Computers and Electronics in Agriculture 150 (2018) 196–204

Contents lists available at ScienceDirect

Computers and Electronics in Agriculture

journal homepage: www.elsevier.com/locate/compag

Original papers

Seed-per-pod estimation for plant breeding using deep learning

L.C. Uzal[a,*], G.L. Grinblat[a], R. Namías[a], M.G. Larese[a], J.S. Bianchi[b], E.N. Morandi[b], P.M. Granitto[a]

[a] CIFASIS, French Argentine International Center for Information and Systems Sciences, CONICET - UNR, Argentina
[b] Laboratorio de Fisiología Vegetal, Facultad de Ciencias Agrarias, Universidad Nacional de Rosario & Instituto de Investigaciones en Ciencias Agrarias de Rosario (IICAR), CONICET - UNR, Argentina

ARTICLE INFO

Keywords:
Deep learning
Machine vision
Plant breeding programs
Plant phenotyping
Soybean

ABSTRACT

Commercial and scientific plant breeding programs require the phenotyping of large populations. Phenotyping is typically a manual task (costly, time-consuming and sometimes arbitrary). The use of computer vision techniques is a potential solution to some of these specific tasks. In the last years, Deep Learning, and in particular Convolutional Neural Networks (CNNs), have shown a number of advantages over traditional methods in the area. In this work we introduce a computer vision method that estimates the number of seeds into soybean pods, a difficult task that usually requires the intervention of human experts. To this end we developed a classic approach, based on tailored features extraction (FE) followed by a Support Vector Machines (SVM) classification model, and also the referred CNNs. We show how standard CNNs can be easily configured and how a simple method can be used to visualize the key features learned by the model in order to infer the correct class. We processed different seasons batches with both methods obtaining 50.4% (FE + SVM) and 86.2% (CNN) of accuracy in test, highlighting the particularly high increase in generalization capabilities of a deep learning approach over a classic machine vision approach in this task. Dataset and code are publicly available.

1. Introduction

Plant phenotyping can be defined as the identification and quantification of effects on the phenotype (i.e., the appearance and behavior of plants), using appropriate protocols and measurements, as result of both genotype differences and the interaction with the environment (Fiorani and Schurr, 2013). In their search for increased yields, plant breeding programs require the phenotyping of large populations, evaluating some or even several useful traits (Ghanem et al., 2015). In addition, the results must be validated through multiple environments and replicated trials, increasing the burden of the process. Unfortunately, phenotyping is typically a manual task, therefore laborious, costly, and time-consuming. Even more, visual evaluation over many segregating plants in the field is very difficult and error-prone due to the observer subjectivity. As a consequence, phenotyping has become a bottleneck for plant breeding programs (Singh et al., 2016). Thus, the possibility of developing quick, accurate and repeatable methods to characterize individual plant phenotypes will increase the quality of the selection process and will provide a useful tool to foster the incorporation of desirable traits into commercial germplasm.

In recent years, imaging-based automatic methods have been introduced to plant phenotyping (Fahlgren et al., 2015; Scharr et al., 2016), using diverse devices and multiple scales. For example, Bendig et al. (2014) used aerial images to estimate the biomass of entire crops. Giuffrida et al. (2015) developed an efficient method for counting leaves in rosette plants with images of individual plants in a controlled environment in the context of the Leaf Counting Challenge held in the CVPPP 2015 workshop. This challenge led to ongoing research which is using deep learning techniques, (Aich and Stavness, 2017; Dobrescu et al., 2017) for instance. Also (Pound et al., 2017) used deep learning for localising wheat spikes and spikelets.

In particular, the yield of a soybean crop depends on three major components: the number of pods per plant (PN), the number of seeds per pod (SPP) and the seed size (Fehr, 1987). Under cultivation conditions, yield is subject to strong genotype-environment interaction. However, of the three main components, SPP is the least subject to environmental influence (i.e. it is a characteristic of the genotype (Board and Harville, 1998)) thus offering the opportunity to genetically manipulate it to improve the yield potential of a cultivar.

The main goal of any breeding program is to develop varieties with high yield potential. The capability of selecting for traits linked to yield components such as high PN and SPP, early during the selection process, would increase the efficiency of the breeding program. This implies to cope with large genetic-engineering experiments consisting in the order of tens of thousands plants to be manually labeled pod by pod, among the other previous mentioned features, to select the best

* Corresponding author.
E-mail address: uzal@cifasis-conicet.gov.ar (L.C. Uzal).

https://doi.org/10.1016/j.compag.2018.04.024
Received 21 December 2017; Received in revised form 6 April 2018; Accepted 21 April 2018
Available online 01 May 2018

phenotype.

Counting the number of pods per plant is a simple but tedious task that can be easily automated. Furthermore, estimating the number of seeds per pod is laborious and difficult as well, requiring visual inspection of each pod by a human expert. The difficulty of the task relies on the wide range of maturing sizes of seeds within the pods as a result of environmental or genetic factors. All seeds must be counted even if they have suffered an abortion in the early stages of pod development. In such limit cases a trained expert can still visually distinguish subtle changes in pod shape evidencing the presence of an aborted seed. As a measure of the difficulty of the task, we point out that a group of trained operators achieve an accuracy of 84% (with an standard deviation of 2%).[1]

In this work we introduce an automated tool that could replace human experts in this counting task, allowing the increase in scale of breeding programs without losing accuracy and, in consequence, the speed up of the complete process. There are some precedents for automatic object counting from digital images in the context of agricultural applications (Dorj et al., 2017; Harmsen and Koenderink, 2009; Liu et al., 2017; Maldonado and Barbosa, 2016; Mussadiq et al., 2015; Aich and Stavness, 2017; Dobrescu et al., 2017; Pound et al., 2017). However, our case is different as we need to infer the number of seeds –that are hidden– from the pod shape.

Convolutional Neural Networks (CNNs) have proven to be very effective at solving vision problems in a wide range of fields and they have been one of key elements of the success of Deep Learning (LeCun et al., 2015). In the context of agricultural applications, recent years have witnessed a growing tendency to replace classic techniques with deep learning algorithms for a variety of vision tasks (Grinblat et al., 2016; Ding and Taylor, 2016; Sladojevic et al., 2016; Lu et al., 2017; Tang et al., 2017). Even a typical phenotyping problem, counting leaves, has been tackled with these techniques (Ubbens and Stavness, 2017). Recently, a complete survey of Deep Learning in agriculture has been published (Kamilaris and Prenafeta-Boldú, 2018). According to this survey our application is new and will fall into the area of plant phenology recognition where only two paper were surveyed (Yalcin, 2017; Namin et al., 2017).

