Data Augmentation using Conditional Generative Adversarial Networks for Leaf Counting in Arabidopsis Plants

Yezi Zhu[1]
y.zhu.2@student.tue.nl
Marc Aoun[2]
marc.aoun@signify.com
Marcel Krijn[2]
marcel.krijn@signify.com
Joaquin Vanschoren[1]
j.vanschoren@tue.nl

[1] Department of Mathematics and Computer Science
Eindhoven University of Technology
Eindhoven, The Netherlands

[2] Signify Research
High Tech Campus 7,
Eindhoven, The Netherlands

Abstract

Deep Learning models are being applied to address plant phenotyping problems such as leaf segmentation and leaf counting. Training these models requires large annotated datasets of plant images, which, in many cases, are not readily available. We address the problem of data scarcity by proposing a data augmentation approach based on generating artificial images using conditional Generative Adversarial Networks (cGANs). Our model is trained by conditioning on the leaf segmentation mask of plants with the aim to generate corresponding, realistic, plant images. We also provide a novel method to create the input masks. The proposed system is thus capable of generating realistic images by first producing a mask, and subsequently using the mask as input to the cGANs. We evaluated the impact of the data augmentation on the leaf counting performance of the Mask R-CNN model. The average leaf counting error is reduced by 16.67% when we augment the training set with the generated data.

1 Introduction

Machine Learning and Deep Learning models have shown great potential in addressing plant phenotyping tasks, such as leaf counting [2][19][7], leaf segmentation [22][24], and other related problems.

In many cases, such models rely on a significant amount of data, especially when the approach to solving the phenotyping problem is image-based. Several datasets of top-view plant images have been released [4][18][5] to study new methodologies for leaf counting and leaf segmentation. As it stands, there are limited amounts of labeled images in these phenotyping datasets. The performance of the proposed models can be improved with additional training data. Dataset augmentation techniques (*e.g.*, rotation, scale) are useful, but they do not sufficiently address the lack of labeled training data, given the small size of the original dataset and the fact that these techniques are based on variation of existing items in that dataset. Hence the need for a different approach to overcome the data scarcity problem.

The Generative Adversarial Networks (GANs) model proposed by Goodfellow *et al.* [8] is an architecture that consists of two components: A generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample it is provided with comes from the training data distribution rather than from G. GANs models have achieved impressive results in image generation [21], image editing [25] and other use cases.

A drawback of GANs models is their training instability, which often results in a nonsensical output produced by the generator [21]. Multiple approaches aiming at addressing the training instability problem have been proposed. Radford *et al.* [21] provided a class of architectures named Deep Convolutional GANs (DCGANs), which aims at improving the training stability and the perceptual quality of generated images from GANs. The proposed approach however does not address the root causes of the instability problem; most of the GANs models rely on the Jensen-Shannon (JS) divergence to measure the degree of similarity between the distribution of real images and the distribution of generated images. However, it has been shown that even in very simple scenarios, the JS divergence does not supply useful gradients for the generator [3], resulting in training instability. To better address the problem, Arjovsky *et al.* [3] provided a new architecture, referred to as Wassertein GANs (WGANs), which relies on a perfect discriminator. Although the approach provides stability in training, it comes at the cost of long training time, and potential instability at the discriminator as well as limited capacity at the discriminator due to weight clipping [9]. In order to enforce the Lipschitz constraint without clipping the weights of the discriminator, Gulrajani *et al.* [9] proposed WGANs with gradient penalty (WGAN-GP). The result of WGAN-GP is a much more stable training environment and a higher quality of generated images.

Recently, a conditional DCGAN model named "ARIGAN" has been published by Giuffrida *et al.* [6]. The model can generate images of Arabidopsis plants where the condition is the required number of leaves. Giuffrida *et al.* also put forward a new dataset ("*Ax*") of artificial plant images, which is evaluated in combination with a state-of-the-art leaf counting algorithm. The authors of ARIGAN showed that When *Ax* is used as part of the training data, the testing error is reduced. Similarly, Purbaya*et al.* [20] showed that GANs can synthesize a collection of lanceolate, lyrate and runcinate plant images. However both the ARIGAN model and the work of Purbaya *et al.* are only able to capture the shape of leaves. Both approaches are incapable of generating high frequency components (e.g. leaf veins and petioles), and both cannot generate images with a background.

