IMT School for Advanced Studies Lucca
Lucca, Italy

LEARNING TO COUNT LEAVES OF PLANTS

Ph.D. Program in Image Analysis
XXX Cycle

By
MARIO VALERIO GIUFFRIDA
2018

http://www.valeriogiuffrida.academy

The dissertation of Mario Valerio Giuffrida is approved.

Ph.D. Program in Institutions, Markets and Technologies
Curriculum in Image Analysis
Curriculum Director: Dr. Sotirios Tsaftaris

Advisor:

Sotirios Tsaftaris
The University of Edinburgh

Co-Advisor(s):

Emiliano Ricciardi
IMT School for Advanced Studies Lucca

The dissertation of Mario Valerio Giuffrida has been reviewed by:

Andrew French, University of Nottingham
David Rousseau, University of Angers

IMT School for Advanced Studies Lucca
2018

Table of Contents

List of Figures **x**

List of Tables **xx**

Acknowledgements **xxiv**

Vita and Publications **xxv**

Abstract **xxix**

I Introduction and Background **2**

1 Motivations **3**
1.1 Challenges 7
1.2 Research Questions 9
1.3 Contributions 11
1.4 Thesis Structure 12

2 Background **16**
2.1 Plant Biology Background 16
2.2 Machine Learning Background 19
2.3 Deep Learning Background 21
2.3.1 Under The Hood 22
2.3.2 Convolutional Neural Network 23
2.3.3 Computational Demand 26
2.4 Evaluation Measures 28

3 Prior Art **30**
3.1 Overview 31
3.2 Holistic Approaches 31

3.3 Local Approaches 34
3.4 Hybrid Approaches 37
3.5 Leaf Counting Methods 38
3.5.1 Counting by Segmentation 38
3.5.2 Counting by Regression 40

II A Shallow Model for Leaf Counting 42

4 A Shallow Leaf Counting Direct Regression Model 43
4.1 Proposed Approach 44
4.1.1 Log-polar Representation 45
4.1.2 Patch Extraction 47
4.1.3 Unsupervised Feature Learning 48
4.1.4 Holistic Regression 50
4.2 Experimental Results 51
4.2.1 Setup 52
4.2.2 The *Inner-Outer Leaf Counting* variant 53
4.2.3 Results 54
4.3 Discussion 59

5 Learning Rotation-Invariant Features 62
5.1 Background 64
5.1.1 Some Feature Properties 64
5.1.2 Restricted Boltzmann Machine 65
5.2 Related Works to Rotation Invariance Learning . . . 69
5.3 Explicit Rotation-Invariant Restricted Boltzmann Machine 70
5.3.1 Proposed Model 71
5.3.2 Experimental Results 75
5.4 Determining the Dominant Orientation with an Endogenous Process 78
5.4.1 Theory 78

5.4.2 Inference of the Dominant Orientation 87
5.4.3 Experimental Results 91
5.5 Experiments with Plant Data 98
5.5.1 Implementation Details 98
5.5.2 Results and Discussion 100

III Strategies to Obtain More Labelled Data 102

6 Annotation Tool to Assist Experts 103
6.1 Proposed Method 106
6.1.1 Interactive Segmentation with Random Walks 108
6.2 The Annotation Tool 111
6.3 Experimental Results 113
6.4 Integration on Phenotiki 116
6.4.1 Sensor Specifications 117
6.4.2 Analysis Software 118
6.4.3 Software Validation 121
6.5 Impact of Phenotiki 125
6.6 Extending Phenotiki 126
6.7 Discussion 131

7 Crowd Sourcing and Inter-Observer Variability 132
7.1 Motivation 133
7.2 Methods 135
7.2.1 Employed Data 135
7.2.2 Study Design 136
7.2.3 Citizen-Powered Study 137
7.2.4 Statistics and Evaluation Metrics 138
7.3 Results 139
7.3.1 Intra-Observer Variability 139
7.3.2 Variability Between Tool and Spreadsheet-Based Counting 141

7.3.3 Inter-Observer Variability 141
7.3.4 Influence of Resolution on Intra-Observer Variability 145
7.3.5 Influence of Observer Variation in Longitudinal Analysis 145
7.3.6 Time Results 148
7.3.7 Simulating a Citizen-Powered Study 148
7.3.8 Results from the Citizen-Powered Study . . . 150
7.3.9 Variability Between Algorithmic Leaf Count and Experts 153
7.4 Discussion 155

8 Arabidopsis Rosette Image Generator (through) Adversarial Networks 160
8.1 Generative Adversarial Networks 162
8.2 Methodology 163
8.2.1 Mathematical Background 163
8.2.2 The Model *G* 164
8.2.3 The Model *D* 165
8.3 Experimental Results 166
8.3.1 Dataset 166
8.3.2 Parameter Selection 168
8.3.3 Qualitative Results 168
8.3.4 The *Ax* Dataset 170
8.4 Discussion 171

IV Deep Learning for Leaf Counting 174

9 Pheno-Deep Counter: The Versatile Leaf Counting Deep Network 175
9.1 The Network Architecture 177
9.2 Results 181
9.2.1 Proof-of-concept: data agglomeration helps . 183

9.2.2 Evaluation and comparison with state-of-the-art on the CVPPP 2017 dataset 186
9.2.3 Multiple Modalities and Leaf Counting . . . 188
9.2.4 Evaluation of Network Adaptivity Capabilities 191
9.3 Discussion 195

V Conclusions and Future Work 198

10 Conclusions 199
10.1 Summary 199
10.2 Findings 200
10.3 Limitations 203

11 Future Work 206
11.1 Future Work in Leaf Counting 206
11.1.1 Learning from Noisy Labels 206
11.1.2 Multi-Task Learning 208
11.1.3 Learning to Predict Integer Numbers 208
11.2 Future Work in Data Collection 208
11.2.1 Cross-Modal Image Synthesis 208
11.2.2 Synthesis of Plant Images with a Per-Leaf Segmentation 209

Bibliography 210

Appendices 226

A Neural Network Activation Functions 226

List of Figures

1.1 Vegetable (crop) vs. meat (livestock) productivity. Scales are expressed in billions of tonnes. Data from the Food and Agriculture Organization (FAO) accessed on December 2017 (FAOSTAT dataset). 4
1.2 Population growth (current data and future predictions) vs. available arable land growth. Scales are expressed in billions of people (blue line) and billions of hectares (orange line). Data from the Food and Agriculture Organization (FAO) accessed on December 2017 (FAOSTAT dataset). 5
1.3 Some mutants of the plant Arabidopsis, showing high intra-species leaf shape variability. (Figure adapted from [1]). 8