The advantages of the Deep Learning approach for vision tasks are twofold: there is no need to carefully design handcrafted features extractors for the problem at hand, as CNNs can learn specialized features extractors from raw data, and when provided with enough data, CNNs generally reach higher accuracies than classic techniques (LeCun et al., 2015). An important drawback of neural networks is the high number of hyperparameters associated to the model architecture design (such as number of layer, units per layer, etc) and the training algorithm (learning rate, momentum) as well. Finding appropriate values may feel like a challenging task to an inexperienced user. However, current deep learning libraries offer default values for hyperparameters which are good starting points for optimal search.

The aim of this paper is to tackle the problem of counting the SPP number for soybean pods with a Deep Learning approach based on standard Convolutional Neural Networks and to compare the results with a classic approach based on a set of tailored features extracted specifically for this problem and a SVM classifier. We also offer an extensive study of how hyperparameters selection impacts on the CNN's performance on this task. This paper also aims to contribute to collecting evidence in favor of the widespread use of Deep Learning for agricultural applications even for users not specialized in these tools.

[1] In order to further illustrate the difficulty of the classification task, we have intentionally included in Fig. 2 some hard samples together with more common cases.

2. Background

2.1. Classic approach

The classic procedure in machine vision is to define and extract appropriate features for the problem at hand and then train a classifier (e.g. an SVM with Gaussian kernel) in the corresponding representation space (as done for example in (Wu et al., 2007; Pydipati et al., 2006; Golzarian and Frick, 2011)). Part of our team firstly tackled this problem with such a classic strategy.

In a preliminary stage, three state-of-the-art classic classification methods were implemented, namely SVM, Random Forest (RF) (Breiman, 2001) and Penalized Discriminant Analysis (PDA) (Hastie et al., 1995). The cross-validation error for the three methods were found to be comparable, with a slight difference in favor of SVM. For this reason, in this paper SVM was chosen as the classic classification method for comparison purposes against CNN.

Regarding the handcrafted features, many geometrical (Umbaugh, 2005) and shape (Hu, 1962) features were considered at first. Lately, the addition of a 25-bin histogram of the profile of the pod shown to improve classification. Alternatively, an ad hoc method of ellipse fitting of beans inside the pod was designed, but it did not show any improvement.

2.2. Convolutional neural networks

CNNs, introduced by LeCun et al. (1990), have an architecture specially designed to process images. Its topology leads to a huge reduction in the number of free trainable parameters in comparison to a standard (fully connected) artificial neural network. This is due to its sparse neural connectivity (restricted to small receptive fields) and to the sharing of filter values along image locations exploiting translational invariance. In the following we briefly describe CNN's architecture mainly with the purpose of defining the hyperparameters considered in this work. For a more detailed description of this kind of models, we refer the reader to (LeCun et al., 2010) and references therein.

Fig. 4 depicts a diagram of the considered CNN model, inspired on the VGG architecture (Simonyan and Zisserman, 2014). Each layer is composed of three transforms. First, there is a convolution operation between the input image and a filter bank. Each filter has a bounded size associated to a small receptive field in the input image, typically 3×3 or 5×5 filter sizes are considered. For each filter in the bank, the convolution produces a feature map. Together with the convolution operation a subsampling step may be introduced. This replaces the standard average or max pooling operation by simply setting an stride larger than 1 in the convolution transform (Springenberg et al., 2014). A subsampling of factor 2 in each dimension of the feature map retains 1/4 of the output values (those with even indexes). This subsampling is usually accompanied with a duplication of the number of feature maps with respect to previous convolutional layers.

Second, a Batch Normalization (BN) transform (Ioffe and Christian, 2015) is applied after each convolution. This transform standardizes convolution output by fixing maps mean and deviation to 0 and 1 respectively over small batches of samples. It then applies a learnable gain and bias to each feature map. This layer has a beneficial regularization effect that has been verified in numerous machine vision applications (Vinyals et al., 2015; Radford et al., 2015; He et al., 2016), making it a standard tool.

Finally, the third transform is an element-wise nonlinear function applied to all feature maps. We use in all cases the Leaky ReLU function[2] (Maas et al., 2013), which is widely used to enhance the backpropagation signal.

[2] $LReLU(x) = \max(0.01x, x)$.

The last layer in the network is a *softmax function*. It returns the estimated probability of each class, given a concrete sample. This layer is fully connected to all output feature maps of the last convolutional layer. Alternatively, a *pooling* operation can be applied before this last layer to reduce dimensionality and, hence, the number of trainable parameters. We considered a global *max-pooling* operation that keeps just one output per feature map (the maximum).

A final relevant comment on CNNs is about visualization techniques. The procedures developed in (Zeiler and Fergus, 2014) allow the visualization of the patterns that are detected at each layer of the deep network, contributing to a partial reduction of CNNs from the category of black-box models. In this paper we use a simple procedure considered in (Zeiler and Fergus, 2014) for highlighting the most relevant input image regions for the network output probabilities (see Fig. 7).

3. Materials and methods

3.1. Data collection

In this study we considered populations of soybean plants coming from crosses among parents with diverse SPPs. We also considered populations from two consecutive seasons, in order to account for typical interseasonal variability.

The acquisition protocol starts collecting all the pods of a single plant with full maturity (R8 stage (Fehr and Caviness, 1977)). The pods are placed over a lightbox taking care to separate them from each other to facilitate posterior digital segmentation. Camera setups and position are kept fixed during each photography session where pod pictures of several plants are taken. Before each session, a picture of an object of known size is taken to calibrate the image scale for all the session.[3] For each plant three photographs were taken (see Fig. 1), one for each class (2-, 3-, and 4-SPP) where pods of the corresponding class are placed on top of the lightbox for the capture. For this task, a trained expert visually inspect each pod and classifies them by SPP class based on features like number of protuberances, shape of pod contour, etc.

3.2. Pods segmentation

We used OpenCV library (Bradski, 2000) for pods segmentation. Since segmentation is a simple task for these lightbox images (Fig. 1) almost any strategy allows for optimal results and therefore we only give here a minimal guideline. White background can be easily identified by thresholding the histogram of the full grayscale image. Borders of the image were removed by considering objects inside the white background external contour. We search for *connected components* and discard small objects. Then, a *watershed* procedure is applied to segment mutually touching objects. At this point we have a binary mask for each pod in the image. We apply the mask to isolate each pod image and then we compute the *bounding rectangle with minimum area* (best fitting rotated rectangle). Finally, we crop the rectangle region and align all images along the longest side. Fig. 2 shows sample images of segmented pods for each class. Table 1 shows how many pods were obtained after this segmentation process for each class and season.

