Tip-burn stress detection of lettuce canopy grown in Plant Factories

Riccardo Gozzovelli ⋆, Benjamin Franchetti◦, Malik Bekmurat⋆, Fiora Pirri⋆
⋆ University of Rome Sapienza, ◦ Agricola Moderna

Abstract

A compelling effort has been made in recent years to face several kinds of plant stresses using a variety of sensors and deep learning methods. Yet most of the datasets are based on single leaves or on single plants, exhibiting explicit diseases. In this work we present a new method for stress detection which can deal with a dense canopy of plants, grown in Plant Factories under artificial lights. Our approach combining both classification and segmentation with self supervised masks, and WGAN based data augmentation, has the significant advantage of using normal rgb low cost cameras, simple data aquisition for training and it can both localize and detect the tip-burn stress on the plant canopy with very good accuracy as shown in the results. We have tested our results also on datasets available on tensorflow.org.

1. Introduction

Plant stress detection is a long standing research field and, among the stresses, tip-burn affecting particularly lettuce has been intensively studied, see for example [44, 26, 13]. Nowadays, the combination of new methods arising from computer vision and deep learning, the availability of new low cost sensors together with an increased attention on the transparency, quality and healthiness of the farm to fork process are making plant stress analysis a challenging research topic. New data-sets are being created such as PlantLeaves [11], PlantsDoc [41], PlantsVillage [19] and Plantae-K [47] made available as tensorflow datasets at tensorflow.org. These new data-sets and their ease of accessibility have thrived the research improving deep learning models for the stress detection application.

A limit of the currently available dataset is their inadequateness for stress analysis in Controlled Environment Agriculture (CEA) and specifically in Plant Factories , where plants are grown indoors under artificial lights, densely packed together and stacked on multiple layers. In such highly dense growing conditions the plants are compacted on tables of trays and stress problems need to be studied from this specific perspective, as shown in Figure 1 and Figure 2. The detection and localization of stress in Plant Factories has to deal with complex surfaces agglomerating several plants, where the single leaf shape is not specifically relevant, and at the same time stresses such as tip-burn occur on the leaf tip (see Figure 3). Moreover, typically plants affected by tip-burn are few, sparse and hidden in the canopy of other healthy leaves. The underlying cause of tip-burn is a lack of calcium intake by the plants. This however, is a result of multiple factors such as: lack of airflow, high humidity, excessive lighting and inadequate watering and nutrient supply. A key advantage of growing plants indoors is the possibility to control all aspects of the plant growth including the light recipe and climate, thereby providing the optimal mix of conditions to optimise plant development and quality. However, high-density crop production, limited dimensions, lack of natural ventilation and the need for artificial lighting for photosynthesis makes plants grown in plant factories especially vulnerable to tip-burn. Consequently, tip-burn has become a metric for the healthiness of the plants and being able to monitor its advent

Figure 1. On the left, rolling tables collecting trays at the end of the growth cycle. On the right, inside the production cells in a Plant Factory.

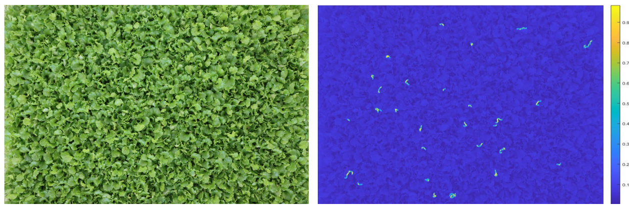


Figure 2. Lettuce canopy and detected tip-burn as a heat map. The bar on the right indicates the color code of the probability the network assigns to the detected regions of being affected by tip-burn. The image has size 4640×6960.

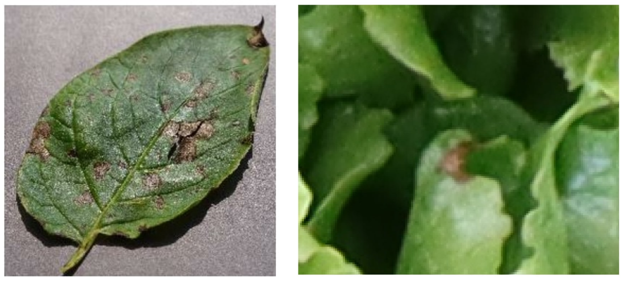


Figure 3. Image of a single leaf taken for data collection purpose (*left*) and in a real-life environment and growing condition (*right*).

is extremely relevant in indoor growing conditions. By automatically detecting tip-burn the vertical farm control software can adjust the growing recipes in real time to provide the plants with the optimal growing conditions.

In this work we propose a novel, and first, model for tip-burn detection in lettuce that fills the gap between already explored techniques of DL applied to plant stress detection and their practical implementation in Plant Factories. Our work includes the realization of an adequate data-set made of real and generated images. Yet, we have tested our model also on other data-sets.

2. State of the art on stress detection and tip-burn

In the past decade Computer Vision and Deep Learning became the new standard for many plant phenotyping tasks, in particular for stress and disease detection and analysis.

Disease Detection Plant disease detection in most recent studies focus on single leaves images, and use hyperspectral cameras. Nagasubramanian et al. [28] achieved charcoal rot disease identification in soybean leaves by implementing a 3D Deep-CNN with an hyperspectral camera. Zhang et al. [49] carried out a similar study using high-resolution hyperspectral images, to detect the presence of yellow rust in winter wheat. Digital cameras are used by [14] and [39]. Dechant et al. [14] consider the classification of the Northern Leaf Blight in maize plants, taking images of leaves in the field. Shrivastava et al. [39] studied the strength of transfer learning for the identification of three different rice plant diseases. A review on computer vision and machine learning methods for disease detection is done in [8].