2.1 Schematic of the *Arabidopsis Thaliana* life cycle (data taken from [2]). 17
2.2 Example of a plant tray with 24 plants of *Arabidopsis Thaliana* (a) from different 5 ascensions (b). In (c), we show a time lapse of a plant subject development in 24 days (image adapted from [3]). 18
2.3 Anatomy of an Arabidopsis leaf. 19
2.4 Example of neural network. 22
2.5 Schematic of a Convolutional Neural Network for a classification or regression task. 24
2.6 Convolution operation applied between an image I of size 5×5 and a kernel W of size 3×3. 25

2.7 Evolution over time of Nvidia GPUs supporting CUDA vs the popularity of deep learning (Google Trend; *cf* Section 2.2).[1] (GFLOPS = Giga floating-point operations per second.) Shaded areas display the Nvidia GPUs' microarchitecture evolution. The latest microarchitecture (Volta) is not displayed as no benchmark data are available at this time. 27

3.1 Example of dot annotations on a frame taken from UCSD dataset [4]. 34
3.2 Counting pipeline of the method in [5]: (a) features are extracted in K cells, (b) features are used to train a multivariate ridge regressor, which provides (c) predictions simultaneously for all the cells. 35
3.3 Sample of RGB Arabidopsis Thaliana Col-0 image taken from [6] (a), with its corresponding per-leaf segmentation mask (b). 39

4.1 Some examples of the A1, A2, and A3 images from the CVPPP 2015 dataset. 44
4.2 Flowchart describing the leaf counting method we proposed for the CVPPP Workshop 2015. 45
4.3 Centre detection in a complex object. Shown are: (a) plant mask available from expert annotation (together with classical calculations of a centre and proposed); (b) skeleton obtained from (a) and endpoints as red dots; and (c) most traversed segment, with detected centre marked with a white cross. 46
4.4 Example images (background was removed) of Arabidopsis taken from the A1 (top), and A2 (middle), datasets respectively, and tobacco taken from A3 (bottom). First and third columns show the same plant (with leaf centre annotations in purple) few days after. Second and fourth columns show the corresponding log-polar representations. 47

4.5 FG/BG ratio: a sliding window moves rightwards to compute the ratio between the number of foreground and background (black) pixels within it. Observe that we have local maxima even when leaves overlap. . . 48
4.6 Features learned with $K = 50$ using patches taken from the log-polar representation of the plants. . . . 49
4.7 Pooling regions in the log-polar image. 50
4.8 Graphical example how the inner and the outer part of a plant is determined: the log-polar image is scanned from the top downwards until any background pixel is found. The separation line in the log-polar representation corresponds to a circle enclosing the centre of a plant, containing the smallest leaves. 54

5.1 Learned features in [7] (c.f. Chapter 4) show centroids that are rotated versions of each others. 63
5.2 Visual example of rotation invariance. The two versions of a leaf share the same representation. 64
5.3 Representation of an RBM. An input vector x is given to the network to compute the probability of observing h given x. During optimisation, the best values for the weight matrix W and bias vectors $\mathbf{c}$ and $\mathbf{b}$ are found. 66
5.4 Graphical representation of Gibbs sampling. An input is provided to the visible layer to compute Equation (5.6). From this distribution, samples are drawn and a $\mathbf{h}$ vector is computed. The, this vector is used to compute the reconstructed data $\tilde{\mathbf{x}}$. The representation $\tilde{\mathbf{h}}$ is computed for the reconstructed data. These four vectors are used to determine the gradients to update the parameters in Θ. 67
5.5 Example of learned filters using RBM on the MNIST-rot dataset [8]. 68

5.6 The dominant orientation ϕ^t is determined for the provided image and is used to compute the gradient $\nabla W^{(t)}$. The contribution of this gradient is shared amongst the other weight matrices $\nabla W^{(s)}$, $s = 1, 2, \ldots, S$, $t \neq s$, rotating the learned filters by the angle $\phi^s - \phi^t$ to generate the $\nabla \mathring{W}^{(t)}$ term. 72

5.7 Computation of the dominant orientation for a sample image taken from the MNIST dataset: (a) original sample, (b) gradients of the image, (c) histogram of oriented gradients with highlighted mode ψ, (d) sample rotated by ψ degree. The region marked by a green ellipse corresponds to the same portion of the number 3 in the original and rotated image. Observe the differences due to image interpolation introduced during rotation. 74

5.8 Filters learned by our ERI-RBM at $S = 9$. We highlight a filter that appears at rotations $0°$, $40°$, $80°$, and $120°$, showing that our model learns rotation-invariant filters. The remaining weight matrices are omitted for brevity. 77

5.9 Representation of the proposed rotation-invariant RBM. In this example, we have $S = 4$ rotations, corresponding to the equidistant angles $\Phi = \{\phi_0 = 0°, \phi_1 = 90°, \phi_2 = 180°, \phi_3 = 270°\}$, each of those associated with a matrix W_s. When an image is provided to the network, the weight matrix minimising the reconstruction error is chosen, as highlighted in bold red. We depict the unfolded steps of the CD-1 [9]. . 79

5.10 Graphic representation of the shared gradient step. Following the example in Figure 5.9, the gradient ∇W_2 for the slice W_2 is computed. By applying proper transformations, rotated versions of ∇W_2 are applied to the other slices in W, as shown above. 82

5.11 Probability (in log-scale) of a prediction change of the dominant orientation occurring during training on the *mnist-rot* dataset (*change probability*). The stability of our inference method increases exponentially over time. We also show two inset transition matrices: the left-hand side inset shows the amount of misclassification between the first two epochs, whereas the right-hand side for the last two epochs. 89
5.12 Sample images of the employed datasets. *Top row*: mnist-rot [8]. *Middle row:* MPEG-7 Shape Silhouette database [10]. *Bottom row:* rotated version of the zalando fashion mnist dataset (original from [11]). . . 91
5.13 Probability (in log-scale) that a change in inferring the dominant orientation occurs during training. Epochs 20, 40, 60, and 80 are marked with a thicker grey line, highlighting the times which random errors are introduced in the training. Overall, the injection of noise at regular interval does not affect the learning process. The probability of a change in inferring the dominant orientation decreases asymptotically (c.f. Figure 5.11). We plot baseline (no changes), 20%, and 40% for brevity. 97
5.14 Three concentric pooling regions 99