3.3. Preprocessing

We considered a few simple image transformations before feeding the CNN. The processing protocol was designed bearing in mind the preservation of the relevant features for SPP visual estimation.

First we transform images to grayscale. Using color features would require increasing acquisition specifications such as white balance, illumination, exposure time, etc. and also the lapse that the plant is stored before being photographed (since this also alters the intensity of green). With these extra specifications the whole classification system becomes less robust. On the other hand, we know that the problem at hand can be solved by looking only at shape features.

We also rescale each pod image (keeping the aspect ratio) to fit 96×32 resolution. We span each image histogram to full range (0–255). Finally we invert intensity values in order to set background as 0. Fig. 3 shows the resulting images after preprocessing for the same set of Fig. 2. The full dataset of processed images can be downloaded from http://www.cifasis-conicet.gov.ar/uzal/dataset/soybean_pods.tar.gz.

3.4. Data augmentation

It is known that artificially increasing the number of training samples by applying simple random transformations to input images tends to improve CNNs performance[4] (Chatfield et al., 2014). Examples of such transformations –which should not alter the class label of the image– are flipping, shifting, scaling, rotating, shearing, among others (Keras Documentation – Image Data Generator). We also considered a random remapping of the image tonality by setting random *curves* (Gimp Documentation – Curves Tool) for pixelwise intensity transformation. Except for flipping, the amplitudes of all these transformations are continuous variables that can be regulated. For example, a random rotation can be bounded to a maximum angle ranging from 0 (no rotation based augmentation) to 180 degrees (free random rotations). We consider these amplitudes as hyperparameters of the training procedure and include them in the analysis of Section 3.8.

3.5. Feature extraction details

For the classic approach, a total of 38 tailored features were extracted from each automated segmentation of the pod. These features include essential geometrical characteristics (Umbaugh, 2005) (area, perimeter, major and minor axis length), shape features (Umbaugh, 2005) (density, elongation, compactness, rugosity and axis ratio), first 4 Hu moments (Hu, 1962), and finally a 25 bins histogram of the profile of the pod straighten mask added along the short axis. Segmented pods were binarized before feature extraction. All features were standardized before learning. The feature extraction process was done using standard OpenCV methods (Bradski, 2000).

3.6. SVM implementation details

We consider a support vector classifier with a Gaussian radial basis function (rbf) as kernel. This model has two hyperparameters: (i) the penalty parameter `C` of the error term in the SVM formulation and (ii) the shape parameter `gamma` of the rbf kernel.[5] We consider an exhaustive grid search for both hyperparameters with grid values [0.5, 1, 3, 5, 10, 50, 100, 200, 1000] for `C` and [50, 20, 14, 10, 5, 2, 1, 0.5, 0.2, 0.1, 0.05] for `gamma`. For the SVM implementation we use the Scikit-learn machine learning library (Pedregosa et al., 2011).

3.7. CNN implementation details

We explored different model architectures with the main structure as depicted in Fig. 4. We consider blocks of layers with a fixed number of convolutional layers. In the last convolutional layer of the block a stride factor of 2 in each spatial dimension reduces next block input feature map size by 4. Following the standard practice, together with this stride, we duplicate the number of feature maps. We call `block-Size` to the number of layers in each block and `nBlocks` to the number

[3] The real scale of the pod images is used only for the classic feature extraction approach. For the deep learning approach we discard scale information.

[4] The kind of transformations considered are ineffective for classic features, as they are based on binary masks and scale- and rotation-independent measurements.

[5] $K(u,v) = \exp(-\gamma|u-v|^2)$.

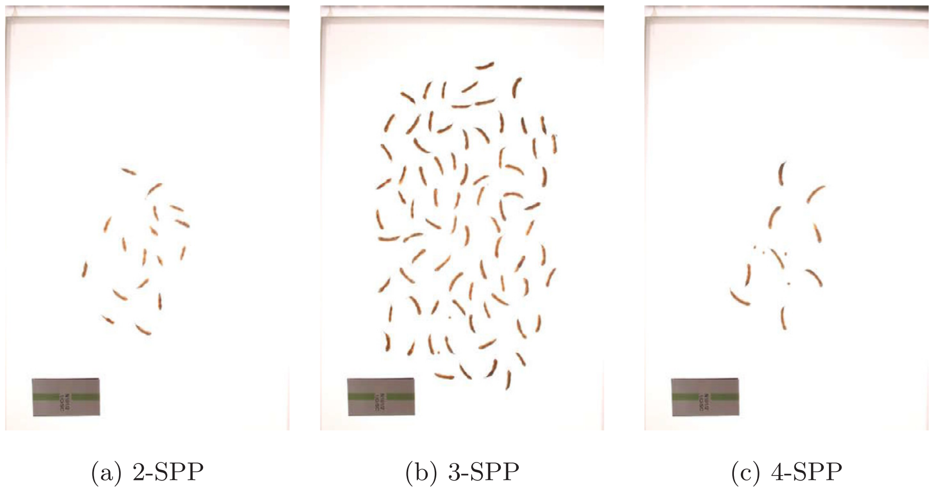

Fig. 1. Sample photographs of soybean pods used to build the dataset. Each pod is manually classified by an expert and photographed within its class group defined by the SPP number.

Table 1
Total number of examples corresponding to each class and season.

Class	Season 1	Season 2
2-SPP	811	3746
3-SPP	4598	5075
4-SPP	2444	1504
Total	7853	10325

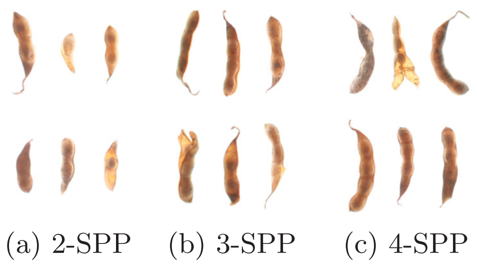

Fig. 2. Pod images obtained after segmentation process. Each panel corresponds to one of the three class label to be recognized, defined by the number of seeds per pod (SPP). Images shown preserve the original relative sizes.