To solve the aforementioned lack of availability of large training datasets suitable to address phenotyping problems, we present an approach to generate artificial plant images using a Conditional Generative Adversarial model inspired by the work of Mathieu *et al.* [16]. We trained our model on the CVPPP 2017 LSC dataset [18][4]. Our network learned how to map random noise z into an Arabidopsis plant image, under a condition x. In our case, x encodes the leaf segmentation mask that the artificially generated plant image should have. The model can create 512×512 RGB images of Arabidopsis plants. We evaluated our model by first tasking it with generating an image subset, referred to as *Ax*, and subsequently using *Ax* as augmentation to the original training set. The combined dataset is then used as training input to a state-of-the-art instance segmentation algorithm [11] for leaf counting, and the improvement in leaf counting performance is measured relative to the case where only the original dataset is used for training.

The paper is organized as follows: Section 2 describes the overall proposed methodology to generate Arabidopsis plant images; Section 3 reports the results of our experiments and discusses the achieved performance of our model; Section 4 concludes the paper.

2 Methodology

To achieve data augmentation, we generate new plant images by first creating high-resolution masks, and then mapping each mask to a high-resolution, artificial, plant image. We build on the assumption that although the input mask and output image differ in appearance, they present the same underlying structure. Even though the mask has a different texture compared to the output image, our model is still able to extract from it underlying structure information, especially low-level information, *e.g.* edges.

Our model is based on the cGANs structure proposed by the "pix2pix" framework of Isola *et al.* [13]. The model introduces one extra convolutional layer and one deconvolutional layer to the generator as shown in Figure 2, as well as one convolutional layer to the discriminator, as shown in Figure 3. These additional layers are used to generate 512×512 images rather than just the 256×256 images that the model of Isola*et al.* is able to generate.

The overall data generation solution we devise is illustrated in Figure 1. It is based on first automatically generating high-resolution plant masks from leaf mask samples according to pre-defined rules, then providing the generated masks to the conditional GANs model, and finally using the GANs output as artificial data that can serve to augment the original training dataset.

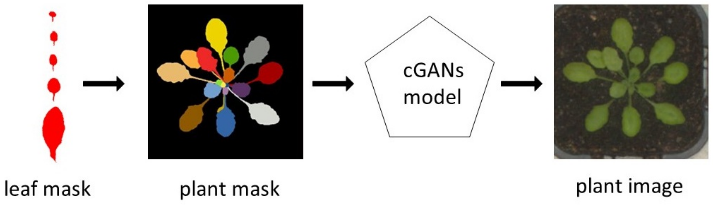

Figure 1: Image Generation Pipeline

The following subsections provide a general description of cGANs and how we train the model, describe the generator and discriminator in our modified cGANs model, present the approach we devise to generate the input masks, and provide an evaluation of the data augmentation technique and its impact on the performance of an instance segmentation model for leaf counting.

2.1 Conditional Generative Adversarial Networks

Conditional GANs learn a mapping from observed image x and random noise vector z, to target image y, $G : \{x,z\} \to y$. The generator G is trained to produce outputs that cannot be distinguished from "real" images by an adversarially trained discriminator, while the discriminator is trained to detect the generator's output as "fakes". The two models are trained simultaneously.

To control the generated image, we use the mask as condition (input), and take the dropout operations as random noise z. The generator network then learns a set of parameters to generate image $G(x,z)$ that follows the distribution of the real training images. The discriminator D learns another set of parameters to classify $x \sim p_{data}$ as real images, and $G(x,z)$ as fake images.

The training process maximizes the probability of D assigning the correct label to y and $G(x,z)$, while G is trained to generate images that follows the same distribution of the real data. We use cross-entropy as loss function to capture the characteristics that occur with high-frequency, and the $L1$ term to force low-frequency correctness.

