# Template for Technical Reports
# 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

*The ABSTRACT should be self contained and explain what the paper is about. Usually abstracts are no longer than 300 words. You should state what are the main contributions of your work and tempt the reader to continue to read your paper. A good abstract briefly describes your problem, approach, and key results. This document is based on the CVPR submission template, and it has been adapted to submit a technical report of a project of the Deep Learning and Image Classification course.*

## 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 [5], plant diseases [11], and others. Thus, these factors cause direct impacts on the population, such as economic, health, and livelihood impacts [14]. 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 [15], and reports of yield loss of more than 50% due to pests and diseases are common [3]. Furthermore, the largest fraction of hungry people (50%) live in smallholder farming households [9], making smallholder farmers a group that is particularly vulnerable to pathogen-derived disruptions in food supply. 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 [7].

Historically, disease identification has been supported by agricultural extension organizations or other institutions, such as local plant clinics that have provided expertise support 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 [6]. 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 dataset (leaves images) [10] and secondly on the targeted sensitivity analysis of the dataset which has been, in fact, used in state-of-the-art related works [2, 12]. The rationale behind choosing a particular specie of plants (i.e., tomato plants) is that farmers do know what their plantations is about, hence we exploit this fact as prior knowledge. Furthermore, Tomato crops are highly affected by diseases, which decrease the yield and cause dramatic losses in agriculture economy [**?**].

As first step we reproduce the experiments of [2] improving their performance results. Then, we show how the learned models respond to input images by using two visualization techniques: Occlusion [17] and GradCAM [13]. Finally, on that basis, we conduct a sensitivity analysis of the dataset by building ad hoc variations of it. These variations, together with models visualization, give 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 such as: hyperspectral proximal sensing techniques to evaluate plant stress to environmental conditions [**?**], optical technologies like thermal and fluorescence imaging methods for estimating plant stress produced mainly by increased gases, radiation, water status, and insect attack, among others [**?**], chemical elements were applied to leaves in order to estimate their defense capabilities against pathogens [**?**]. Previous methods show outstanding performance, nonetheless they do not provide yet a scalable and cost-effective solution [**?**].

After analysis of their work and investigation presented by the authors of [**?**, **?**], it was 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) [4], K-Nearest Neighbors [**?**], Random Forests [**?**], and so on). Some of the best-known handcrafted feature methods which have been proposed for plant diseases identification are: the Histogram of Oriented Gradients (HOG) [**?**] and Scale-Invariant Feature Transform (SIFT) [**?**]; YcbCr, HSI, and CIELB colour models [**?**] effective against noise from different sources; shape feature method to determine leaf and lesioning area [**?**]; texture feature such as inertia homogeneity, and correlation obtained by calculating the gray level co-occurrence matrix (GLCM) [**?**]. In [**?**], GLCM features have been extracted from a dataset of 800 tomato leaf images and classified distinguishing healthy from infected using SVMs with 99.83% accuracy. Combination of more features provides a robust feature set for image improvement and better classification accuracy. In [**?**], the authors have presented a survey of well-known conventional methods for handcrafted feature extraction.

The main drawback of these methods regard 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 depend 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 [8]. In particular, Convolutional Neural Networks (CNNs), first introduced in [16], have showed, in fact, how to bind together feature extraction to classification in image recognition task by means of LeNet architecture. Tremendous achievements have been made by CNNs in the past few years on image classification of, for instance, ImageNet dataset [**?**].

The principles of CNNs have spread also to plant diseases identification. The authors of [**?**], 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 [**?**] 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 and in ours we argue that this is not always true.

The work proposed in [**?**] 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 [**?**] 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.