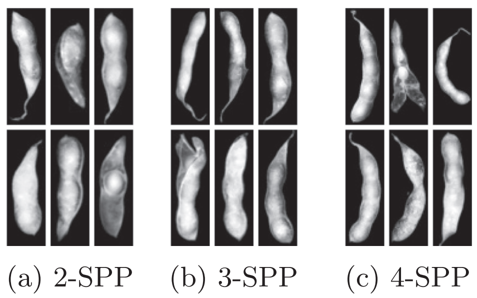

Fig. 3. Images obtained after preprocessing step. Samples are the same of Fig. 2.

of blocks in the network. We have therefore two hyperparameters for controlling model depth: `blockSize` and `nBlocks`.[6]

A global spatial pooling is applied to the last convolutional layer output. Finally, the last layer is a fully- connected layer (affine transformation) with a softmax nonlinearity that provides a probability output for each class.

Fig. 4 shows, as an example, the model corresponding to `blockSize` = 4 and `nBlocks` = 3, totalling 12 convolutional layers. The number of maps (`nMaps` in the figure) is `widthFactor` in the first 4 layers, `2*widthFactor` in layers 5 to 8 and `4*widthFactor` in the last 4 layers.

The models were trained using the Adam optimization algorithm (Kingma and Adam, 2014). This broadly adopted extension of the classic stochastic gradient descent algorithm computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradient. We set default parameters for this method except for the learning rate parameter for which we test random uniform values in a logarithmic scale around default value (see Table 2).

All the models were implemented[7] using Theano (Theano Development Team, 2016) and Lasagne (Dieleman et al., 2015), two libraries for Python which enormously simplify the training and usage of neural nets, and particularly convolutional neural nets. Regarding hardware, we used computers equipped with NVidia GeForce GTX 970 GPUs. With this setup, the considered 6000 iterations of training, for an average model size, take about 15 min.

3.8. Hyperparameter search

Table 2 shows the hyperparameters explored both for SVM and CNN. With the latter, we also test varying amplitudes for different data augmentation strategies. The size of the hyperparameter set for the CNN training makes the standard grid search infeasible. Instead, we consider a random search of hyperparameters (Bergstra and Bengio, 2012) which simply consists in progressively testing random combinations of hyperparameters values. This strategy is suitable when we expect that not all the hyperparameters have significant impact on model performance. In Section 4, we show that this hypothesis was confirmed.

3.9. Group k-fold cross-validation

To evaluate the classification performance for both considered approaches and for each hyperparameter combination we follow a standard *group k-fold cross-validation* procedure. This variation of *k*-fold ensures that the same group is not represented in both validation and training sets. For our dataset, the groups are defined by the photo

[6] `depth = blockSize * nBlocks + 1`.

[7] The code is available at https://github.com/CIFASIS/spp_estimation.

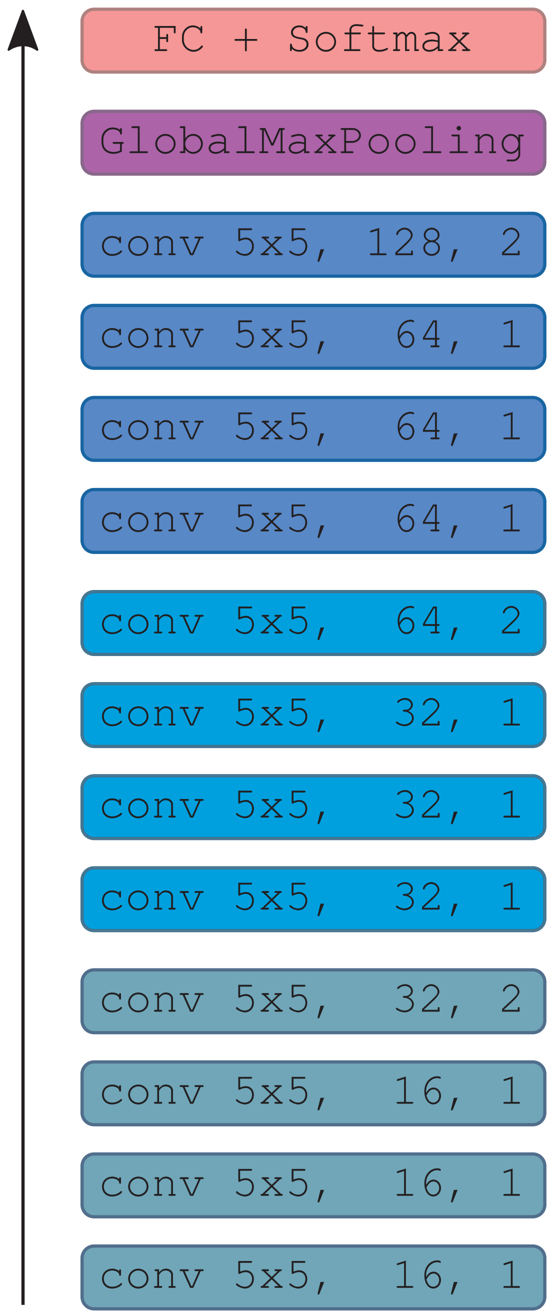

Fig. 4. Architecture of the Convolutional Neural Network selected in this work. There is a total of 12 convolutional layers plus an output softmax layer defining a depth of 13 layers. Each convolutional layer is labeled with its filter size, the number of filters, and the stride, respectively. This model has 3 blocks of 4 convolutional layers each. Every block ends with a convolutional layer with a stride of 2 in each spatial dimension together with a duplication of the number of filters. A global spatial pooling is applied before the fully- connected softmax layer. This architecture results from a search over networks of different numbers of blocks, block sizes and initial numbers of filters.

session detailed in Section 3.1. This grouping avoids possible correlations between pods of the same plant or unexpected photo capture artifacts (such as illumination or focus shift) which may bias performance estimation. As we just possess a small number of sessions, we set one fold per session.

3.10. Experimental setup

We first perform a hyperparameter search for both SVM and CNN. Hyperparameter combinations were chosen randomly (see Section 3.8). We consider as performance measure the classification accuracy (fraction of samples correctly classified). Each selected hyperparameter combination was evaluated over Season 1 data by group *k*-fold cross-validation (see Section 3.9). We report as validation accuracy the mean accuracy (and standard deviation) over the *k* folds of this season. Based on this search we chose a single set of hyperparameters for the CNN and the SVM.

The second stage is to evaluate and compare the CNN and SVM models using only the selected hyperparameters. To this end we consider Season 2 data as test set. Reported test accuracy mean and standard deviation were computed over test session groups and averaged over *k*-fold models.

