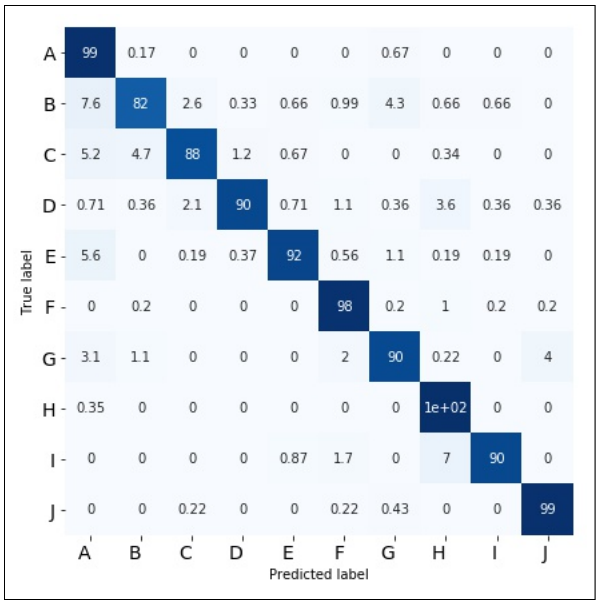

Figure 4. *Confusion matrix of AlexNet pretrained in multi-class classification problem in percentage using per-leaf normalization and random background images.*

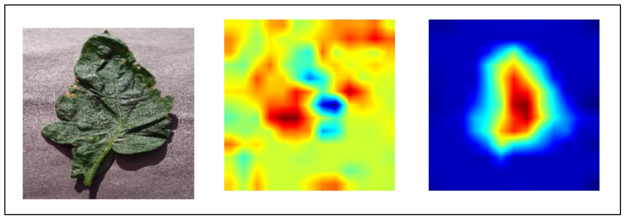

Figure 5. *From left to right: Original image, class: Target Spot, GradCAM on Original dataset, GradCAM on Random Background Image.*

by the Occlusion method proposed in [2]. We have tried to reproduce the same results with our models, but we have not been able at all to come up with the same conclusion about the independence between the class prediction and the background.

In order to develop further assess the performances of our final model on the Random Background Image dataset, we have created a new test dataset containing 256 images found on Internet. The average accuracy we have obtained is 29.8%. Moreover, we have tried to train a VGG11 model on Random Background Image dataset and tested it against the images coming from Internet; the average accuracy we have obtained is 35.8%. We think this result is so poor if compared with the results on Original dataset because no one validated this test dataset. In order to improve this work we suggest to use a dataset of images taken from plantations and validated by a domain expert.

Acknowledgements

aaa

A. Supplementary Material

References

[1] A. P. Tai, M. V. Martin, and C. L. Heald. Threat to future global food security from climate change and ozone air pollution. *Nat. Clim. Change*, 4(9):817–821, 2014. https://doi.org/10.1038/nclimate2317. 1

[2] M. Brahimi, B. Kamel, and A. Moussaoui. Deep Learning for Tomato Diseases: Classification and Symptoms Visualization. *Applied Artificial Intelligence*, 31(4):299–315, Apr. 2017. https://doi.org/10.1080/08839514.2017.1315516. 1, 2, 5, 6

[3] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, Oct. 2001. https://doi.org/10.1023/A:1010933404324. 2

[4] C. A. Harvey et al. Extreme vulnerability of smallholder farmers to agricultural risks and climate change in Madagascar. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 369(1639):20130089–20130089, Feb. 2014. https://doi.org/10.1098/rstb.2013.0089. 1

[5] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep. 1995. https://doi.org/10.1007/BF00994018. 2

[6] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, Jan. 1967. https://doi.org/10.1109/TIT.1967.1053964. 2

[7] J. L. B. Diederik P. Kingma. Adam: A method for stochastic optimization. *international Conference on Learning Representations*, 2015. https://arxiv.org/pdf/1412.6980.pdf. 5, 6

[8] 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*, 17(9), 2017. https://doi.org/10.3390/s17092022. 2, 4