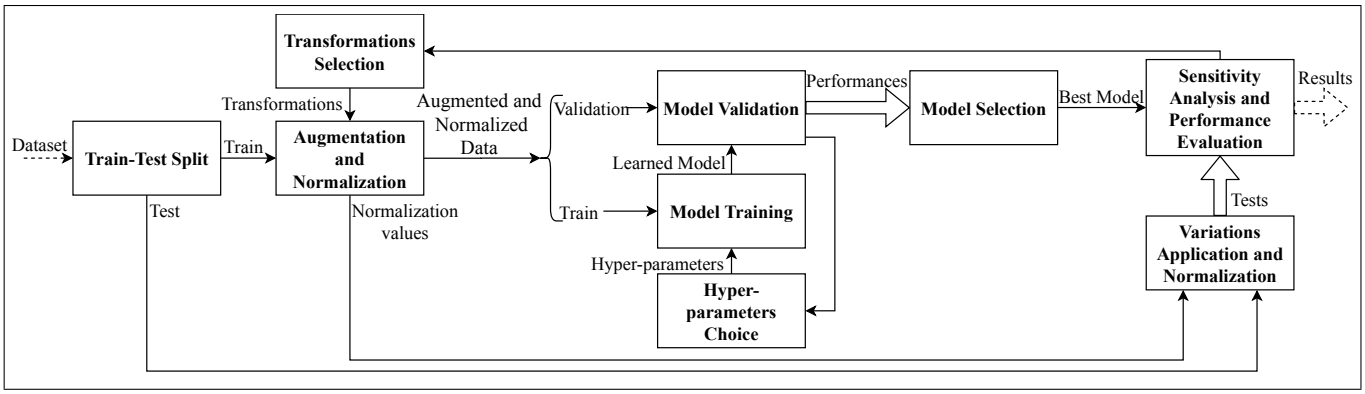


Figure 1. Workflow overview schema.

## 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 random flips and rotations transformations to images. 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 all data splits are normalized by subtracting the mean and dividing by the standard deviation computed on the augmented train data.

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 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 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. Furthermore, we visualize the model on test datasets by using Occlusion and GradCAM methods and its first layer kernels. 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

| Measure | Formula |
|---|---|
| $AverageAccuracy$ | $\frac{\sum_{i=1}^{C} \frac{tp_i+tn_i}{tp_i+fn_i+fp_i+tn_i}}{C}$ |
| $Precision_\mu$ | $\frac{\sum_{i=1}^{C} \frac{tp_i}{tp_i+fp_i}}{C}$ |
| $Precision_M$ | $\frac{\sum_{\mu=1}^{C} Precision_\mu}{C}$ |
| $Recall_\mu$ | $\frac{\sum_{i=1}^{C} \frac{tp_i}{tp_i+fn_i}}{C}$ |
| $Recall_M$ | $\frac{\sum_{i=1}^{C} Recall_\mu}{C}$ |
| $F_1score_\mu$ | $\frac{2 \times Precision_\mu \times Recall_\mu}{Precision_\mu + Recall_\mu}$ |
| $F_1score_M$ | $\frac{2 \times Precision_M \times Recall_M}{Precision_M + Recall_M}$ |

Table 1. Measures of classification performance.

On the basis of the set up framework, we conduct several classification experiments. As first attempt we approach the simple task of binary classification. The two considered classes are healthy and infected. Then, we tackle the multi-class classification problem for tomato plant diseases recognition. 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 ([**?**]).

Measures for multi-class classification, showed in Table 1, are based on a generalization of binary classification measures for C classes: $tp_i$ are true positive for class $i$, and $fp_i$ – false positive, $fn_i$ – false negative, and $tn_i$ – true negative counts respectively. $\mu$ and $M$ indices represent micro- and macro-averaging.

### 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.

- Visualizing the kernels 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.
- Occlusion method gives insights whether the model is truly identifying the location of the relevant object in the image, or just using the surrounding context. This is done 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.
- Gradient-weighted Class Activation Mapping (Grad-CAM) 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.

## 4. Experiments

Our experiments are on two main task: Binary Classification, 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 were we describe how we have conducted the experiments to accomplish the two task and finally we show the results.

### 4.1. Datasets

We analyze 18158 images of tomato plant leaves, which are splitted in 10 classes, 9 diseases and 1 healthy. The scope is to predict the correct class of a leaf given its image. All the images are sized 256x256.

Given the original dataset we have conducted the training phase on different variations of the original dataset, we have worked on the background in order to find a way to overcome the limits highlined in [] and confirmed by us. In Figure 2 we can see all the images the different versions of the dataset that we have tested and built.