4. Results and discussion

Fig. 5 summarizes the CNN results on validation accuracy for a random search in the space of hyperparameters. Accuracies shown are averages over group *k*-fold cross-validation sets from Season 1 data. The main conclusion derived from these results is that it is not necessary to make a careful selection of hyperparameters, but simply to consider a sufficiently deep network. This statement follows from observing that the accuracy does not correlate with training and data augmentation hyperparameters and yet it has a strong dependence on the network depth. It can also be observed that a very small `batchSize` limits accuracy. We believe that this behavior is rooted in the BN layers which perform statistics over batch samples. Very low `batchSize` may cause an unstable behavior of these layers. Another second-order effect is observed for the learning rate: very low learning rates (below 10^{-4}) amplify accuracies dispersion below optimal values. Given that we are training with a fixed number of iterations, low learning rates may simply not reach optimal accuracies and require more training time.

From the above exposed, we have to select the best combination of hyperparameters as candidate CNN model for further analysis. Given that a huge number of setups have near optimal performance (34% of random trials reach accuracies above 0.94) we can bias our choice to low size models without a significant loss of performance. In particular, the selected setup (pointed with a red arrow in Fig. 5) ranked second in terms of accuracy and it has two orders of magnitude less trainable parameters than setups of equivalent performance. Choosing the smallest model is justified by practical considerations such as reducing training and testing time.

When we evaluate the selected CNN model over test data (Season 2) we obtain a mean accuracy of 0.862 ± 0.052. This is just slightly above the accuracy reached by trained operators (0.84 ± 0.02). This may suggest that the CNN have the same difficulties as humans in classifying some hard samples. On the other hand we have not detected mislabelling on the dataset (but, as in any large dataset, we cannot discard the existence of mislabeled samples) that could explain part of the misclassified samples.

The next step was to compare the selected CNN model with the classifier based on the classic approach (feature extraction plus an SVM). Table 3 shows this comparison; it presents the results obtained with an SVM and with a CNN (with and without data augmentation). Regarding validation accuracy (based on group *k*-fold over the Season 1 dataset) we can observe that the difference between the methods is small nonetheless consistent. However, if the obtained models are applied to a test sample corresponding to Season 2 (not used in the training process) the differences become very important. While the classifier based on feature extraction + SVM strongly deteriorates, the CNN-based ones maintain an acceptable generalization capacity. In addition, data augmentation improves the test accuracy over the next season. This implies that the variations in shape, size and illumination performed by this process may recreate part of the diversity present in the test season but not in the training one. The low generalization of SVM models could be related to changes in the mean size of a pod from

Table 2
Explored hyperparameters and selected values (see Section 4) for SVM, CNN and Data Augmentation.

Method	Hyperparameter	Range	Selected Valu	Description
SVM	`C`	[0.5, 1, 3, 5, 10, 50, 100, 200, 1000]	10	SVM C parameter
	`gamma`	[50, 20, 14, 10, 5, 2, 1, 0.5, 0.2, 0.1, 0.05]	10	Gaussian kernel gamma
CNN	`blockSize`	[1–4]	4	Layers per block
	`nBlocks`	[1–5]	3	Number of blocks
	`widthFactor`	[8, 16, 32, 48, 64, 96, 128]	16	Multiplicative factor for the number of maps
	`log10lr`	[−5.0 to −2.0]	−2.43	Learning rate (log10 scale)
	`log10wd`	[−5.0–0.0]	−1.10	Weight decay (log10 scale)
	`batchSize`	[16, 32, 48, 64, 96, 128]	128	Samples per minibatch
Data Augm.	`zoom`	[0.9–1.1]	0.97	Zoom range center
	`zoomRange`	[0.0–0.25]	0.18	Amplitude of zoom interval
	`shear`	[0.0–0.35]	0.14	Maximum shear angle (radians)
	`wShift`	[0.0–10.0]	2.8	Maximum horizontal shift (%)
	`hShift`	[0.0–10.0]	1.6	Maximum vertical shift (%)
	`curves`	[0.0–0.75]	0.58	Maximum curve strength
	`rot`	[0–20]	20	Maximum random rotation (degrees)

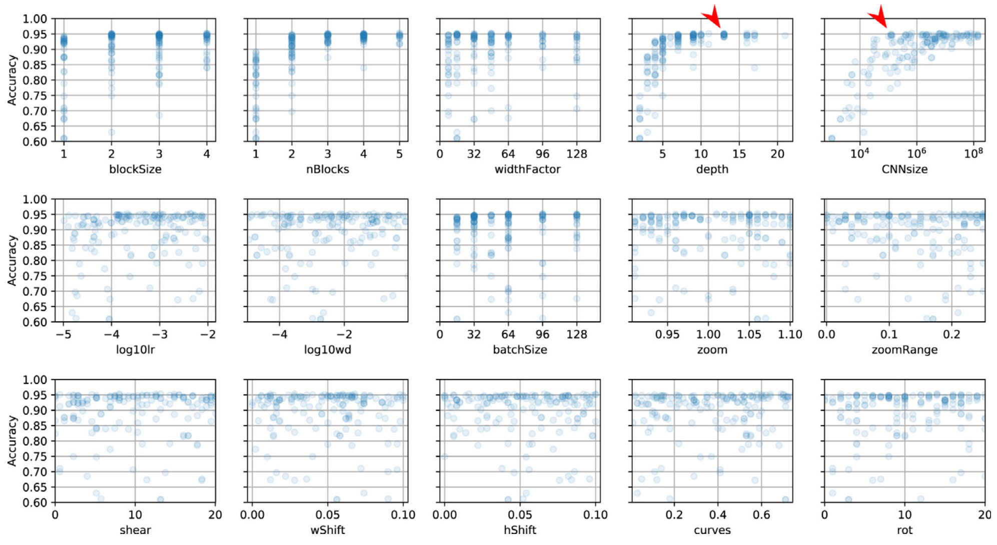

Fig. 5. Random search of hyperparameters for CNN training. The first row corresponds to model hyperparameters (defining network architecture). Variables `depth` and `CNNsize` are significative quantities derived from hyperparameters `blockSize`, `nBlocks`, and `widthFactor`. The rest of the panels corresponds to training algorithm and data augmentation hyperparameters (see Table 2 for details). Validation accuracy is almost insensitive to these training and data augmentation parameters. In order to reach high accuracies (above 90%), the only thing needed is to take a deep enough, high capacity network. Red arrows show the model selected which is a tradeoff between maximizing accuracy and minimizing model size.

Table 3
Accuracy for different methods trained on Season 1 data and tested on Season 2 data. Mean and deviation for validation accuracies were computed with a group *k*-fold procedure over training data. Test accuracy mean and standard deviation were computed over test session groups and averaged over *k*-fold models.