6.1 Example of a rosette plant (*A. thaliana*) and annotations: (a) Original image, (b) plant segmentation mask, (c) leaf segmentation ground truth, and (d) Zoom-in of (a), showing examples of different types of scribbles that can be used as input to our tool. . . 105
6.2 Example images (background was removed) of Arabidopsis taken from the A1 (left), and A2 (middle), datasets respectively, and tobacco taken from A3 (right). Rows show the same plant (with leaf center annotations in purple) at different stages of development. 106

6.3 Example of 1-dimensional discrete random walk. The probability p defines the likelihood of a random walker to go right ($q = 1 - p$ is for leftwards steps). 109
6.4 Graphical user interface of our annotation tool. In this example, we add as many seeds as the number of leaves and the underlying random walks segmentation algorithm is able to propagate those seeds and provide an accurate leaf delineation. 111
6.5 (a)-(c) Example annotations used to evaluate the random walks segmentation algorithm. (d) Leaf segmentation obtained using the annotations in (c), which matches almost perfectly the (e) ground truth leaf mask (obtained by manually labelling each pixel [12]).114
6.6 Overview of the Phenotiki system and screen captures showing the graphical user interfaces to operate its hardware and software components. (a) Schematic of the proposed distributed sensing and analysis framework illustrating the main components of our phenotyping platform. (b) Web interface to configure and operate the Phenotiki device from the browser. (c) Stand-alone version of the image-analysis software. (d) Cloud-based version of the image analysis software that runs on a web browser. 117
6.7 Pictures of the proposed affordable Phenotiki device. 118
6.8 Screen captures showing the user interface of our stand-alone plant image analysis software. 119
6.9 Leaf-counting data (a, b), estimated by our automated leaf-counting algorithm and (c, d) derived from the expert annotations. Results are shown as (a, c) time-series plots (mean and standard deviation) and (b, d) growth progression bars [2]. The learning-based counting algorithm was trained on a subset of plant images and then applied to the entire dataset. 122

6.10 Number of downloads of the *Phenotiki Analysis Software* per country. 127
6.11 Camera station setup to acquire images of chickpea RSA. (A) Camera station; (B) LED strip close-up; (C) Phenotiki sensors arrangement. The image station is covered by black velvet during acquisition. 128
6.12 Chickpea images stitching. (A1) Robust keypoint detection and matching [13]; (A2) Pairwise image stitching; (B) Full image stitching result; (C) Root segmentation. 130

7.1 Screenshot of the Zooniverse site used here showing annotations and the confidence question. 137
7.2 A Intra-observer variability of experienced (A1) or non-experienced (A2) observers in RPi. B Influence of the tool in intra-observer measurements in experienced (B1) or non-experienced (B2) observers in RPi 140
7.3 Inter-observer and influence of resolution. A: Inter-observer variability among experienced (A1) or non-experienced (A2) observers in RPI; B: same as in A but in Canon data. (*This figure continues on the next page*) 142
7.4 Average longitudinal count curves (solid) of the two cultivars [red: *col-0*; blue: *pgm*] and 1 standard deviation (shaded area), shown in A: relying on a single experienced (left: A1) or non-experienced observer (right: B1); B: relying on all experienced (left: B1) or non-experienced (right: B2) observers; C: relying on all together; and in D: relying on the consensus citizen147

7.5 Citizen distribution and variability. A) Number of images annotated per user (citizen); B) Relationship between leaf count variation and average user confidence per plant; C) Variability between the consensus citizen and the reference observer; D) Variability between the consensus citizen and a random selection of counts (from the 3 available per-plant). 151

8.1 Schematic of the proposed method: a conditional generative adversarial network is trained to map random uniform noise z into *Arabidopsis* plants, given a condition y on the number of leaves to generate. . . 161

8.2 The generator takes as input a variable z (random noise) together with the condition vector y. These inputs are then provided to two fully connected layers, where *fc2* has the same amount of hidden units of the first deconvolutional layer. The information is then processed through 5 deconvolutional layers, where the last one provides an 128x128 RGB image. The condition y is applied to all the stages (fc and deconv layers). We showed it in the input only for sake of clarity. 164

8.3 The discriminator takes as input an RGB image concatenated with the condition vector y properly reshaped to be stacked as an additional channel. The rest of the network is a reversed version of G (c.f. Figure 8.2). The last node of the network is a binary classifier that discriminates between real and generated (fake) images. 166

8.4 Images used to train ARIGAN. We used the segmentation mask to relax the learning process, due to the significant variability of these setups. 167

8.5 Fixed sample noise z is provided to the generator during training of ARIGAN. The number reported in the bottom right corner of each image refers to the epoch number. 169
8.6 Samples from the Ax dataset generated with ARIGAN. Bottom-right numbers refer to the leaf count. 170

9.1 Schematic of the proposed deep architecture. (A) a modality branch, consisting of ResNet50 [14], extracts modality-dependent plant features as a feature vector of 1,024 neurons. (B) The fusion part combines those features to retain the most useful information from each modality. (C) The regression part, relates fused information with leaf count as a non-linear regression. 178
9.2 Sample images of the employed datasets. *First row:* RGB, near-infrared, and fluorescence images of the same plant from the multi-modal imagery database for plant phenotyping [15]. *Second row:* images from the A1, A2, A3, and A4 datasets from CVPPP 2017 [6, 12, 16]. *Third row:* samples of Komatsuna plants from [17]. *Last row:* samples of nocturnal images of Arabidopsis plants in [18]. 182
9.3 Visual diagram showing which part of a plant image contributes the most for the counting. We shift a 60×60 black patch entirely over a plant image and we show that areas corresponding to the plant gives the highest contribution to the count. In the top-left corner of each image we report the ground-truth (GT) leaf count of the plant. 185

9.4 Leaf count prediction in the CVPPP dataset (all images altogether). (a) Ground-truth vs. prediction, shown as a scatter plot. Due to integer values color shows how many points are overlapping. Dashed parallel lines show the ± 1 leaf error range. Note that our approach has high agreement w.r.t. the real leaf count. (b) error distribution. Observe that there is 83% chance that the error will be ± 1 within 0 (green area), a number close to the agreement among human observers ($\sim 90\%$; c.f. Chapter 7). 186
9.5 Activations after the first residual block in the RGB, IR, and FMP modality branches. The output of this block layer consists of 256 feature maps. We display the mean for each pixel. 190
9.6 Error distribution of our network fine-tuned using Tobacco plants in A3 dataset [6]. We reported the distribution of the error committed in the testing set, after refining the network parameters with 7 Tobacco plants (a), 14 (b), 21 (c), and 27 (d) plants. When we train with more images (>=21), the green area (error up to ± 1 leaf, c.f. Figure 9.4) contains more than 80% of the cases. 193