- **Original PlantVillage dataset** it consists of images of tomato leaves lain over a gray table.
- **Segmented PlantVillage dataset** it consists of images of tomato leaves without the original background, this variation was used also in []. In our work we have re-segmented the images since in the available segmented dataset was not complete.
- **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 Plantvillage Dataset. We have tried this variation in order to exploit the dependence between original backgrounds and the class we found out in the experiment on the original dataset.
- **Random Noise Backgorund** it consists of leaf images coming from PlantVillage with a background composed of random colored pixels. As before we tried this experiment to exploit the dependence between background and classes higlined in the original dataset.
- **Random Background Image** it consists of leaf images coming form Plant Village with a background taken from a set of 4176 we built looking for tomato plantation images . The objective of this set was not only to exploit the backgorund influence on the classifier but also try to take the leaf in a situation similar to a realistic one, where the leaf image is taken directly in the plantation.

### 4.2. Experiments setup

#### 4.2.1 Data Augmentation and Normalization

As showed in Figure 1 after the Train-Test Split and Transformation Selection Phases our train set goes trough augmentation and normalization.

We have performed data augmentation applying to the original sample two 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.

After augmentation we have three samples for each original image (1 original image + 2 transformed images). We have done this in order to train the networks with more data and strength the obtained model.

Another key aspect of our setup is the normalization of

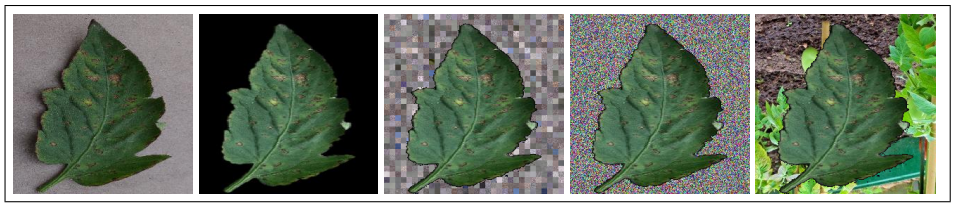

Figure 2. Sampels of training images, from left to right: **Original PlantVillage dataset**, **Segmented PlantVillage dataset**, **Random background Crop dataset**, **Random Noise Backgorund** and **Random Background Image**

the input images, the chosen normalization changes dramatically the validation accuracy. In our work we have explored the space of possible normalization considering these:

- **Dataset Normalization** given the chosen variation of the dataset we have computed the mean and the standard deviation of the whole dataset.
- **Per-leaf Normalization** starting from the segmented dataset we have computed the mean and standard deviation of the non-black pixels.

#### 4.2.2 Binary Classification Setup

**LeNet** This network proposed in [16] is one of the simplest deep model available. We have adapted it to perform our binary task.
**Weights Inizialization:** for this network we started using training from scratch approach. The weights in the net are initialized using Xavier initialization [**?**]. This helps training since we start from a normally distribute set of weights and we avoid that certain weighs vanish trough epochs.
**Activation Function:** layers activation function is rectified linear unit (ReLU) since diminishes the likelihood of a gradient to vanish.
**Loss Function:** the chosen loss function for the binary classification is BCEWithLogitsLoss since it is designed for binary tasks and more stable compared to cross entropy.
**Validation:** for this task we have performed k-fold cross validation with k = 10 over the training set. This kind of validation was used to estimate the number of training epochs and the batch size to use.

#### 4.2.3 Multi-class Classification Setup

**AlexNet** for this task we have chosen a more complex model since we have tested LeNet over the ten class with results very far from the state of the art [2]. In order to compare our results with [2] we have chosen AlexNet.
**Weights Inizialization:** for this network we have tried both training from scratch approach and transfer learning.

The weights in the first case are initialized using Xavier initialization [**?**]. This helps training since we start from a normally distribute set of weights avoids that certain weighs vanish trough epochs.