Tip-Burn studies Tip-burn studies date back long ago [26, 44, 12], essentially exploring causes induced by lack of nutrients absorption, such as in [42] and [48]. As far as we know only [38] conducted tip-burn identification in PFALS using GoogLeNet, for binary classification of single lettuce images. Their work is incomparable with our work, as they check from manually collected images of a single plant whether it has or not tip-burn.

Disease detection on public datasets So far, several datasets have been proposed and well specified for deep learning studies. PlantVillage [19] contains more than 50K low-resolution images of 14 different plant species with 26 stress conditions. In such a case, not only the images of single leaves are taken on a solid background labeled only by a class name (Figure 3), but they also show visible stress conditions at a stage when the leaf is beyond recovery. Agarwal et al. [1] trained a CNN on tomato leaves images taken from the PlantVillage dataset [19] and Saleem et al. [36] realized a comparative evaluation study between multiple CNNs and optimizers - trained again on PlantVillage - for the task of plant disease classification, in order to find the combination with the best performances. Similar to PlantVillage there is PlantLeaves[11] with 4502 high resolution images of healthy and unhealthy leaves divided into 22 categories. Another dataset is PlantDoc[41] in which samples have been collected in a quite realistic setting, with leaves being heavily affected by diseases. A plant diseases dataset using offline augmentation from earlier datasets has been uploaded on Kaggle. This new dataset consists of about 87K images of healthy and diseased crop leaves.

Plants data augmentation with generative models Several studies on plant stress analysis are based on local data collection. An in depth analysis of factors such as lack of adequate datasets influencing the use of deep learning for plants disease detection has been addressed in [6, 7] and some data augmentation methods have been described in [4]. Data augmentation is routinely used in deep learning [24] and augmenting data using GAN has been typically used also for balancing data [27] and cross domain adaptation [18]. In 2017, Odena et al. [30] addressed the problem by proposing AC-GAN. Few years later, a more robust and reliable improvement was presented with CEGAN[43], in which a classifier is trained in combination with a GAN. The presence of the classifier guarantees that the Generator can learn to produce samples that are consistent with their target class. Other specific approaches for data augmentation with GAN have been proposed in [2, 45]. The first introduced DAGAN, while the latter introduced DAG, a data augmentation optimized GAN.

Considerations As far as we know there are no publicly available studies on stress detection of plant canopies grown in Plant Factories. In particular, the only research on tip-burn we found is about binary classification on single lettuce images. Most of the works on plant diseases are carried out just for classification or single leaf disease detection. As a consequence no publicly available canopy datasets exists so far.

3. Method

There are two possible kinds of settings for tip-burn detection: either inside the growing cell (Figure 1) right) or

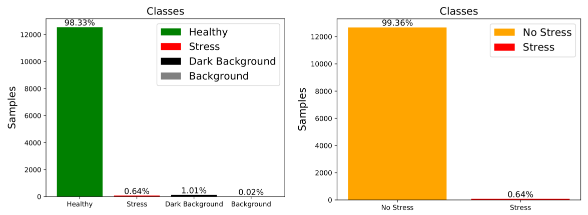


Figure 4. On the left, distribution of class samples in collected dataset. On the right, distribution of class samples after No Stress class grouping.

at harvest (Figure 1 left). We choose to do tip-burn detection at harvest, at the end of the growing cycle. Taking as reference Figure 2, it is easy to see that at the end of the growth cycle the task of identifying tip-burn on large mostly healthy canopy is rather hard, also because tip-burn type of stress affects a very small region of a leaf with respect to its whole area, as shown in Figure 3 (right).

To detect and localize tip-burn on quite large images of lettuce canopies, not having the images labeled, we tile the image into patches, which are used for both samples generation and prediction. In fact, the real problem for tip-burn prediction is not only the lack of labeling but also the unbalanced dataset due to the scarcity of tip-burn samples with respect to healthy plants, as it is shown in the histogram in Figure 4. Our contribution is along two main research lines:

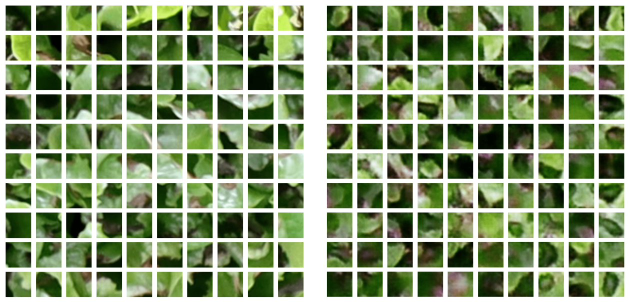
-*Samples generation* based on Wasserstein Gans, to re-equilibrate the dataset. We show that according to the metric to evaluate quality and coverage of the samples produced by GANs, as defined in [25], our method obtains a high value of realism score. The method we provide can generate any amount of patch samples of lettuce with and without tip-burn.

- *Tip-burn segmentation of canopy*. The probability distribution of each patch, being affected by tip-burn, is estimated via YoloV2 backbone darknet-19 [33]. The classification network generates patch-level labels from which we obtain, with further processing and random fields, more accurate labeled regions. From here, with a U-net type segmentation of partial trays into healthy and tip-burn stressed plants, we obtain pixel-level classification leading to a further accurate tip-burn prediction.

Results and ablation studies in Section 6 prove the goodness of our approach, for such a hard problem. Not having available other approaches to compare with, we have used the PlantLeaves dataset [11] and the PlantsVillage dataset [19] to prove the generalization of our method to different settings.

3.1. Data generation

Since tip-burn manifests on the leaves tip, it is mandatory to acquire images with a top view of the whole table.