Method	Valid. Accuracy	Test Accuracy
Features + SVM	0.902 ± 0.022	0.504 ± 0.145
CNN without Data Augmentation	0.936 ± 0.009	0.827 ± 0.043
CNN with Data Augmentation	0.951 ± 0.005	0.862 ± 0.052

one season to another. We checked (data not shown) that deleting the four scale-dependent features (e.g area) does not improve the inter seasonal performance of the SVM models, suggesting that size is not the exclusive difference among seasons.

As to further characterize discrepancies among seasons we considered the inverse problem of training with Season 2 and testing on Season 1. For this experiment, we use fixed hyperparameters, those obtained by the random search. Fig. 6 compares training curves for both cases. The green curve shows the accuracy computed over a random minibatch during training. Every 100 iterations the accuracy is computed on the validation fold (blue line). When this validation accuracy reaches a new maximum (blue empty circles), the model is saved and the test accuracy is computed (red dots). The best model is chosen (i.e. early stopping of training) based only on validation performance and test accuracy is computed for visualization purposes only.

Comparing the two panels in Fig. 6, it can be seen that although the model trained on Season 1 does not completely generalize on Season 2, the behavior is not symmetrical. When training on Season 2, the accuracies tested on Season 1 are systematically better than those of validation. This confirms that the low generalization between these two seasons is not simply due to a bias in the pod features (e. g. a shift in the mean size) but that Season 2 dataset appears to contain more diverse and difficult examples which are not present in Season 1. On the other hand, Season 1 dataset characteristics seem to be totally included in Season 2 data. The difference in performance between classic and deep learning methods highlights the generalization capability of CNNs, based on their ability to learn appropriate and simple high level features from the data.

Finally, we considered training over joined Seasons 1 and 2.[8] To this end we concatenated datasets and considered nested group *k*-fold cross-validation loops. The outer loop separates each fold for testing, while the inner loop considered the remaining (k−1) folds and separates one for validation (for CNN early stopping) and the rest for training. This concatenated dataset has 11 sessions and therefore we considered

[8] Also for this experiment, we use fixed hyperparameters, those obtained by the random search.

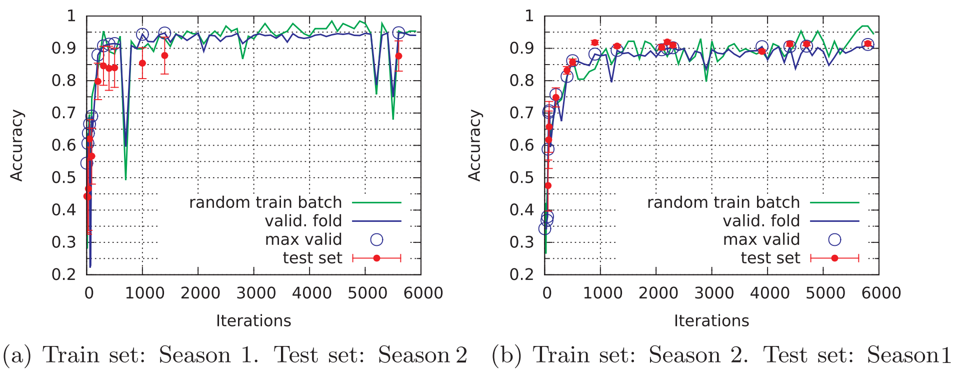

Fig. 6. Training curves for the two different seasons. Green lines: accuracy computed over a random minibatch during training. Blue lines: accuracy computed on the validation fold. Blue empty circles: new maximum in validation accuracy. Red dots: test accuracy. Season 2 dataset appears to contain more diverse and difficult examples which do not exist in the Season 1 version. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

$k = 11$ folds for the outer loop and 10 folds for the inner loop. This means $11 \times 10 = 110$ training runs over which we compute mean and standard deviation of model accuracies on test sets. The obtained value is 0.92 ± 0.05. This value is our best proxy of expected accuracy over new unseen data. It does not take into account possible new changes in data distribution as seen from Season 1 to Season 2.

4.1. Visualizing relevant patterns

We also performed an analysis to visualize which elements of an image are the most relevant for the CNN model to reach the final classification. We intend to show which regions of a pod image were more relevant to the CNN as to determine its output. We follow a simple but very effective technique based on evaluating the image with partial occlusions (Zeiler and Fergus, 2014).

In this technique, the image is occluded with a sliding block, in our case a 5x5 pixels black patch. Each occluded image –i. e., with the block occluding each different part of the original image– is classified with the trained model, and the output is compared with the output corresponding to the original image. The difference between these two outputs will be large when the sliding block occludes an important pattern for classification and small when the block occludes an irrelevant pattern. In this way we are able to visualize which regions of a given image are relevant for the model to perform its classification. It is worth mentioning that this method does not show all the patterns detected by the model, but only the most relevant ones for a given image and the relevance of each pattern can differ in each case.

Results are shown in Fig. 7 for representative samples of the confusion matrix elements. The heatmap indicates the locations of the image with more influence on the output. Green colored regions correspond to a decrease in the output probability of the correct class when those regions were occluded. Therefore green means a positive correlation with the correct class probability. On the other hand, red indicates an increase of correct class probability when the region is occluded implying a negative correlation.

In the main diagonal of the confusion matrix (Fig. 7), shown examples were correctly classified and therefore this occlusion procedure typically lowers the probability of the correct class, biasing the heatmap result to green as observed. More remarkably, the model seems to be detecting each seed individually in order to make the classification. This can be concluded from the green spots that are properly located over the seeds. It is worth mentioning that this phenomena does not imply that it is straightforward to build an individual seed detector from our trained CNNs (as in (Pound et al., 2017)), but that the network is using this pattern for class estimation in this sample cases.

There are also other kind of patterns relevant for classification in some other cases. In the 4-SPP examples wrongly classified, the occlusion of the pod ends further decrease the probability assigned to the correct class. The opposite is true for the 3-SPP examples wrongly classified as 4-SPP: the ends at the right of both examples are red colored. In these two cases the pointed region contributed to mislead the net. Finally, it is worth noting that in several cases the net pays attention to the contour of the pod. This can be seen in the 2-SPP examples classified as 3-SPP, where the red regions are focusing on thinning the middle of the pod. This behavior is observed systematically on many 2-SPP and 3-SPP examples as shown in Fig. 8. It is worth noting that although the model correctly classifies these examples, it gives them a lower probability.