The objective of the conditional GANs with $L1$ term can be expressed as

$$G^* = arg \min_G \max_D \mathcal{L}_{GAN}(G,D) + \lambda \mathcal{L}_{L1}(G) \tag{1}$$

The formula of the GAN term is

$$\mathcal{L}_{GAN}(G,D) = \mathbb{E}_{x,y}[logD(x,y)] + \mathbb{E}_{x,z}[log(1 - D(x, G(x,z)))] \tag{2}$$

The formula of the $L1$ term is

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x,z)\|_1] \tag{3}$$

We also replace the $L1$ term by an $L2$ term for comparison, with the $L2$ term being given by

$$\mathcal{L}_{L2}(G) = \mathbb{E}_{x,y,z}[\|y - G(x,z)\|_2] \tag{4}$$

The optimization of Formula (1) is achieved using the Adam optimizer [14].

To make a fair comparison, we also conduct experiments with the same loss function used by Gulrajani *et al.* [9], to show why we do not choose a more recent WGAN-GP architecture. The objective of the conditional WGAN-GP(cWGAN-GP) is expressed as

$$\begin{aligned} G^* = & arg \min_G \max_D \mathbb{E}_{x,z}[D(x, G(x,z)))] - \mathbb{E}_{x,y}[D(x,y)] \\ & + \lambda \mathbb{E}_{\hat{x},\hat{z}}[(\|\nabla_{\hat{x}} D(\hat{x}, G(\hat{x},\hat{z}))\| - 1)^2] \end{aligned} \tag{5}$$

2.2 Generator Model

As noted in the findings of Mathieu *et al.* [16], there is variability in the input of the generator even in the absence of noise, which means noise is not necessary for the generator to obtain variations when it generates a new image. Inspired by the work of Isola *et al.* [13], we provide noise only in the form of dropout in the first four decoder layers of the generator.

As shown in Figure 2, we use modules of Leaky ReLu-BatchNorm-convolution[12][10] in the encoder part of the generator, and modules of ReLu-BatchNorm-convolution in decoder part of the generator. Furthermore, to pass low-level features directly to deeper layers across the network, we add skip-connections between each layer i and layer $n-i$, where n is the total number of layers, following the general setting of "U-Net"[23]. The original model only generates 256×256 images, which is not consistent with the image size of the Arabidopsis plants dataset. We therefore add one convolutional layer and one deconvolutional layer to the generator to obtain 512×512 images, and then rescal them to 441×441(same size as the images of the $A4$ subset).

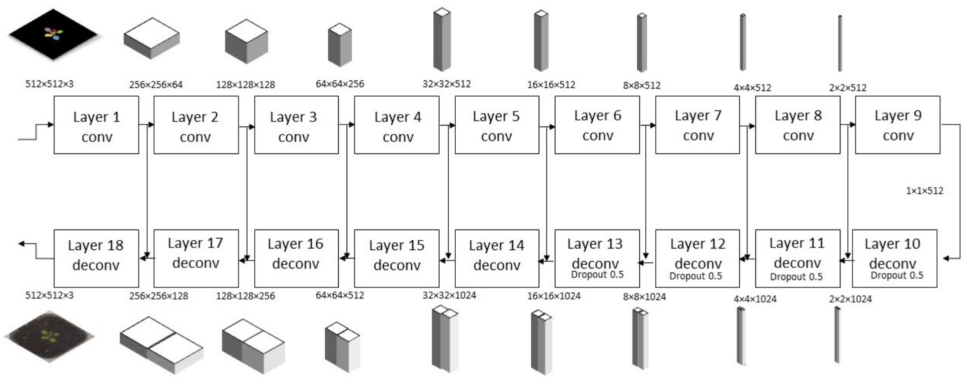

Figure 2: **Generator.** The network takes condition vector y as input, and takes the dropout operations as variable z (random noise). The numbers denote spatial resolution and channels.

The input layer takes the mask image as a condition. This input is then provided to a fully-convolutional layer *conv*1, which is followed by 8 convolutional layers and 8 deconvolutional layers. After the last fully convolutional layer with tanh activation function, the generated image is ready.

The deconvolutional steps are achieved through the use of $(2,2)$ stride convolutional layers. We adopt a 4×4 kernel size for all convolutional and deconvolutional layers. The output of each layer is batch-normalized and passed through leaky ReLU at the encoder side and ReLU at the decoder side, before it is provided to the next layer.