(a) Real (b) Synthetic

Figure 5. Real and synthetic images generated by the proposed WGAN

We do so by taking images with a HR digital camera fixed above the rolling table. A table is the base on which plants are grown. Each table assembles 18 trays, which in turns are further divided into 104 cells where plant seeds are placed. The image of a table has size (4640×6960) for 3 channels. Each image is tiled into patches of size $(28 \times 28 \times 3)$, forming a set of about $41K$ patches. For tip-burn prediction for the lettuce species, 43 table images have been collected. From these images, only 137 trays were affected by tip-burn. To approximately measure the data unbalance we have sampled from this set about 12200 patches and divided them into four classes: *Healthy, Stress, Background* and *Dark-background*. The distribution of samples in each class is shown in Figure 4.

The histogram clearly shows the disproportion: 98.3% of all samples belong to one class only. If we consider Healthy, Background and Dark-background to be a single class, for easiness let us call it *No Stress*, the percentage reaches 99.36%.

Solving the class imbalance problem The problem of data imbalance in DL is not entirely new and one promising technique that is being widely used for the generation of synthetic samples in real-life datasets are GANs[15]. Inspired by the CEGAN approach [30], we train our own GAN to generate synthetic images of tip-burn and solve the imbalance problem. To our knowledge, this is the first time that such a solution has been tested in the field of plant phenotyping.

The GAN architecture developed follows a DCGAN-like structure[31], but with some differences. First, three strided convolutions, not four, are used for both the generator and the discriminator. Further, we resorted to the Wasserstein GAN[3], or WGAN. We recall that a WGAN, introduces a critic (the discriminator) for the Earth-Mover distance between the distributions of true and generated images, which amounts to measure the cost of transporting one probability on the other. The advantage of the WGAN is that they

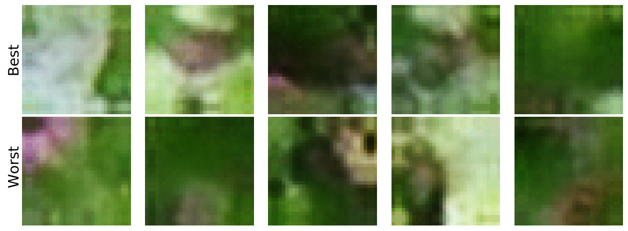


Figure 6. Image patches (28×28) showing the realism score for best 1.9 and worst 0.4, R score

do not require balancing generator and discriminator as the critic (discriminator) does not saturate and can be trained to optimality so that the estimate of the quality of the image from the critic can only improve.

The number of images in which stress occurs and on which the WGAN is trained is 744. An example of the type of images generated at the end of the training procedure, can be seen in Figure 5, where real and synthetic samples are shown side-by-side.

At a first glance, the visual aspect of the synthetic images closely resemble the original ones. However, simple visual quality is not enough to assess the correctness of the generated samples and thus, we explored how to solve this problem in a systematic way.

3.2. Metric assessment of generated data

Recent evaluation metrics for GAN measure the ℓ_2 distance between the real and generated distributions by taking a non-linear embedding η of the real and generated images. The embedding is usually obtained by a CNN pretrained network classifier on ImageNet [37, 16, 35, 25]. Among the different methods comparing the distributions, such as the Inception scores [37], the Frechet Inception distance [16], the Kernel Inception distance [9] and the recent improved (with respect to [35]) precision and recall metric of [25], we have been using this last because it provides also a realism score, which turns out to be relevant for our domain of images, formed by patches showing tip-burn on lettuce.

In [25] samples for the distributions are taken via a non-linear embedding, as gathered above, and the non-parametric densities are estimated by the kNN-distance (the k-nearest neighbour). They use the embedded feature vectors A and B to estimate two manifolds $M(A)$, $M(B)$ separately (see algorithm 1 in [25]). Namely, they estimate an approximation of $M(A)$ according to a $k+1$-minimum distance δ between all feature vectors $a, a' \in A$, with $a \neq a'$ and then compute the number n_B of feature vectors in B that are within the estimated manifold $M(A)$, returning $N_B = n_B/|B|$, $|\cdot|$ being the cardinality of the set. Then they repeat the process to estimate $M(B)$ and the number N_A of feature vectors in A that are within the estimated manifold $M(B)$ returning $N_A = n_A/|A|$. Letting A being the feature vectors from real images, B the feature vectors of generated images and $f(X, Y)$ the indicator function which has value 1 if the estimated manifold $M(X)$ for the features vectors in X return a non-zero N_Y and 0 otherwise, then the precision and recall are defined as:

$$\begin{aligned} \text{precision}(A, B) &= \frac{1}{|B|}\sum_{k=1}^{|B|} f(A, B) \\ \text{recall}(A, B) &= \frac{1}{|A|}\sum_{k=1}^{|A|} f(B, A) \end{aligned} \tag{1}$$

Therefore precision, according to the improved measure of [25], computes the average number of feature vectors from generated images that are found in the estimated manifold of the real images embedding, and the recall computes the average number of feature vectors from real images that are in the estimated manifold of the generated images embedding.

We have fine-tuned VGG-16 network pre-trained on ImageNet to classify real and synthetic stress samples. A dataset of 200 patches is collected and used to train the network for 40 epochs. The feature space is obtained by retrieving the output of the second convolutional layer in VGG-16. The manifold estimation, as described above, is obtained using $k = 3$ nearest neighbour, which is considered a robust choice, while $|A| = |B| = 50176$.

The final precision and recall are then calculated by comparing batches of real and synthetic sample with size 12 and computing the average over all the samples. The synthetic images obtain 39% for precision and 40% for recall. The results obtained confirm that more than $1/3$ of the generated images are realistic and of good quality.

We have also computed the realism score R that increases according to the inverse distance between an image and the manifold, namely the above defined indicator function $f(B, A) = 1$ iff $R(B, A) \geq 1$, that is, at least a feature vector from the embedding of generated images is in the estimated manifold of the real images. In Figure 6 are shown examples of images achieving the highest realism score $R = 1.9$ (top row) and the minimum realism score $R = 0.4$. Since the higher the score the closer is a sample to the manifold estimated from real images, we augmented the original stress dataset only with those images achieving a realism score over a threshold $\tau \geq 1.1$.