List of Tables

3.1 Features extracted in [19]. 33

4.1 Training results of our proposed method (IOLC and GLC versions). 55
4.2 Evaluation of different methods to determine the centre of a plant. Bold values indicate best performace. 56
4.3 Training results of our proposed method (IOLC and GLC versions) using the *augmented* dataset. 57
4.4 Results for the testing set of our proposed GLC method with regressor(s) and features learned on the *augmented* dataset. For comparison the findings of *Pape and Klukas* [20] on the same testing set are shown (values for only two metrics were available). 58

5.1 Testing accuracies of standard RBM, Dominant RBM, Oriented RBM, TI-RBM [21], and our proposed ERI-RBM. 76
5.2 Test accuracy of our method compared with SVM [22], the original formulation of RBM [23], the transformation-invariant RBM [21], and the explicit rotation-invariant RBM [24] on three different datasets. We report *best result (mean ± std)* [25] of 5 random initialisations. . 93
5.3 Ablation results showing that our method benefits from shared gradients and the regulariser presented in Equation (5.35). We also report the upper bound (ours trained with actual rotations) values for all the applicable datasets. (We could not perform this on the *MPEG-7 database* [10], as the images are already transformed and their rotations are unknown). . . . 95

5.4 Testing accuracy on *mnist-rot* dataset of our architecture trained by altering the inference of the dominant orientation at gradually increasing portions of the training set. 98

6.1 Currently publicly available labelled datasets for plant phenotyping (with the number of labelled images). 104
6.2 Leaf segmentation accuracy (*SymmetricBestDice*, expressed as %) obtained by the random walks segmentation approach [26] adopted here, using different types of leaf annotations. Ref. [20] is adopted as baseline. Results are shown as *mean* $\pm$ *standard deviation*. 115
6.3 Quantitative performance of the leaf counting algorithm in Phenotiki. 124
6.4 Statistical significance of differences in leaf count between genotypes is not affected by using the leaf count algorithm. Shown is the pairwise post-hoc comparison with the Tukey-Kramer method between leaf count data of Col-0 and the other genotypes, following a two-way repeated measure ANOVA testing once on RaspiCam derived data analysed with Phenotiki and once on manual annotation by an expert, separately. No difference in the significance of the tests is observed between results obtained with Phenotiki and manual annotation, respectively. Sample size (24 subjects, 52 time points) is equal among the two tests. 126

7.1 Measurement of agreement between experienced and non-experienced observers. For shorthand definitions see text. For DiC and |DiC| average and standard deviation are reported. Note that these correspond also to bias and limits of agreement (when standard deviation is multiplied by 1.96) of the Bland-Altman plots reported. ↓ means lower is better, whereas ↑ means higher is better. 144
7.2 F and p-values for the ANOVA tests corresponding to the plots in Figure 7.4. Only time*cultivar interaction is shown corresponding to the factor of interest (longitudinal trend). Results with 'All' and Consensus citizen average (or max) across per-plant observations. 146
7.3 A simulated citizen-powered experiment. P-values corresponding to an ANOVA test randomizing the number of observations available per each plant at a specific time point. Process is repeated sampling from any of the observers (i.e. the sampling may contain a mix of experienced and non-experienced observers) or only from experienced (ExP) or non-experienced (i.e. NExP) ones. 150
7.4 Algorithmic leaf counting results obtained using the method in [7]. Four metrics are reported. We first compare between the algorithm and the 728 images in the training set (ie. how well the algorithm learns). Then we compare how well the algorithm predicts counts on a testing set of 130 images (also used in this study) comparing the algorithm with the counts of the annotator (that also was involved in deriving annotations for the training set). Lastly we compare the annotator (the data of which we used to train the algorithm and was not involved in this study) with the reference observer used throughout in this study. 154

8.1 We trained the leaf counting algorithm in [7] using A4 dataset only (top set of lines) and A4+Ax (bottom set of lines). Results obtained with 4-fold cross validation. Results for DiC and |DiC| are reported as *mean* (*std*). 171

9.1 Testing set results of PhenoDC trained on RGB images from the CVPPP 2017 dataset [6,12,16]. Evaluation metrics are detailed in Section 2.4. 187
9.2 Testing performance of *PhenoDC* on the multi-modal dataset [15]. We report results when the network is trained using only a single input and when also using all inputs. 189
9.3 Fine-tuning of the parameters of PhenoDC on Tobacco images [6] previously pre-trained with Arabidopsis plants A1, A2, and A4 [6,16]. We progressively increase the number of training images to find a suitable number of images required to create a meaningful model that can count Tobacco leaves. Below we report the results on the held-out testing set. 192
9.4 A similar process to that described in Table 9.3 but repeated for komatsuna plant leaf counting based on data available in [17]. The model has been trained on Arabidopsis as described in Table 9.3. Results shown refer to the testing set. 194

A.1 Most frequent activation functions used on artificial neural networks. 227

Acknowledgements

I express my sincere gratitude to the person who made everything written in this thesis possible: my Ph.D. supervisor Sotos Tsaftaris. He guided me throughout the last four years to develop myself as a researcher. I also want to thank all the colleagues from IMT Lucca and University of Edinburgh (UoE) that have supported me for so many years. Starting from Massimo Minervini, who collaborated with me during my first steps of my Ph.D. programme. İlkay Öksüz, the friend who gave me all the good advice to succeed in this quest. Vasilis Sevetlidis, my dearest buddy who I spent countless days, and shared many nice experiences with. Agis Chartsias and Tom Joyce, my friends from the UoE with whom coffee breaks are a *must-do*. All my other friends from IMT Lucca: Alessandro, Andreas, Anita, Cocca (Valentina), Giovanna (the other Cocca), Manas (my former roommate), Puya, Valerio, Viggy, Vihang. I also want to thank Paola from FZJ, who patiently stayed by my side while I was mourning for the loss of my laptop.

I would like to thank my co-authors: Hanno Scharr, Pierdomenico Perata, Peter Doerner, Hao Chen, Andrei Dobrescu, and Feng Chen.

I also thank my friends from the University of Catania (UniCT): Antonino and Nicoletta, Matteo, Marco and Davide. A special thanks goes to Giovanni, who allowed me to participate to several summer schools, giving me the opportunity to meet interesting people from all over the computer vision community.

Thanks a lot to my friends at the Alan Turing Institute, who helped to make my time in London a bit more bearable.

To my mom, who patiently raised me and made me the man I am today.