In the transfer learning approach we have used the weights coming from the training on ImageNet [**?**].
**Activation Function:** layers activation function is again rectified linear unit (ReLU).
**Loss Function:** in this case we have used Cross Entropy as loss function.
**Validation:** in the case of Multi-class Classification we opt to proceed with cross validation. We have chosen a different approach from binary classification. In fact k-fold cross validation applied on AlexNet requires a lot of computing time that is not justified in our dataset since there is small amount of variance in our pictures and consequently in each possible fold.
**Optimizer:** As optimizer we have choosen Adam [**?**]. An important annotation should be done about this optimizer. In fact trough the training process we noticed that accuracy after a bunch of epochs drops. This was caused by the $\varepsilon$ coefficient, this coefficient used to avoid division by zero when the gradient is almost zero in parameters update. By default is set to $1*10^{-8}$ and it causes large weights updates. So we set it to 0.1 to avoid this problem.

### 4.3. Results and discussion:

#### 4.3.1 Binary Classification:

After a training phase using 10-fold cross validation, we select as best hyperparameters for our binary classification problem, the batch size equal to 32 and the total number of epochs equal to 15. With this configuration, after retraining the net on the global train set, we obtain the results on the test set shown in the table 2. Beeing a very unbalanced dataset because of the number of healthy leaves with respect to the non healthy ones, we expected to have an healthy accurancy lower than non healthy. The result show that our assumptions was correct. In fact LeNet accurancy for healthy aren't good as for non healthy one. By the way the average performances of our net are good.

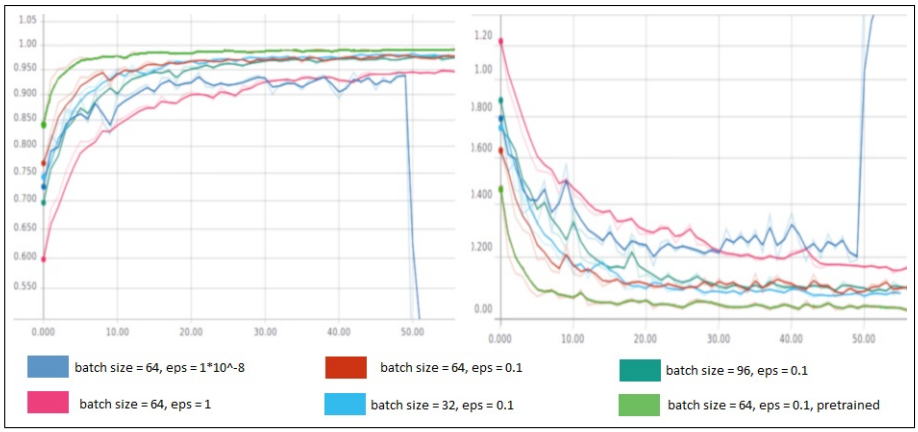


Figure 3. Tensorboard graph of experiments, from left to right: **Validation Accurancy over epochs**, **Valdiation error over epochs**.

| Measure | Results |
|---|---|
| $AverageAccuracy$ | 0.999184 |
| $HealthyAccuracy$ | 0.996678 |
| $NonHealthyAccuracy$ | 0.999407 |
| $Precision_M$ | 0.993377 |
| $Recall_M$ | 0.996678 |
| $F_1score_M$ | 0.995025 |

Table 2. Performance of binary classification.

#### 4.3.2 Multi-class Classification:

As we did for binary classification, we try different run using different hyperparameters and we test our result on the validation set. As figure 3 shows, it's possible to understand that the best configuration it's given from Alexnet pretrained, with batch size equal to 64, eps equal to 0.1 for 57 epochs. The results obtained from this configuration are described in the table 3 and in the figure 4. As the confusion matrix shows, AlexNet pretrained has a very good performance for all the classes and obtained also better result than the state of the art [2]. We expected that classes with less samples, e.g. Tomato Mosaic Virus, would have been missclassified compared to classes with an higher number of samples, e.g. Tomato Yellow Leaf Curl.

| Measure | Results |
|---|---|
| $AverageAccuracy$ | 0.998560 |
| $Precision_M$ | 0.998339 |
| $Recall_M$ | 0.998173 |
| $F_1score_M$ | 0.998254 |

Table 3. Performance of multi-class classification.