4. Tip-burn prediction

Tip-burn prediction, in the absence of labels, requires two steps: 1) a two class classifier returning the tip-burn localization, according to the patch localization in the image, and its probability distribution within the canopy; 2) from patch-level to pixel-level segmentation for a two class

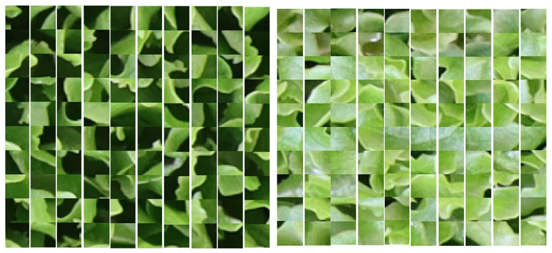


Figure 7. On the left the patches with dark spots and on the right the patches without dark spots.

segmentation of the lettuce canopy.

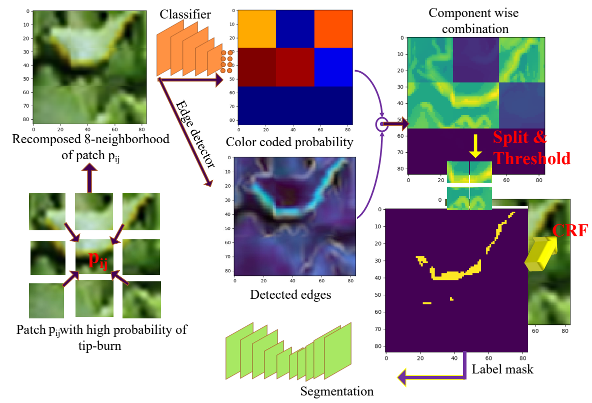
4.1. Patch-level localization and detection

For the first step we take at most $60K$ patches as input, in equilibrium between the stressed and non-stressed ones, where the stressed samples were generated as described in previous sections. Yet, we had to split this dataset into two sets the *darkSpots* and the *non-darkSpots*, since the dark spots, due to factors such as light, background, tables color and reflectance, affect the classifier reducing its accuracy, see the Results section for accurate explanations and Figure 7. To separate the dataset into two clusters we used *KMeans*. For clustering we have considered edge features extracted with the Sobel edge detector, color features and, finally, the feature vectors obtained by scattering the Haar wavelets [46]. For detection and localization we implement a two class-classifier architecture highly inspired from DarkNet-19, YOLOv2 backbone[32], and replicate it for the *stress VS darkSpots* and *stress VS non-darkSpots* classification, see Section 5 for more details.

4.2. Mask generation

As noted in Section 3.1 each image is tiled into about $41K$ patches of size 28×28 along the three channels. The classification network generates patch level annotations assigning to each patch a probability which is proportional to the number of pixels on which tip-burn appears. We want to obtain a segmentation mask for each patch classified as stressed, at pixel level.

Each patch has index i, j, specifying its position on the complete image of the canopy. Since tip-burn occurs along the borders of the leaf we use the Sobel edge detector already used for clusterization. Let $P(p_{ij})$ and $E(p_{ij})$ be resp. the probability map of the patch and the computed edges map, we obtain for patch p_{ij} the activation map $M_{ij} = P(p_{ij}) \odot E(p_{ij})$, with $\odot$ the component-wise product between matrices. A map M_{ij} of a patch p_{ij} can be tiled into n mini-patches denoted by p_{i_u,j_v}, with $u, v \in \{1, \ldots, n\}$ and $n \leq 28$. For example, if $u, v \in \{1, \ldots, 4\}$, we get a tiling of the map M_{ij} into 16 mini-patches (or superpixels) of size 7×7. Each mini-patch value is computed as follows:



Classifier
Component wise combination
Recomposed 8-neighborhood of patch p_{ij}
Color coded probability
Edge detector
Split & Threshold
CRF
p_{ij}
Detected edges
Patch p_{ij} with high probability of tip-burn
Label mask
Segmentation

Figure 8. Prediction: unsupervised tip-burn mask generation by merging probability outcomes from classification and detected edges, CRF and segmentation.

$$V_{i_u,j_v} = \frac{1}{K} \sum_{x,y \in p_{i_u,j_v}} wM_{ij}(x,y), \quad u,v \in \{1,\ldots,n\} \tag{2}$$

Here x, y indicate the pixel values of the mini-patch, w is a mini-patch specific weight reinforcing the contribution of edges, and $K{=}\max M_{ij}$. The *label map* of a patch p_{ij} tiled into n mini patches has *tile-level* n, and it is defined by a square $n \times n$ matrix thresholding the values defined in (2):

$$LM_n(p_{ij}) = \eta([V_{i_u,j_v}]^{n\times n}) \tag{3}$$

Here $\eta : [V_{i_u,j_v}]^{n\times n} \mapsto \{0,1\}^{n\times n}$ is a thresholding operation based on the max value of the matrix $[V_{i_u,j_v}]^{n\times n}$ quantization. LM_n labels each pixel (or superpixel depending on the tiling level chosen) according to whether it is most likely healthy or stressed (see Figure 8).