I also want to express my gratitude to my former high school teacher Maria Cristina Floreno, the person who taught me how beautiful and full of opportunities the university world is.

To you all, I say *grazie mille*.

Vita

January 13, 1988	Born, Catania, Italy
2006–2010	B.Sc. in Computer Science University of Catania Catania, Italy
2010 (6 months)	Erasmus Student Aarhus University Aarhus, Denmark
2010–2013	M.Sc. in Computer Science University of Catania Catania, Italy
2011 (3 months)	Visiting Scholar University of California, Los Angeles Los Angeles, USA
2012 (3 months)	Visiting Scholar Hanoi University Hanoi, Vietnam
2014–Ongoing	Ph.D. in Image Analysis PRIAn Research Unit IMT Institute for Advanced Studies Lucca, Italy
2015–2016	Erasmus+ Traineeship Institute for Digital Communications University of Edinburgh Edinburgh, United Kingdom
2016–2017	Enrichment Year The Alan Turing Institute London, United Kingdom
2017–Ongoing	Research Associate

Institute for Digital Communications
University of Edinburgh
Edinburgh, United Kingdom

Publications

[1] **Mario Valerio Giuffrida**, Sotirios A Tsaftaris, "Explicit Factorization of Rotations in Restricted Boltzmann Machines", Under review on IEEE Transactions of Image Processing, 2018.

[2] **Mario Valerio Giuffrida**, Peter Doerner, Sotirios A Tsaftaris, "Pheno-Deep Counter: a unified and versatile deep learning architecture for leaf counting", The Plant Journal, 2018. **PDF**

[3] Hao Chen, **Mario Valerio Giuffrida**, Peter Doerner, Sotirios A Tsaftaris, "Root Gap Correction with a Deep Inpainting Model", in Computer Vision Problems in Plant Phenotyping (CVPPP) workshop held in conjunction with BMVC, 2018 **PDF**.

[4] **Mario Valerio Giuffrida**, Feng Chen, Hanno Scharr, Sotirios A Tsaftaris, "Citizen crowds and experts: Observer variability in image-based plant phenotyping", The Plant Methods, 2018. **PDF**

[5] Agisilaos Chartsias, Thomas Joyce, **Mario Valerio Giuffrida**, Sotirios A. Tsaftaris, "Multimodal MR Synthesis via Modality-Invariant Latent Representation," IEEE Transaction on Medical Imaging, 2018. **PDF**

[6] **Mario Valerio Giuffrida**, Hanno Scharr, Sotirios A Tsaftaris, "ARIGAN: Synthetic Arabidopsis Plants using Generative Adversarial Network," in Computer Vision Problems in Plant Phenotyping (CVPPP) workshop held in conjunction with ICCV, 2017. **PDF**

[7] Andrei Dobrescu, **Mario Valerio Giuffrida**, Sotirios A Tsaftaris, "Leveraging multiple datasets for deep leaf counting," in Computer Vision Problems in Plant Phenotyping (CVPPP) workshop held in conjunction with ICCV, 2017. **PDF**

[8] Massimo Minervini, **Mario Valerio Giuffrida**, Pierdomenico Perata, Sotirios A. Tsaftaris, "Phenotiki: An open software and hardware platform for affordable and easy image-based phenotyping of rosette-shaped plants," The Plant Journal, 2017. **PDF**

[9] **Mario Valerio Giuffrida** and Sotirios A Tsaftaris, "Theta-RBM: Unfactored Gated Restricted Boltzmann Machine for Rotation-Invariant Representations," arXiv, 2016. **PDF**

[10] **Mario Valerio Giuffrida** and Sotirios A Tsaftaris, "Rotation-Invariant Restricted Boltzmann Machine Using Shared Gradient Filters," in International Conference on Artificial Neural Networks, 2016. **PDF**

[11] Vasileios Sevetlidis, **Mario Valerio Giuffrida**, Sotirios A Tsaftaris, "Whole Image Synthesis Using a Deep Encoder-Decoder Network," in SASHIMI Workshop held in conjunction with MICCAI, Springer, 2016. **PDF**

[12] **Mario Valerio Giuffrida**, Massimo Minervini, Sotirios A Tsaftaris, "Learning to Count Leaves in Rosette Plants," in Computer Vision Problems in Plant Phenotyping (CVPPP) workshop held in conjunction with BMVC, BMVA Press, 2015. **PDF**

[13] Massimo Minervini, **Mario Valerio Giuffrida**, Sotirios A Tsaftaris, "An interactive tool for semi-automated leaf annotation," in Proceedings of the Computer Vision Problems in Plant Phenotyping (CVPPP), BMVA Press, 2015. **PDF**

[14] Giovanni Farinella, **Mario Valerio Giuffrida**, Vincenzo Digiacomo, Sebastiano Battiato, "On Blind Source Camera Identification", Advanced Concepts for Intelligent Vision Systems (ACIVS) – LNCS, 2015. **PDF**

[15] **Mario Valerio Giuffrida**, Giovanni Farinella, Sebastiano Battiato, Mirko Guarnera, "Exploiting Time-Multiplexing Structured Light with Picoprojectors," in Proceedings of SPIE – Electronic Imaging, 20. **PDF**

Abstract

Plant phenotyping refers to the measurement of plant visual traits. In the past, the collection of such traits has been done manually by plant scientists, which is a tedious, error-prone, and time-consuming task. For this reason, image-based plant phenotyping is used to facilitate the measurement of plant traits with algorithms. However, the lack of robust software to extract reliable phenotyping traits from plant images has created a bottleneck.

Here, we will study the problem of estimating the total number of leaves in plant images. The leaf count is a sought-after plant trait, as it is related to the plant development stage, health, yield potential, and flowering time. Previously, leaf counting was determined using a per-leaf segmentation. The typical approaches for per-leaf segmentation are: (i) image processing to segment leaves, using assumptions and heuristics; or (ii) training a neural network. However, both approaches have drawbacks. Heuristics-based approaches use a set of rules based upon observations that can easily fail. Per-leaf segmentation via neural networks requires fine grained annotated datasets during training, which are hard to obtain. Alternatively, the estimation of the number of leaves in an image can be addressed as a direct regression problem. In this context, the learning of the algorithm is relaxed to the prediction of a single number (the leaf count) and the collection of labelled datasets is easy enough to be also performed by non-experts.