[9] ImageNet. ImageNet dataset. www.image-net.org/. 5

[10] Intergovernmental Science-Policy Platform on Biodiversity and Ecosystem Services Fourth session. *Report of the Plenary of the Intergovernmental Science-PolicyPlatform on Biodiversity and Ecosystem Services on the work of its fourth session*, Kuala Lampur, 2016. https://www.ipbes.net/event/ipbes-4-plenary. 1

[11] International Telecommunication Union (ITU). *ICT Facts and Figures – the World in 2015*, Geneva, 2015. https://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2015.pdf. 1

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira and C. J. C. Burges and L. Bottou and K. Q. Weinberger, editor, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf. 2

[13] L. E. Ehler. Integrated pest management (IPM): definition, historical development and implementation, and the other IPM. *Pest Management Science*, 62(9):787–789, 2006. https://doi.org/10.1002/ps.1247. 1

[14] Y. LeCun, Y. Bengio, and G. Hinton. Deep Learning. *Nature*, 521:436–444, 2015. https://doi.org/10.1038/nature14539. 2

[15] O. W. Liew, P. C. J. Chong, B. Li, and A. K. Asundi. Signature Optical Cues: Emerging Technologies for Monitoring Plant Health. *Sensors*, 8(5):3205–3239, 2008. https://doi.org/10.3390/s8053205. 2

[16] A.-K. Mahlein, E.-C. Oerke, U. Steiner, and H.-W. Dehne. Recent advances in sensing plant diseases. *European Journal of Plant Pathology*, 133, May 2012. https://doi.org/10.1007/s10658-011-9878-z. 2

[17] M. Mazarei, I. Teplova, M. R. Hajimorad, and C. N. Stewart. Pathogen Phytosensing: Plants to Report Plant Pathogens. *Sensors*, 8(4):2628–2641, 2008. https://doi.org/10.3390/s8042628. 2

[18] M. Meroni, M. Rossini, V. Picchi, C. Panigada, S. Cogliati, C. Nali, and R. Colombo. Assessing Steady-state Fluorescence and PRI from Hyperspectral Proximal Sensing as Early Indicators of Plant Stress: The Case of Ozone Exposure. *Sensors*, 8(3):1740–1754, 2008. https://doi.org/10.3390/s8031740. 2

[19] PlantVillage. PlantVillage Dataset. https://github.com/spMohanty/PlantVillage-Dataset. 1

[20] R. N. Strange and P. R. Scott. Plant Disease: A Threat to Global Food Security. *Annual Review of Phytopathology*, 43(1):83–116, Sep. 2005. https://doi.org/10.1146/annurev.phyto.43.113004.133839. 1

[21] P. R. Reddy, S. Divya, and M. R. Vijayalakshmi. Plant disease detection techniquetool-a theoretical approach. *IJITR*, pages 91–93, 2015. 2

[22] T. Reed and J. Dubuf. A Review of Recent Texture Segmentation and Feature Extraction Techniques. *CVGIP: Image Understanding*, 57(3):359–372, 1993. https://doi.org/10.1006/ciun.1993.1024. 2

[23] S. P. Mohanty, D. P. Hughes, and M. Salathé. Using Deep Learning for Image-Based Plant Disease Detection. *Frontiers in Plant Science*, 7:1419, Sep. 2016. https://doi.org/10.3389/fpls.2016.01419. 1, 2, 4

[24] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. *CoRR*, abs/1610.02391, 2016. http://arxiv.org/abs/1610.02391. 3

[25] M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009. https://doi.org/10.1016/j.ipm.2009.03.002. 3

[26] UNEP, International Fund for Agricultural Development (IFAD). *Smallholders, Food Security, and the Environment*, Rome, 2013. https://www.ifad.org/documents/10180/666cac24-14b6-43c2-876d-9c2d1f01d5dd. 1

[27] X. Glorot, Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010. http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf. 4, 5

[28] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324", Nov. 1998. https://doi.org/10.1109/5.726791. 2, 4

[29] M. D. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks. *CoRR*, abs/1311.2901, 2013. http://arxiv.org/abs/1311.2901. 2, 3