Given a label map LM_n of a patch, we combine label maps into a patch neighborhood system. The N-neighborhood of a patch is defined by the $N{=}8$ patches at distance δ_1, or by the $N{=}24$ patches at distance δ_2, and so on. Therefore, a label map *induced* by a patch has size $N{+}1$. In Figure 8 we show, within the tip-burn prediction process, the simple computation to obtain a refined label map for a N-neighborhood with $N{=}8$. Given a choice of N we refine the obtained masks by a random field (CRF), where the objective is to predict a label $x_p \in \{0, 1\}$ for each pixel (or superpixel) p. Forming unary and binary potentials and ignoring the partition function, the energy function of the CRF decomposes into nodes (the pixels p) and edges:

$$E(\mathbf{x}_p) = \sum_i \varphi_i(x_{p_i}) + \sum_{i,j \in LM_n} \varphi_{i,j}(x_{p_i}, x_{p_j}) \tag{4}$$

Here, the potential $\varphi_i(x_{p_i})$ encodes the probability that the pixel p_i denotes either stress or health, and it is encoded by a classifier, while $\varphi_{i,j}(x_{p_i}, x_{p_j})$ enforces the labels to be the same if the pixels are similar and it is encoded by a distance. Inference is done by MAP minimization of the energy function:

$$\arg\min_{\mathbf{x}_p} E(\mathbf{x_p}) \tag{5}$$

In our case the energy function is of class $\mathcal{F}^2$ [23] and it is graph-cut representable. In this case it is known that graph cut gives an optimal solution for the minimization [22], also removing noise from the training labels built as described above. These refined masks are then used for training the pixel-level segmentation described below.

4.3. Pixel-level segmentation

Tip-burn prediction amounts to obtain the image shown in Figure 2 where tip-burn regions occurring in the image of a canopy are segmented. Since the work of [34] many further progress have been made on U-shaped segmentation models based on convolution and deconvolution [29]. The idea behind a deep segmentation model is to have a network built by an encoder and decoder for labeling each pixel. The encoder contracts the spatial resolution of the image up to a single vector, which forms the bottleneck, learning more and more abstract features in this contraction process. Beyond the bottleneck, deconvolution layers restore the original image resolution by upsampling layers [20]. Since DenseNet [17] skip connections have signed the U-Net evolution enhancing models for recovering fine-grained details of the target, improving the flow between layers by connecting each layer to all subsequent ones.

In our case we have to cope with very small regions with strong shape variation, imposing a relatively shallow net. Input to the net is a binary image $[0, 1]^{196\times196}$ with $N = 48$ and the corresponding RGB image cropped from the image of full canopy, tiled into tray images, to which we applied only random jittering, as flipping is not necessary, due to the mask being deformable. The implementation is essentially a relatively shallow U-net, as described in Section 5. Maps whose sum is zero are not considered in either training and validation.

5. Experiments

Here, we explore more in-depth the key implementation details of the models used for our experiments. All our networks are implemented in Tensorflow 2.3 and training are performed on a RTX-GPU 2080 and on a GTX 1050Ti. Reported values and thresholds for training are established empirically within a standard range.

WGAN implementation for sample generation. The Generator and the Critic are deep CNN architectures. Regarding the training procedure, we apply to every samples from two to three random rotations to increase the original dataset dimension and use batch size of 62. The images are also normalized in the range $[-1, 1]$. The noise dimension for the Generator is set to 256. We use standard RMSprop with learning rate $1e{-}5$ for both the Generator and the Critic. The networks are trained for 1500 epochs with the Critic being updated 5 times more than the Generator during each step. The convergence of the WGAN depends on a clipping value c. Formally, c enforces a Lipschitz constraint that makes the computation of the Wasserstein distance to be tractable [3]. Practically, if set too low or too high, vanishing and exploding gradients respectively can occur. In our case, we empirically devised that setting c equal to 0.02 is a good compromise between network convergence and training time.

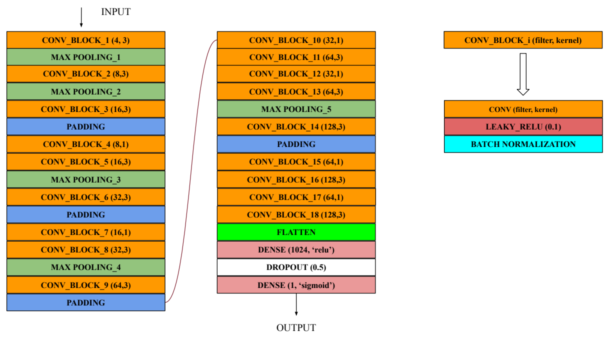


Figure 9. Patch-level classification network (*Stress vs darkSpot*).

VGG-16 for generated data metric. We use the VGG-16 pretrained model on the ImageNet dataset, provided by tensorflow.org. The base network implementation [40] works with inputs of size $(224\times224\times3)$. Tensorflow allows to create links between different model layers. We exploit this to make the base VGG-16 accept smaller inputs, up to a minimum of $(32\times32\times3)$, requiring us to rescale the images. We also convert them to the BGR format and zero-center each channel with respect to the ImageNet dataset. We freeze the base model and train only the head for 30 epochs, with batch size of 32 and Adam optimizer with learning rate scheduled on epochs, starting from $1e{-}3$. At last, we unfreeze the base model with a learning rate set to $1e{-}5$.