This thesis discusses the first machine learning algorithm for leaf counting for top-view rosette plants. This approach extracted patches from the log-polar representation of the image, allowing us to cancel out leaf rotation. These patches were then used to learn a visual dictionary, which was used to encode the image into a holistic descriptor. As a next step, we developed a shallow neural network to extract rotation-invariant features. Using this architecture, we could learn features to explicitly account for the radial arrangement of leaves in rosette plants. Although the results were promising, the

leaf counting with rotation-invariant features could not outperform the previous approach.

For this reason, we moved our attention to deep neural networks. However, it is widely known that deep architectures are *hungry* of data. Therefore, we addressed the problem of how to collect more labelled plant image datasets, using three approaches: (i) we developed an annotation tool to help experts to annotate images; (ii) we uploaded images in a crowdsourcing online platform, allowing citizen scientists to annotate them; (iii) we used a generative deep neural network to synthesise the images of plants with the leaf count. Lastly, we will show how a deep leaf counting network can be trained with data from different sources and modalities, showing promising results and reducing the performance gap between algorithm and human annotators.

Part I

Introduction and Background

1 Motivations

Plants are the main source of four essential components of our daily life: *food*, *feed*, *fibre*, *fuel* [27]. Available data show that from 1960 to 2014, the gap between the productivity of vegetable and meat products is one order of magnitude wide, as shown in Figure 1.1 (FAOSTAT data). The increase in food productivity is clearly related to the constant increase in world population (7.5 billion in 2017), which causes an increase in food production. However, the availability of arable land, which is vital for plant production, does not experience such increase. In fact, whilst the world population has had a 300% increase in the last 70 years (~ 10 billion people are expected by 2050), the amount of area allocated as arable land has only had an increment of 10% (cf. Figure 1.2). In order to support the worldwide food demand, resources devoted to agriculture have to be optimised. The FAO (Food and Agriculture Organisation) suggests developing novel plant phenotyping techniques to study plant growth under optimal conditions [28].

Plant phenotyping allows the comprehensive study of the plant development. Whilst the genetic code of plants has been studied extensively in the last decades, we still know little about the correlation between genetic traits (genomes) and morphological traits

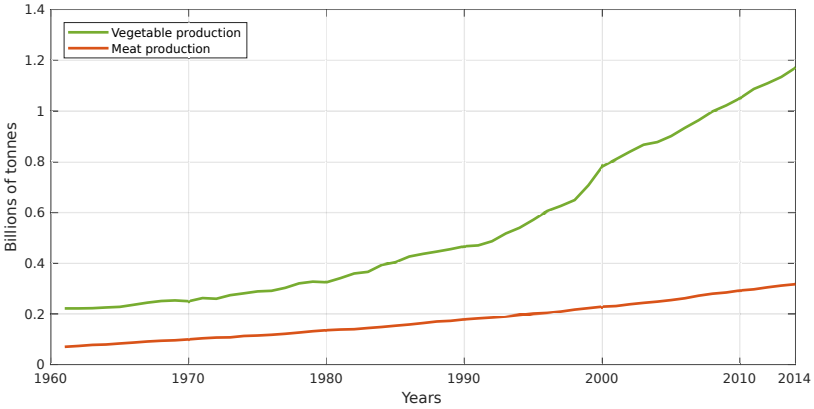
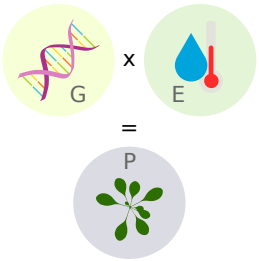


Figure 1.1: Vegetable (crop) vs. meat (livestock) productivity. Scales are expressed in billions of tonnes. Data from the Food and Agriculture Organization (FAO) accessed on December 2017 (FAOSTAT dataset).

(phenomes) [29]. Specifically, a plant trait naturally depends on the set of genes, but also on the environmental conditions under which the plant is grown. We can formalise this relation as follows [28,30]:



where the phenotyping trait P is given by the genome G and the (potentially) infinite environmental conditions E. The knowledge gap

We used a synthetic Arabidopsis plant taken from [31] to graphically represent the relationship $P = G \times E$.

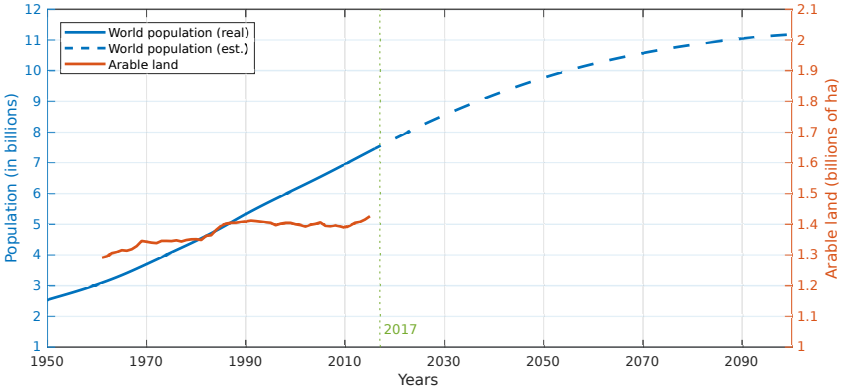


Figure 1.2: Population growth (current data and future predictions) vs. available arable land growth. Scales are expressed in billions of people (blue line) and billions of hectares (orange line). Data from the Food and Agriculture Organization (FAO) accessed on December 2017 (FAOSTAT dataset).

between genome and phenome has been considered a bottleneck in plant science [29].

In order to assess if a particular phenotyping observation is significant, plant biologists need to perform the same experiments on several groups of plants. For statistical purposes, one experiment requires a considerable number of plants per each group (or treatment) to be constantly monitored and analysed. Morphological trait measurements are usually acquired manually, which carries several drawbacks. First of all, an expert scientist has to dedicate a considerable amount of time to acquiring manual traits from plants. Secondly, it is a tiring and error-prone activity. As a result, there is a *race* to make this procedure faster and more reliable. For this reason, *image-based* plant phenotyping has started a new era in this research area. The acquisition of plant images allows the data extraction of multiple subjects at the same time.

Image analysis has made a great contribution to plant phenotyp-

ing. Many commercial and affordable solutions have been proposed in the last decade. Commercial plant phenotyping solutions are typically expensive [3,18] and cannot be financially supported by small research groups. On the other hand, affordable solutions can promote worldwide plant phenotyping, allowing labs from developing countries to contribute to the plant science community. However, the acquisition of plant images require reliable software to analyse and extract meaningful information from them. Therefore, the lack of algorithms that are able to provide precise phenotyping data is actually creating the *real* bottleneck [28]. It has been argued that machine learning and computer vision solutions can contribute greatly to developing reliable software for analysis of plant images [32,33].