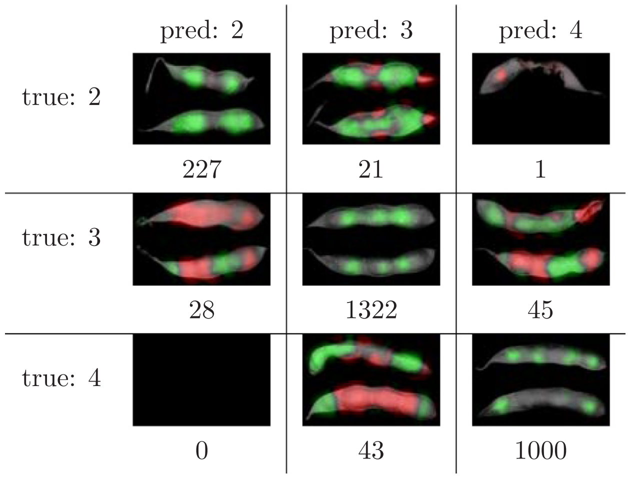

Fig. 7. Confusion matrix with representative samples visualization. Green (red) colored regions indicates regions of positive (negative) correlation with correct class CNN output probability obtained by occlusion experiments. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

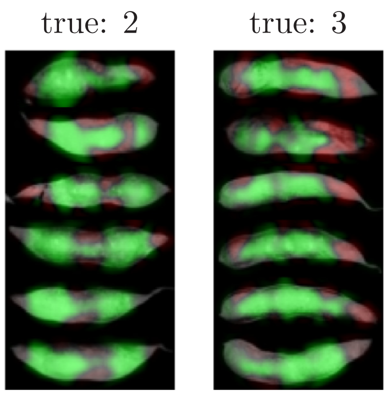

Fig. 8. Visualization of samples where the contour plays an important role. The colors indicate the same as in Fig. 7. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

5. Conclusions

In this work we dealt with the problem of estimating the number of seeds per pod in soybean as an ordinary classification problem using a standard deep learning tool: Convolutional Neural Networks. Although the training of CNNs introduces a large number of hyperparameters, we observed that it is not necessary to choose precise values for them to achieve a high level of accuracy when considering a deep enough architecture.

In addition, CNNs clearly outperformed the classic approach based on pre-designed feature extraction and furthermore, they offer a more robust behavior against seasonal changes.

Finally, using simple visualization techniques we observed that CNNs learnt to individually detect each seed in the pod for most cases, supporting –in such examples– its classification output on this learnt feature. The models also learnt to detect specific shapes in the contour of the pods.

6. Future work

We believe that the capability of extracting simple high level features that can generalize to situations that are clearly different from what was learned makes CNNs ideal for some phenotyping tasks. Our findings add evidence that deep learning techniques can be easily adapted to precision tasks in plant phenology (for example, the detection of desirable growing patterns or the detection of subtle characteristics in leaves, pods, etc.), helping in the improvement of current and future breeding programs.

The proposed methodology can be easily extended to the study of other species and varieties by augmenting the plant database. Moreover, we are currently working in ways to avoid pod extraction and lightbox captures in favor of faster in-field imaging. This outlook involves handling pod (partial/total) occlusions, harder pod segmentation, different pod orientations, etc. However, we believe that the presented CNN models, trained with such in-field images, could be good candidates as a part of a better pipeline for detecting and classifying pods by SPP number from in-field images.

Acknowledgements

The authors acknowledge grant support from ANPCyT PICT-2016-4374.