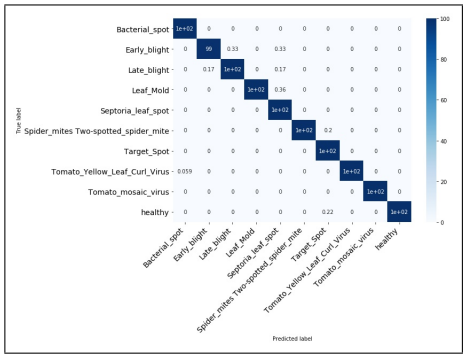


Figure 4. Confusion matrix of AlexNet pretrained in multi-class classification problem in percentage.

#### 4.3.3 Sensitivity Analisys:

After having reached a good level of accuracy for our multi-class classification task, we move our attention to analise the robustness of our dataset by computing gradcam and occlusion methods to analise which parts of the leaf trigger the classifier. The table **??** shows all the different approach we try. It's possible to see that the preformances of the results obtained are very far from the what we have done in the previous task. Sometimes also happens that some classes have never be predicted at all. Then we try in a different way, applying both normalization and backgrounds moving to something similar to what a predictor will see when it's applied on the field. Using per-leaf normalization and random background images we obtain the result shows in figure **??**. It's also possibile to see in figure **??** how the system is more robust to some background variation, activating only on the leaf surface.

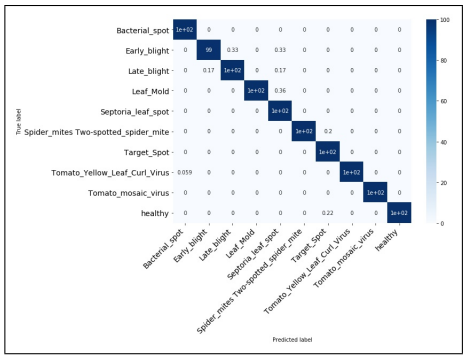


Figure 5. Confusion matrix of AlexNet pretrained in multi-class classification problem in percentage.

| Train | Test | Performance |
|---|---|---|
| Original | Segmented | $AverageAccuracy = 0.390169$<br>$Precision_M = NaN$<br>$Recall_M = 0.285877$<br>$F_1score_M = Nan$ |
| 8*Segmented | Original | $AverageAccuracy = 0.617933$<br>$Precision_M = 0.730197$<br>$Recall_M = 0.494125$<br>$F_1score_M = 0.486940$ |
| | Segmented | $AverageAccuracy = 0.966511$<br>$Precision_M = 0.959547$<br>$Recall_M = 0.956894$<br>$F_1score_M = 0.958019$ |
| 8*Random Background Crop | Original | $AverageAccuracy = 0.769175$<br>$Precision_M = 0.793841$<br>$Recall_M = 0.615085$<br>$F_1score_M = 0.671527$ |
| | Segmented | $AverageAccuracy = 0.848037$<br>$Precision_M = 0.885544$<br>$Recall_M = 0.737432$<br>$F_1score_M = 0.739969$ |
| 8*Random Noise Background | Original | $AverageAccuracy = 0.578862$<br>$Precision_M = 0.666754$<br>$Recall_M = 0.490099$<br>$F_1score_M = 0.499304$ |
| | Segmented | $AverageAccuracy = 0.748469$<br>$Precision_M = 0.820928$<br>$Recall_M = 0.700873$<br>$F_1score_M = 0.697417$ |

Table 4. Performance of multi-class classification with difference training and test sets. We use a NaN value for the class that are never predicted to avoid division by zero in precision, recall and F1 score.

## 5. Conclusion

## 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, 5, 6

[3] 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

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

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

[6] 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

[7] J. 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

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

[9] P. A. Sanchez and M. S. Swaminathan. Cutting World Hunger in Half. *Science*, 307(5708):357–359, Jan. 2005. https://doi.org/10.1126/science.1109057. 1

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

[11] 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

[12] 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

[13] 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. 1

[14] T. Bouley, M. Gilbert, Whung Pai-Yei, F. L. Gall, C. Plante. *Reducing Climate-Sensitive Risks*, volume 1 of *Agriculture and environmental services discussion paper*. World Bank Group, Washington, DC, 2014. http://documents.worldbank.org/curated/en/486511468167944431/Reducing-climate-sensitive-disease-risks. 1

[15] 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

[16] 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, 5

[17] M. D. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks. *CoRR*, abs/1311.2901, 2013. http://arxiv.org/abs/1311.2901. 1