Many solutions have been proposed to extract most of the geometric traits from RGB images of plants, such as perimeter, diameter, projected leaf area, and so forth [18,34–37], but also other image domains can be used for the same purpose, such as near-infrared [18,38] or fluorescence [34,38,39]. Typically, all these techniques to extract morphological plant traits use ad-hoc image processing algorithms to segment the plant. In fact, most of those traits can be extracted by analysing only the plant segmentation mask. These algorithms make extensive use of hand-designed heuristics that require specific acquisition conditions (e.g., HTPheno [40]), which are hard to generalise and thus easily fail.

The focus of this thesis is demonstrating the benefits of developing new machine learning approaches to extract a simple, yet extremely important biological trait: the *leaf count*. From a phenotyping point of view, the number of leaves in a plant is related to, for example, developmental stage [41], growth regulation [42,43], flowering time [44], and yield potential [45]. Because of its importance, leaf counting is a sought-after trait. However, the automatic identification of such quantity is hard, owing to several challenges that need to be addressed.

1.1 Challenges

Overall, leaf counting is a challenging task from the computer vision perspective. Even though it might seem an easy task from a human perspective, in [46], authors have shown that experts cannot achieve a consensus over leaf count. Therefore, algorithms are challenged by several factors, which we discuss below.

Scale variation. A leaf is a plant organ that generates from the stem. The range that the leaf size spans is incredibly high, starting from extremely small in the juvenile phase, until reaching the maximum size for a particular plant species. Typically, plant biologists count a leaf when it reaches a certain length (usually 1mm for rosette-shaped plants [2]).

Shape variation. Clearly, leaves differ across plant species. However, the shape of leaves also differs greatly within the same plant species. In Figure 1.3, some mutants of the plant Arabidopsis are shown, exhibiting a huge variability in shape.

Occlusions. Plants move due to nastic and tropic movements. The arrangement of leaves, when observed from a static point of view (e.g., top), can cause self-occlusions. For example, it can be seen in Figure 1.3 that very dense plants exhibit heavily occluded leaves. Clearly, if a leaf is completely occluded from the field of view, it cannot be counted. However, partially visible leaves can be counted, presenting a leaf counting algorithm with harder cases.

Instance isolation. Within the same plant, leaves of the same size/age appear similar to each other. If they do not overlap, it is very easy to distinguish between them. However, as previously said, leaves tend to heavily overlap. Therefore, distinguishing between two leaves located next to each other (and even one on top of the other) is a hard task. Isolating each individual leaf for counting

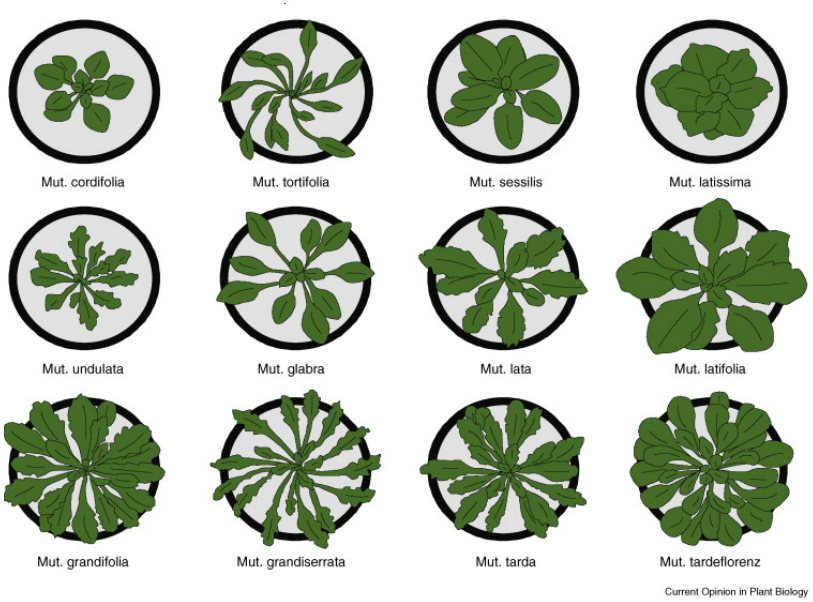


Figure 1.3: Some mutants of the plant Arabidopsis, showing high intra-species leaf shape variability. (Figure adapted from [1]).

is challenging, because leaves (within the same plant) may exhibit a similar shape, texture, and sometimes size as well. Furthermore, in the case of rosette-shaped plants (cf. Figure 1.3), juvenile leaves located at the centre of the plant are extremely hard to individually isolate and enumerate.

Insufficient annotated data. From a machine learning perspective, publicly annotated plant images are lacking. The current released datasets [6, 12, 15–17] have approximately $1,500$ annotated images in all, spanning different plant species and mutants. Machine learning techniques, especially deep learning, need diversified datasets to better generalise and reduce overfitting. Learning in such a restricted domain is challenging and requires constant monitoring.

Heterogeneous setups. The aforementioned plant datasets include images of plants taken using different acquisition setups. As an example, the *Arabidopsis* plants in [6] look very different from the *Arabidopsis* plants in [16], because different cameras have been used (e.g. different image quality) and light conditions are different. Therefore, machine learning algorithms not only have to accommodate for the leaf variability, but also for the acquisition setup variability, exhibiting changes in illumination and image quality.

1.2 Research Questions

In the previous section, we unfolded the major challenges arising from plant phenotyping, specifically for the leaf counting problem. Based on those, we detail below the research questions that will be addressed throughout this thesis. In addition, we will highlight disseminated work and scientific and engineering output, since parts of this thesis have been published or are under consideration for publication.

• Research Question 1 •

Can machine learning help in plant phenotyping?

It is widely known in the computer vision community that machine learning outperforms other approaches. In fact, we were the first to propose a leaf count algorithm on top-view images of rosette plants [7], using a machine learning approach. We learn a visual dictionary in an unsupervised fashion. This dictionary is used to encode plant images with a holistic descriptor. Then a non-linear regression model is trained to map plant images and total leaf count.

• Research Question 2 •

Can machine learning phenotype?

Since machine learning has only recently been employed in the plant community, it is necessary to show that it can produce reliable results. We answered this question in [3], where we showed no statistical significance of differences in leaf count amongst genotypes. In this work, we propose an improved version of the leaf count

algorithm in [7], which is also embedded in the *Phenotiki Analysis Software*.

• Research Question 3 •

Can we learn better and more compact features using rotation redundancies?