2.3 Discriminator Model

Our discriminator tries to classify each $N \times N$ patch in an image as being either real or fake. We run this discriminator convolutionally across the image, averaging all responses to provide the ultimate output of the discriminator. We implement the convolution-BatchNorm-ReLu module here.

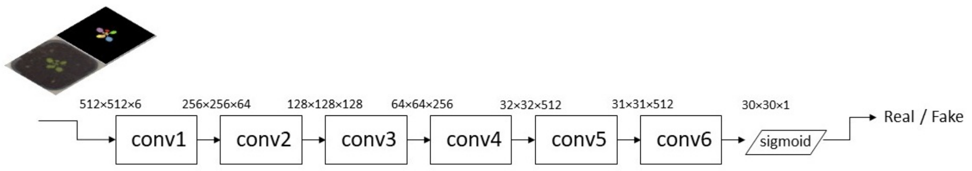

Figure 3: **Discriminator.** The network takes an RGB image concatenated with condition mask image x as input. Numbers denote spatial resolution and channels.

The discriminator network, as illustrated in Figure 3, takes an RGB image concatenated with condition mask image x as input. There are 6 convolutional layers, and the last layer of the network is a single node that outputs a binary value activated by a sigmoid function. The discriminator uses Leaky ReLu [15] at each layer, which shows a fast loss convergence during training.

2.4 Mask Generation

We collect 7016 leaf masks from the test set(500 images) of the CVPPP A4 subset. To make the plant mask generation more convenient, we pick 250 leaf masks from the collection of leaf masks, and divide them into 5 folders based on the leaf mask size, as shown in Figure 4. We generate plant masks automatically based on the rules detailed in Algorithm 1.

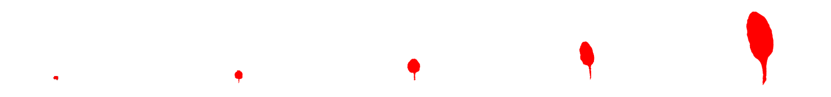
(a) folder 1 (b) folder 2 (c) folder 3 (d) folder 4 (e) folder 5

Figure 4: Examples of leaf masks of different sizes.

Algorithm 1: Mask Generation

```
   Data: The number $n$ of leaves the plant mask should contain.
   Result: Plant mask. A mask consists of many regions of different colors. Each region
           corresponds to a leaf segmentation.
 1 if $n \leq 4$ then
 2     choose $n$ leaf masks randomly from folder 1.
 3 else if $4 < n \leq 8$ then
 4     select $n - 4$ leaf masks from folder 2, and 4 leaf masks from folder 1.
 5 else if $8 < n \leq 12$ then
 6     select $n - 8$ leaf masks from folder 3, 4 leaf masks from folder 2, and 4 leaf masks
         from folder 1.
 7 else if $12 < n \leq 16$ then
 8     select $n - 12$ leaf masks from folder 4, 4 leaf masks randomly from folder 3, 4
         leaf masks from folder 2, and 4 leaf masks from folder 1.
 9 else
10     select $n - 16$ leaf masks from folder 5, 4 leaf masks from folder 4, 4 leaf masks
         from folder 3, 4 leaf masks from folder 2, and the rest 4 leaf masks from folder 1.
11 end
12 Each leaf mask has a $140 - 200$ degrees random rotation, compared to the previously
     selected (and rotated) one. It also has a $-10\%$ to $10\%$ zooming, selected randomly.
13 As a final step, overlay the leaf masks one by one.
   Note that as a result of this procedure, the smaller leaf masks are naturally positioned
     on top of the larger ones.
```

2.5 Dataset

We train our model on the CVPPP 2017 LSC plant dataset [18][4], containing Arabidopsis plant images. The dataset, which also includes Tobacco plant images, consists of 4 subsets ($A1$, $A2$, $A3$, and $A4$). We select the $A4$ subset for our study, since it has the largest number of Arabidopsis images (624 images in total), has flower pot background information, and as such is the only one among the four subsets large enough to provide all the information we need in this experiment. 441×441 input images are pre-processed to be all in the same size of 448×448 through the application of padding. 500 images (80%) of the $A4$ subset are used as training set for both the cGANs model and the leaf counting model, and 124 images(20%) are used as test set. No augmentation such as rotation and scale is used for training the cGANs model. Flipping the images left/right and up/down as well as rotation by 90, 180, and 270 degrees are used as data augmentation techniques for the training of the leaf counting model. Image pixel values are normalized to a range of $[-1, 1]$.