Patch-level classification networks. Both the two class-classifier architectures use their own DarkNet-19 network as backbone (Figure 9). The head of the network is replaced with two dense layers with relu and sigmoid functions respectively, a dropout of 0.5 in-between them, and L2 weight regularization in each convolutional block. The structural difference between the *Stress VS no-darkSpots* and the *Stress VS darkSpots* lies in the number of parameters used: the former has 4 times the number of parameters in each convolutional block than the latter. Both the architectures are trained from scratch, using a total of $60k$ images evenly divided between the two class categories. The dataset is split into training and validation sets with a ratio of $90/10$. We use batch size of 64 and normalize the

Table 1. Performance metric of the proposed CNN models for patch-level and pixel-level detection and segmentation tasks

Metric	Patch-level detection (*healthy class split*)		Pixel-level detection (*via patch classification only*)	Patch-level detection (*Ablation: no healthy class split*)		Pixel-level segmentation
	Validation	Test	Test	Validation	Test	Test
Accuracy	96.01%	87.05%	86.51%	74.65%	49.23%	-
Recall	96.52%	40.45%	65.12%	98.02%	98.02%	-
Precision	95.68%	8.34%	67.74%	69.04%	2.12%	-
F1	96.10%	13.83%	64.80%	81.01%	4.15%	-
mAP	-	-	-	-	-	87.00%
IOU	-	-	-	-	-	77.20%
Dice score	-	-	-	-	-	75.02%

images in the range $[0, 1]$. We use SGD with Nesterov Momentum and learning rate $1e{-}5$ to train the *Stress VS darkSpots* model for 38 epochs. The training process automatically stops if, after 5 consecutive epochs, the validation loss does not show meaningful improvements. The *Stress VS no-darkSpots* model is trained for 32 epochs. SGD with Nesterov Momentum is again used but the learning rate is now set smaller, to a value of $6e{-}6$. Batch size and the early stopping criteria are the same as the previous model.

Pixel-level segmentation network. The network follows a U-net structure with 4 convolution layers, both for the encoder and the decoder, with kernels of size 3 and leaky-relu as activation functions. Instance normalization and dropout, with a value of 0.3, are applied respectively before and after the max-pooling operation to reduce overfitting. We use batch size of 1 and Adam optimizer with learning rate set to $1e{-}4$. We train the model for 50 epochs with binary cross-entropy as loss function. As for the metrics, we use both the intersection over union and the dice coefficient. The intersection over union is computed as the ratio between the area of overlap among the ground truth and the predicted mask, and the union of ground truth and predicted mask, in pixels. The dice coefficient is defined as:

$$\frac{2 \times |Gt(mask)| \cap |Pred(mask)|}{|Gt(mask)| + |Pred(mask)|} \tag{6}$$

and it benchmarks the similarity between two samples.

6. Results

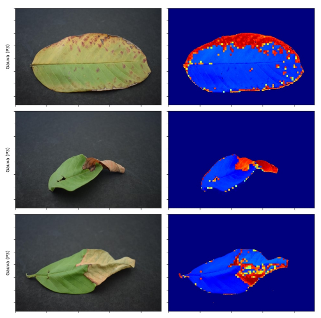
In Table 1, the *Patch-level detection* column shows the combined performances of the two class-classifiers on the validation and test set, respectively. Let τ be a threshold (in our case set to 0.37):

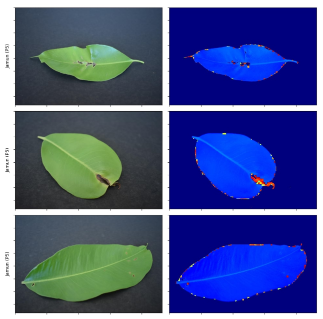
$$predicted(p_{ij}) = \begin{cases} \text{stress} & \text{if } P(p_{ij}) > \tau \\ \text{healthy} & otherwise \end{cases} \tag{7}$$

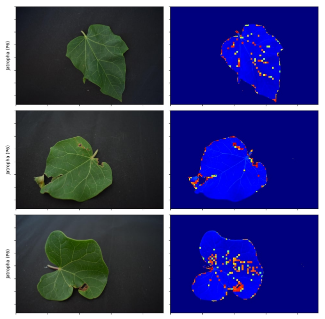
Here we consider TPs all the patches labeled as stressed and predicted as stress. FPs are all the samples labeled as healthy and predicted as stress, FNs all patches labeled as stress, and predicted as healthy. TNs all patches labeled and predicted as healthy. The huge drop in both precision and recall for the test set shows that patch-level detection, despite its flexibility, needs further refinement to remove FPs. In fact, the two patch-level models predict about $[1000 - 4500]$ more stress samples than there actually are. To check pixel-level accuracy, we collect a test set of 10 canopy images where tip-burn masks have been manually extracted by the agronomists. First we tested the pixel-level accuracy of the *Patch-level detection*, shown in Table 1 (column *Pixel level-detection*), by computing TP, FP, TN, FN via the intersection of patches and test masks, we see that precision and recall actually improve. Then we tested the tip-burn segmentation, obtained by unsupervised labeling, CRF and U-Net segmentation. Table 1 column *Pixel-level segmentation* shows the IOU and Dice score metrics for the test obtaining values higher than 75%.

6.1. Ablation study

An ablation study is motivated by the specific decision of splitting the healthy class into two subsets, taken during the experiments. We conduct the ablation study by training a third CNN classifier avoiding the *informative* sampling for the *healthy* class, hence no split is made between *darkSpot* or *no-darkSpot* type. The settings and hyper-parameters used for training are the same as the one presented for the other two networks. In such a scenario, the validation loss starts oscillating immediately after 7 epochs until the model completely overfits the training dataset in less than 30 epochs. Stopping the model before that, leads to the performance metrics shown in Table 1, column *Patch-level detection (Ablation:..)*. A drop for both the validation and test sets, most notably precision, is visible. Increasing network capacity, learning rate scheduling and loss and weights regularization were unsuccessful. We noticed that most of the misclassified samples were of the *darkSpot* type. Increasing the importance of these misclassified samples via cost sensitive learning proved unsatisfactory yet again. We strongly believe that this issue can be overcome even without splitting the samples in two categories. Simply increasing the number of samples given as input could do the trick, we reserve the search for a better solution in a future study.