References

Aich Shubhra, Stavness Ian, 2017. Leaf counting with deep convolutional and deconvolutional networks. arXiv preprint arXiv:1708.07570.
Bendig, Juliane, Bolten, Andreas, Bennertz, Simon, Broscheit, Janis, Eichfuss, Silas, Bareth, Georg, 2014. Estimating biomass of barley using crop surface models (csms) derived from uav-based rgb imaging. Remote Sens. 6 (11), 10395–10412.
Bergstra, James, Bengio, Yoshua, 2012. Random search for hyper-parameter optimization. J. Machine Learn. Res. 13 (Feb), 281–305.
Board, J.E., Harville, B.G., 1998. Late-planted soybean yield response to reproductive source/sink stress. Crop Sci. 38 (3), 763–771.
Bradski, Gary, 2000. The opencv library. Dr. Dobb's J.: Software Tools Profess. Programm. 25 (11), 120–123.
Breiman, Leo, 2001. Random forests. Machine Learn. 45 (1), 5–32.
Chatfield Ken, Simonyan Karen, Vedaldi Andrea, Zisserman Andrew, 2014. Return of the devil in the details: Delving deep into convolutional nets. arXiv preprint arXiv: 1405.3531.
Dieleman Sander, Schlüter Jan, Raffel Colin, Olson Eben, Kaae Sønderby Søren, Nouri Daniel, et al., August 2015. Lasagne: First release.
Ding, Weiguang, Taylor, Graham, 2016. Automatic moth detection from trap images for pest management. Comput. Electron. Agric. 123, 17–28.
Dobrescu Andrei, Giuffrida Mario Valerio, Tsaftaris Sotirios A., 2017. Leveraging multiple datasets for deep leaf counting. arXiv preprint arXiv:1709.01472.
Dorj, Ulzii-Orshikh, Lee, Malrey, Yun, Sang-seok, 2017. An yield estimation in citrus orchards via fruit detection and counting using image processing. Comput. Electron. Agric. 140, 103–112.
Fahlgren, Noah, Gehan, Malia A, Baxter, Ivan, 2015. Lights, camera, action: high-throughput plant phenotyping is ready for a close-up. Curr. Opin. Plant Biol. (Supplement C), 93–99.
Fehr, Walter R., et al., 1987. Principles of Cultivar Development. Theory and Technique, vol. 1 Macmillan Publishing Company.
Fehr Walter R., Caviness Charles E., 1977. Stages of soybean development. Special Report, 87.
Fiorani, Fabio, Schurr, Ulrich, 2013. Future scenarios for plant phenotyping. Ann. Rev. Plant Biol. 64, 267–291.
Ghanem, Michel Edmond, Marrou, Hélène, Sinclair, Thomas R., 2015. Physiological phenotyping of plants for crop improvement. Trends Plant Sci. 20 (3), 139–144.
Gimp Documentation – Curves Tool. https://docs.gimp.org/en/gimp-tool-curves.html.
Giuffrida Mario Valerio, Minervini Massimo, Tsaftaris Sotirios, 2005. Learning to count leaves in rosette plants. In: Tsaftaris, S.A., Scharr, H., Pridmore T. (Eds.), Proceedings of the Computer Vision Problems in Plant Phenotyping (CVPPP), BMVA Press, pp. 1.1–1.13.
Golzarian, Mahmood R., Frick, Ross A., 2011. Classification of images of wheat, ryegrass and brome grass species at early growth stages using principal component analysis. Plant Methods 7 (1), 28.
Grinblat, Guillermo L., Uzal, Lucas C., Larese, Mónica G., Granitto, Pablo M., 2016. Deep learning for plant identification using vein morphological patterns. Comput. Electron. Agric. 127, 418–424.
Harmsen, Stephan R., Koenderink, Nicole J.J.P., 2009. Multi-target tracking for flower counting using adaptive motion models. Comput. Electron. Agric. 65 (1), 7–18.
Hastie, Trevor, Buja, Andreas, Tibshirani, Robert, 1995. Penalized discriminant analysis. Ann. Statist. 73–102.
He Kaiming, Zhang Xiangyu, Ren Shaoqing, Sun Jian, 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778.
Hu, Ming-Kuei, 1962. Visual pattern recognition by moment invariants. IRE Trans. Informat. Theory 8 (2), 179–187.
Ioffe Sergey, Szegedy Christian, 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning, pp. 448–456.
Kamilaris, Andreas, Prenafeta-Boldú, Francesc X., 2018. Deep learning in agriculture: A survey. Comput. Electron. Agric. 147, 70–90.
Keras Documentation – Image Data Generator. https://keras.io/preprocessing/image/.
Kingma Diederik, Adam Jimmy Ba, 2014. A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
LeCun Yann, Boser Bernhard E., Denker John S., Henderson Donnie, Howard R.E., Hubbard Wayne E., Jackel Lawrence D., 1990. Handwritten digit recognition with a back-propagation network. In: Touretzky, D.S. (Ed.), Advances in Neural Information Processing Systems 2, Morgan-Kaufmann, pp. 396–404.
LeCun, Yann, Kavukcuoglu, Koray, Farabet, Clément, 2010. Convolutional networks and applications in vision. In: Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on. IEEE, pp. 253–256.
LeCun, Yann, Bengio, Yoshua, Hinton, Geoffrey, 2015. Deep learning. Nature 521 (7553), 436–444.
Liu, Tao, Chen, Wen, Wang, Yifan, Wu, Wei, Sun, Chengming, Ding, Jinfeng, Guo, Wenshan, 2017. Rice and wheat grain counting method and software development based on android system. Comput. Electron. Agric. 141, 302–309.
Lu, Jiang, Hu, Jie, Zhao, Guannan, Mei, Fenghua, Zhang, Changshui, 2017. An in-field automatic wheat disease diagnosis system. Comput. Electron. Agric. 142, 369–379.
Maas Andrew L., Hannun Awni Y., Ng Andrew Y., 2013. Rectifier nonlinearities improve neural network acoustic models. In: Proc. ICML, vol. 30.
Maldonado, Walter, Barbosa, José Carlos, 2016. Automatic green fruit counting in orange trees using digital images. Comput. Electron. Agric. 127, 572–581.
Mussadiq, Zohaib, Laszlo, Baranyai, Helyes, Lajos, Gyuricza, Csaba, 2015. Evaluation and comparison of open source program solutions for automatic seed counting on digital images. Comput. Electron. Agric. 117, 194–199.
Namin Sarah Taghavi, Esmaeilzadeh Mohammad, Najafi Mohammad, Brown Tim B., Borevitz Justin O., 2017. Deep phenotyping: Deep learning for temporal phenotype/genotype classification. bioRxiv, pp. 134205.
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in Python. J. Machine Learn. Res. 12, 2825–2830.
Pound Michael P., Atkinson Jonathan A., Wells Darren M., Pridmore Tony P., French Andrew P., 2017. Deep learning for multi-task plant phenotyping. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2055–2063.
Pydipati, R., Burks, T.F., Lee, W.S., 2006. Identification of citrus disease using color texture features and discriminant analysis. Comput. Electron. Agric. 52 (1-2), 49–59.
Radford Alec, Metz Luke, Chintala Soumith, 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.
Scharr, Hanno, Dee, Hannah, French, Andrew P., Tsaftaris, Sotirios A., 2016. Special issue on computer vision and image analysis in plant phenotyping. Mach. Vis. Appl. 27 (5), 607–609.
Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556.
Singh, Arti, Ganapathysubramanian, Baskar, Singh, Asheesh Kumar, Sarkar, Soumik, 2016. Machine learning for high-throughput stress phenotyping in plants. Trends Plant Sci. 21 (2), 110–124.
Sladojevic Srdjan, Arsenovic Marko, Anderla Andras, Culibrk Dubravko, Stefanovic Darko, 2016. Deep neural networks based recognition of plant diseases by leaf image classification. Comput. Intell. Neurosci. 2016.
Springenberg Jost Tobias, Dosovitskiy Alexey, Brox Thomas, Riedmiller Martin, 2014. Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806.
Tang, JingLei, Wang, Dong, Zhang, ZhiGuang, He, LiJun, Xin, Jing, Xu, Yang, 2017. Weed

identification based on k-means feature learning combined with convolutional neural network. Comput. Electron. Agric. 135, 63–70.

Theano Development Team, May 2016. Theano: A Python framework for fast computation of mathematical expressions. arXiv e-prints, abs/1605.02688.

Ubbens, Jordan R., Stavness, Ian, 2017. Deep plant phenomics: A deep learning platform for complex plant phenotyping tasks. Front. Plant Sci. 8, 1190.

Umbaugh, S.E., 2005. Computer Imaging: Digital Image Analysis and Processing. CRC Press.

Vinyals Oriol, Toshev Alexander, Bengio Samy, Erhan Dumitru, 2015. Show and tell: A neural image caption generator. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3156–3164.

Wu Stephen Gang, Bao Forrest Sheng, Xu Eric You, Wang Yu-Xuan, Chang Yi-Fan, Xiang Qiao-Liang, 2007. A leaf recognition algorithm for plant classification using probabilistic neural network. In: Signal Processing and Information Technology, 2007 IEEE International Symposium on, IEEE, pp. 11–16.

Yalcin Hulya, 2017. Plant phenology recognition using deep learning: Deep-pheno. In: Agro-Geoinformatics, 2017 6th International Conference on, IEEE, pp. 1–5.

Zeiler Matthew D., Fergus Rob, 2014. Visualizing and understanding convolutional networks. In: Computer Vision–ECCV 2014, Springer, pp. 818–833.