2.6 Hyper-parameters

To optimize the cGANs model, we alternate between one gradient descent step on D, and one step on G, training to maximize log $D(x, G(x, z))$. We use Adam solver[14], with learning rate 0.0002, and momentum parameters $\beta_1 = 0.5$. The weight for gradient penalty λ is set to 0.25 for WGAN-GP experiment. The higher λ is, the more stable the training, but the slower the convergence. The model is trained on 500 images, with 200 epochs, and a batch size equal to 1.

We implement the model in Tensorflow [1]. The training is performed on a GTX 1080Ti GPU. In our setup, training takes ~ 1 minute per epoch.

2.7 Evaluation Metrics

Using the overall data augmentation pipeline, we artificially generate a subset Ax of plant images. The subset contains 500 images in total, along with the corresponding segmentation masks. The segmentation mask images are just the input images of cGANs model that were used to generate the artificial plant images.

We evaluate the impact of augmenting the training set with Ax on the performance of the leaf counting algorithm (Mask RCNN model[11][17]). In particular we measure the average leaf counting error of the algorithm under three training conditions: Training using only the A4 dataset, training solely based on the Ax dataset, and finally, training based on the $A4$ and Ax datasets combined.

To evaluate segmentation accuracy, we employ the Symmetric Best Dice (SBD) metric used by Scharr *et al.* [24]. SBD is used to estimate average leaf segmentation performance. It corresponds the symmetric average dice among all objects (leaves), where for each input label the ground truth label yielding maximum Dice is used for averaging. High SBD value indicates a good performance of instance segmentation.

SBD between L_{gt}, the ground truth, and L_{ar}, the algorithmic result, is defined as

$$SBD(L_{ar}, L_{gt}) = \min(BD(L_{ar}, L_{gt}), BD(L_{gt}, L_{ar})) \tag{6}$$

with Best Dice (BD) being defined as

$$BD(L^a, L^b) = \frac{1}{M} \sum_{i}^{M} \max_{1 \le j \le N} \frac{2|L_i^a \cap L_j^b|}{|L_i^a| + |L_j^b|} \tag{7}$$

where $|\cdot|$ denotes leaf area (number of pixels), and L_i^a for $1 \le i \le M$ and L_j^b for $1 \le j \le N$ are sets of leaf object segments belonging to leaf segmentations L_a and L_b, respectively.

To evaluate how good an algorithm is in identifying the correct number of leaves present in an image, we rely on the Difference in Counting(DiC) and Absolute Difference in Counting($|DiC|$ metrics.

$$DiC = \frac{1}{N}\sum_{i}^{N}(\#L_{ar}^{i} - \#L_{gt}^{i}) \tag{8}$$

$$|DiC| = \frac{1}{N}\sum_{i}^{N}|\#L_{ar}^{i} - \#L_{gt}^{i}| \tag{9}$$

where N is the number of images, L_{ar} is the predicted leaf number, and L_{gt} is the ground truth leaf number.

3 Experimental results

We aim at identifying whether our data augmentation technique results in better leaf counting performance. Our investigation covered multiple variants of the model from which the Ax dataset is obtained. We ran several experiments to compare between the cGANs model guided by only a GAN term in the loss function, the cGANs model using a loss function with extra $L1$ term or $L2$ term, and the cGANs model with conditional WGAN-GP improvements. This section provides the obtained qualitative and quantitative results.

3.1 Qualitative results

Figure 5 shows the qualitative effects of the different loss function variations on the generation of plant images. A loss function that only contains the GAN term leads to reasonable results, but introduces visual artifacts along the edge of the flower pot, and has some checkerboard artifacts. Adding both the GAN term and the $L1$ term reduces such artifacts and helps to obtain an Arabidopsis plant image with clear pot surrounding background.