(a) Guava [mAP: 93%]


(b) Jamun [mAP: 85%]


(c) Jatropha [mAP: 80%]

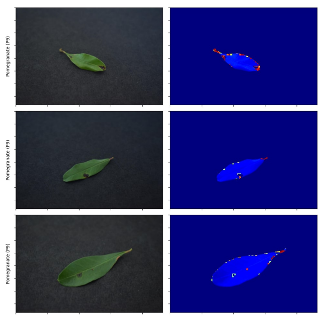

(d) Pomegranate [mAP: 79%]

Figure 10. Comparison results on PlantLeaves

7. Comparison on other datasets

Models performance on PlantVillage and PlantLeaves As already gathered in Section 2, as far as we know there is no work on canopy segmentation, nor on disease segmentation even on a single image. The closest methods is [21], which provides bounding box detection of stressed leaves on a custom apple leaf disease dataset obtaining a mAP of 78.80%. Similarly, [5] obtains mAP scores ranging from 91.8% to 92.7% in bounding box detection of a new PlantDisease[10] dataset not publicly available.

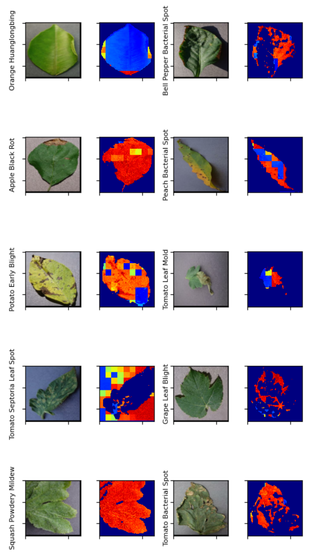
Given a lack of comparable works, to prove that our method is quite flexible and easy generalizable, we test it on the PlantVillage and PlantLeaves datasets, by randomly sampling images from both dataset, despite the networks were trained on our canopy dataset. Figure 10 shows the mAP accuracy of segmentation maps from samples of PlantVillage, while Figure 11 shows the accuracy of segmentation maps for PlantLeaves dataset. Our method achieves a mean Average Precision (mAP) of 67% and 85% respectively for PlantVillage and PlantLeaves showing very challenging results, especially considering the above results of [21] and [5] obtained by training on their own datasets.

8. Conclusions

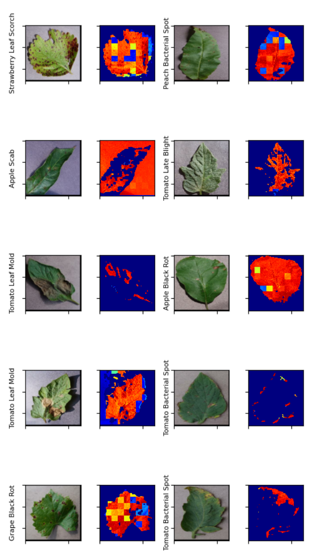
This study has explored, for the very first time, how DL can solve the problem of tip-burn detection on highly dense plant canopies in Plant Factories. We propose a WGAN to overcome the problem of dataset imbalance. The quality of the generated synthetic samples is confirmed by precision and recall metrics [25]. Two Deep CNNs estimate the probability of images containing tip-burn which refined by labeling maps and CRF, allow to extract masks for the most probable regions. At last, patch merging and pixel-level segmentation with a deep network allow to carry out a pixel-level segmentation of the whole canopy images achieving a mAP value of 87%. The quick fix provided to overcome the local minimum caused by the full *Healthy* dataset can definitely be improved and we leave for a future study the task of finding a better solution to it. In addition to this, the auspicious results presented in the ablation study motivate us to extend the developed system also on the other other plant varieties grown inside Plant Factories and their relevant stresses. Finally, we shall deliver the collected dataset to the entire research community so as to foster the search of better implementations to the same problem or finding new ones for other applications.

9. Acknowledgments

The work has been carried out under funds Sunspring Rep. c/third 1/2020 of Agricola Moderna. We thanks all the personnel at Agricola Moderna for helping us in collecting the images and in all the steps of this research.



(a) Test 1 [mAP: 69%]



(b) Test 2 [mAP: 65%]

Figure 11. Comparison results on PlantVillage.

References

[1] Mohit Agarwal, Abhishek Singh, Siddhartha Arjaria, Amit Sinha, and Suneet Gupta. ToLeD: Tomato leaf disease detection using convolution neural network. *Procedia Computer Science*, 167:293–301, 2020. International Conference on Computational Intelligence and Data Science. 2

[2] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017. 2

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017. 3, 6

[4] Marko Arsenovic, Mirjana Karanovic, Srdjan Sladojevic, Andras Anderla, and Darko Stefanovic. Solving current limitations of deep learning based approaches for plant disease detection. *Symmetry*, 11(7):939, 2019. 2

[5] Marko Arsenovic, Mirjana Karanovic, Srdjan Sladojevic, Andras Anderla, and Darko Stefanovic. Solving current limitations of deep learning based approaches for plant disease detection. *Symmetry*, 11(7), 2019. 8

[6] Jayme GA Barbedo. Factors influencing the use of deep learning for plant disease recognition. *Biosystems engineering*, 172:84–91, 2018. 2

[7] Jayme Garcia Arnal Barbedo. Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification. *Computers and electronics in agriculture*, 153:46–53, 2018. 2

[8] Jayme Garcia Arnal Barbedo. Detection of nutrition deficiencies in plants using proximal images and machine learning: A review. *Computers and Electronics in Agriculture*, 162:482–492, 2019. 2

[9] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018. 4

[10] Med Brahimi, Marko Arsenovic, Sohaib Laraba, Srdjan Sladojevic, Boukhalfa Kamel, and Abdelouahab Moussaoui. *Deep Learning for Plant Diseases: Detection and Saliency Map Visualisation*. 06 2018. 8

[11] Siddharth Chouhan, Ajay Koul, Dr. Uday Singh, and Sanjeev Jain. A data repository of leaf images: Practice towards plant conservation with plant pathology. 11 2019. 1, 2, 3

[12] EF Cox and JMT McKee. A comparison of tipburn susceptibility in lettuce under field and glasshouse conditions. *Journal of Horticultural Science*, 51(1):117–122, 1976. 2

[13] EF Cox, JMT McKee, and AS Dearman. The effect of growth rate on tipburn occurrence in lettuce. *Journal of Horticultural Science*, 51(3):297–309, 1976. 1

[14] Chad DeChant, Tyr Wiesner-Hanks, Siyuan Chen, Ethan Stewart, Jason Yosinski, Michael Gore, Rebecca Nelson, and Hod Lipson. Automated identification of northern leaf blight-infected maize plants from field imagery using deep learning. *Phytopathology*, 107:1426–1432, 06 2017. 2

[15] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. 3

[16] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 4

[17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 6

[18] Sheng-Wei Huang, Che-Tsung Lin, Shu-Ping Chen, Yen-Yi Wu, Po-Hao Hsu, and Shang-Hong Lai. Auggan: Cross domain adaptation with gan-based data augmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 718–731, 2018. 2

[19] David. P. Hughes and Marcel Salathe. An open access repository of images on plant health to enable the development of mobile disease diagnostics, 2016. 1, 2, 3

[20] Simon Jégou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 11–19, 2017. 6

[21] Peng Jiang, Yuehan Chen, Bin Liu, Dongjian He, and Chunquan Liang. Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks. *IEEE Access*, 7:59069–59080, 2019. 8

[22] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009. 6

[23] Vladimir Kolmogorov and Ramin Zabin. What energy functions can be minimized via graph cuts? *IEEE transactions on pattern analysis and machine intelligence*, 26(2):147–159, 2004. 6

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 2

[25] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *arXiv preprint arXiv:1904.06991*, 2019. 3, 4, 8

[26] Benjamin Franklin Lutman. *Tip burn of the potato and other plants*. Vermont Agricultural Experiment Station, 1919. 1, 2

[27] Giovanni Mariani, Florian Scheidegger, Roxana Istrate, Costas Bekas, and Cristiano Malossi. Bagan: Data augmentation with balancing gan. *arXiv preprint arXiv:1803.09655*, 2018. 2

[28] Koushik Nagasubramanian, Sarah Jones, Asheesh Singh, Arti Singh, Baskar Ganapathysubramanian, and Soumik Sarkar. Explaining hyperspectral imaging based plant disease identification: 3d cnn and saliency maps. 04 2018. 2

[29] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015. 6

[30] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans, 2017. 2, 3

[31] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016. 3

[32] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger, 2016. 5

[33] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018. 3

[34] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. (available on arXiv:1505.04597 [cs.CV]). 6

[35] Mehdi SM Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. *arXiv preprint arXiv:1806.00035*, 2018. 4

[36] Muhammad Hammad Saleem, Johan Potgieter, and Khalid Mahmood Arif. Plant disease classification: A comparative evaluation of convolutional neural networks and deep learning optimizers. *Plants*, 9(10), 2020. 2

[37] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29:2234–2242, 2016. 4

[38] Shigeharu Shimamura, Kenta Uehara, and Seiichi Koakutsu. Automatic identification of plant physiological disorders in plant factories with artificial light using convolutional neural networks. *International Journal of New Computer Architectures and Their Applications*, 9(1):25–31, 2019. 2

[39] Vimal Shrivastava, Monoj Pradhan, S. Minz, and M. Thakur. Rice plant disease classification using transfer learning of deep convolution neural network. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-3/W6:631–635, 07 2019. 2

[40] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. 6

[41] Davinder Singh, Naman Jain, Pranjali Jain, Pratik Kayal, Sudhakar Kumawat, and Nipun Batra. Plantdoc. *Proceedings of the 7th ACM IKDD CoDS and 25th COMAD*, Jan 2020. 1, 2

[42] Jung Eek Son and Tadashi Takakura. Effect of ec of nutrient solution and light condition on transpiration and tipburn injury of lettuce in a plant factory. *Journal of Agricultural Meteorology*, 44(4):253–258, 1989. 2

[43] Sungho Suh, Haebom Lee, Paul Lukowicz, and Yong Lee. Cegan: Classification enhancement generative adversarial networks for unraveling data imbalance problems. *Neural Networks*, 133:69–86, 01 2021. 2

[44] GP Termohlen and AP vd Hoeven. Tipburn symptoms in lettuce. In *Symposium on Vegetable Growing under Glass 4*, pages 105–110, 1965. 1, 2

[45] Ngoc-Trung Tran, Viet-Hung Tran, Ngoc-Bao Nguyen, Trung-Kien Nguyen, and Ngai-Man Cheung. On data augmentation for gan training. *IEEE Transactions on Image Processing*, 30:1882–1897, 2021. 2

[46] Michael Unser. Texture classification and segmentation using wavelet frames. *IEEE Transactions on image processing*, 4(11):1549–1560, 1995. 5

[47] Sakshi Arora Vippon Preet Kour. Plantaek: A leaf database of native plants of jammu and kashmir. Mendeley Data, 2019. 1

[48] Ukrit Watchareeruetai, Pavit Noinongyao, Chaiwat Wattanapaiboonsuk, Puriwat Khantiviriya, and Sutsawat Duangsrisai. Identification of plant nutrient deficiencies using convolutional neural networks. In *2018 International Electrical Engineering Congress (iEECON)*, pages 1–4. IEEE, 2018. 2

[49] Xin Zhang, Liangxiu Han, Yingying Dong, Yue Shi, Wenjiang Huang, Lianghao HAN, Pablo González-Moreno, Huiqin Ma, Huichun Ye, and Tam Sobeih. A deep learning-based approach for automated yellow rust disease detection from high-resolution hyperspectral uav images. *Remote Sensing*, 11:1554, 06 2019. 2