Adding the $L2$ term provides a sharper and more contrasting image compared to the $L1$ term version. The edges of the leaves are clearer, but the checkerboard artifacts and unreasonable details along the flower pot still exist. A loss function based on a combination of the GAN term and the $L1$ term provides comparable performance to a loss function based on a combination of the GAN term and the $L2$ term.

Due to how the loss is calculated in cWGAN-GP, no $L1$ term or $L2$ term can be added to the loss function. Images generated via cWGAN-GP are less blur and have fewer artifacts than the one generated via a GAN term that does not use any extra loss term. However, images obtained through cWGAN-GP still have more artifacts than the ones obtained through the cGANs + $L1$ term variant, and the images generated by the cGANs + $L2$ term variant. Therefore, qualitatively, cWGAN-GP is not a good alternative to these two $cGANs$ variants.

We hereafter further provide a comparison to establish whether it also underperforms these two variants quantitatively.

3.2 Quantitative results

To investigate the performance of our cGANs-based data augmentation method, we evaluated the aforementioned leaf counting algorithm on three different datasets, namely; $A4$ only, Ax only, and the dataset resulting from combining $A4$ and Ax. The results are shown in Table 1. In addition to providing DiC, |DiC| and percentage SBD values, the table shows

4 Conclusion

We proposed a novel approach to data augmentation of plant image datasets for improving the performance of leaf counting models in phenotyping applications. From our experiments, we found that our model learned how to generate realistic plant images after 200 epochs. We created an artificial Arabidopsis plant image subset via the model. We first evaluated the quality of the dataset by observing the qualitative characteristics of the generated images. We subsequently provided a quantitative evaluation of the model by using the generated dataset as a data augmentation to existing data, prior to training a state-of-the-art instance segmentation model and measuring its resulting leaf counting performance. Our quantitative experiments show that the extension of the training dataset with the images in Ax reduced the average leaf counting error by 16.67%. The proposed approach can effectively address the annotated data scarcity problem encountered when solving phenotyping problems.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning.

[2] Shubhra Aich and Ian Stavness. Leaf counting with deep convolutional and deconvolutional networks.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[4] Jonathan Bell and Hannah M Dee. Aberystwyth leaf evaluation dataset. 2016.

[5] Jeffrey A Cruz, Xi Yin, Xiaoming Liu, Saif M Imran, Daniel D Morris, David M Kramer, and Jin Chen. Multi-modality imagery database for plant phenotyping. *Machine Vision and Applications*, 27(5):735–749, 2016.

[6] Mario Valerio Giuffrida, Hanno Scharr, and Sotirios A Tsaftaris. Arigan: Synthetic arabidopsis plants using generative adversarial network.

[7] Mario Valerio Giuffrida, Massimo Minervini, and Sotirios A Tsaftaris. Learning to count leaves in rosette plants. 2016.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[9] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5769–5779, 2017.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.

[12] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[13] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks.

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models.

[16] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

[17] matterport. Mask r-cnn for object detection and segmentation(keras implementation), 2017. URL `https://github.com/matterport/Mask_RCNN`.

[18] Massimo Minervini, Andreas Fischbach, Hanno Scharr, and Sotirios A Tsaftaris. Finely-grained annotated datasets for image-based plant phenotyping. *Pattern recognition letters*, 81:80–89, 2016.

[19] Jean-Michel Pape and Christian Klukas. Utilizing machine learning approaches to improve the prediction of leaf counts and individual leaf segmentation of rosette plant images.

[20] Muhammad Eka Purbaya, Noor Akhmad Setiawan, and Teguh Bharata Adji. Leaves image synthesis using generative adversarial networks with regularization improvement.

[21] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[22] Mengye Ren and Richard S Zemel. End-to-end instance segmentation with recurrent attention.

[23] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[24] Hanno Scharr, Massimo Minervini, Andrew P French, Christian Klukas, David M Kramer, Xiaoming Liu, Imanol Luengo, Jean-Michel Pape, Gerrit Polder, Danijela Vukadinovic, et al. Leaf segmentation in plant phenotyping: a collation study. *Machine vision and applications*, 27(4):585–606, 2016.

[25] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016.