Top-view plant images exhibit leaves that, even though they might share similar shapes, appear at different rotations. This fact is even more evident for rosette-shaped plants, as shown in Figure 1.3. Given this prior knowledge, we address the question of whether it is possible to learn rotation-invariant features in order to learn more discriminative plant features, thus improving leaf count. We answer this question in [24, 47], where we proposed a rotation-invariant Restricted Boltzmann Machine (ERI-RBM). Our architecture needs the inference of the dominant orientation for each input image. We propose two methods to perform such an inference.

• Research Question 4 •

Can deep learning be employed in plant phenotyping?

Despite the fact that deep learning is an extremely powerful tool, it requires big labelled datasets to learn good models. We took up this challenge and we answered this question in [48], where we proposed a multi-modal deep neural network for leaf counting; and in [49], where we presented that a conditional generative adversarial network could be used to synthesise images of *Arabidopsis* plants with a given leaf count.

• Research Question 5 •

How can more labelled data be collected?

Lack of data in plant phenotyping is a problem. Typically, plant biologists tend to be reluctant to release plant images of their experiments, and even when they do, they lack proper annotations (cf. [18]). Therefore, given plant images, we address the challenge of annotating them effortlessly. In addition, when images are missing, we proposed a tool inspired by state-of-the-art technologies to generate synthetic data. These questions are addressed in [3, 46, 49, 50], where we proposed three different approaches to obtain more labelled plant data. Specifically, in [3, 50], we presented an annotation

tool to assist plant experts in the annotation process. In [46], we used an online crowdsourcing platform to obtain more labels from citizen scientists. Lastly, in [49], we used an adversarial network to generate synthetic plant data.

• Research Question 6 •

Can multi-modal learning perform a better leaf count?

Plant images can be acquired using RGB cameras, although this is not the only method. In fact, some approaches [18, 34, 38, 39] extract plant traits in different domains, such as near-infrared or fluorescence. Can these modalities improve leaf counting? We answer this question in [48], where we presented a deep learning architecture that improves the leaf count predictions by learning from multiple modalities.

1.3 Contributions

The contributions brought by this thesis in the field of image-based plant phenotyping are multi-fold. We have:

1. proposed the first machine learning, leaf counting algorithm [7];
2. developed software for plant-image analysis, bundling all the algorithms proposed by our research group [3];
3. proposed a neural network based on the RBM, which explicitly accounts for rotations [24, 47];
4. proposed the first deep architecture able to count leaves using different datasets at the same time to compensate for the lack of training data [51];
5. proposed a methodology that uses DCGAN (deep convolutional generative adversarial networks), which is able to generate synthetic plant images [49];
6. proposed the first deep architecture able to count leaves, using multi-modal data [48];

7. systematically and quantitatively assessed for the first time inter-observer variability across expert and non-expert leaf counting [46].

1.4 Thesis Structure

In this thesis, we tackle the problem of determining the total number of leaves from plant images, using machine learning algorithms. The content of this thesis is organised into five parts, containing 11 chapters. Specifically, the content of each chapter is summarised as follows:

- *Chapter 2* details the background knowledge for this thesis. Firstly, we briefly discuss the plant biology background, mostly specific to *Arabidopsis thaliana* (rosette plants). Secondly, we show the machine learning background, detailing in general neural networks. Then we offer a brief background in deep learning and GPU computation. Lastly, we detail the metrics used in this thesis to evaluate the leaf counting algorithms.
- *Chapter 3* discusses the literature for counting. Specifically, we categorise the current state-of-the-art approaches as *holistic* or *local*. The methods in the former category extract global features from images to infer the object count, whereas the latter ones extract local features from the image to obtain a localised count.
- *Chapter 4* shows the first leaf counting algorithm, using a holistic machine learning approach. This work was published during the second *Computer Vision Problems in Plant Phenotyping* (CVPPP) workshop in 2015, held in conjunction with the *British Machine Vision Conference* (BMVC). The proposed pipeline is: (i) log-polar transformation to remove the rotation nuisance from the rosette plants; (ii) patch extraction; (iii) visual dictionary learning; and (iv) feature encoding and

regression. This approach was the best performing algorithm in the *Leaf Counting Challenge* (LCC).

- *Chapter 5* discusses the explicit learning of rotation-invariant features. The leaf counting algorithm discussed in the previous chapter applies the log-polar transformation to remove the rotation variability. This approach has two drawbacks: (i) a foreground mask is necessary to estimate the centre of the plant; and (ii) the central part of the plant shows interpolation artefacts. Therefore, in this chapter we show a shallow neural network to learn rotation-invariant features. We show two variants of our *Explicit Rotation-Invariant Restricted Boltzmann Machine* (ERI-RBM).
- *Chapter 6* presents a graphic tool to assist experts for the semi-automatic annotation of plant images. We argue that deep learning can further help plant phenotyping. However, deep architecture needs large labelled datasets to train models with good generalisation capabilities. Therefore, we need to collect more labels from plant images. Later, our annotation tool was embedded into the *Phenotiki Analysis Software*, a program designed for plant biologists, bundling together several other algorithms developed by our research group, including the leaf counting approach outlined in *Chapter 4*.
- *Chapter 7* discusses the problem of the intra- and inter-observer variability in annotating plant images. For this reason, we started a Zooniverse project to allow citizen scientists to annotate plant images. Then, we studied the observer variability of the crowdsourced data and annotations collected from experts and non-experts, when they annotated the same data using the annotation tool outlined in Chapter 6.
- *Chapter 8* shows an alternative approach to obtaining labelled datasets. Using *Generative Adversarial Networks*, we learn the distribution of the plant images to generate synthetic images of annotated *Arabidopsis*.

- *Chapter 9* introduces the *Pheno-Deep Counter*, a deep neural network that is capable of learning the leaf count from multiple image modalities (e.g. visible light, near infrared, fluorescence). This architecture is versatile and flexible, being able to be adapted for any number of input modalities.
- *Chapters 10 and 11* summarise the thesis and detail the limitations of our work. Then, we discuss several possible future ventures to improve and expand the research work presented in this thesis.

2 Background

In this chapter, we discuss the plant and computer science background. Due to the nature of the leaf counting application, this document presents several biological, engineering, and mathematical terms that may not be known to a broad audience. Therefore, we will introduce notions of: (i) plant biology; (ii) machine learning (in general); (iii) deep learning; and (iv) the evaluation metrics to measure the performance of the leaf counting algorithms.

2.1 Plant Biology Background

In Chapter 1, we said that worldwide experiments in plant phenotyping would improve crop productivity. In order to perform and obtain fast results in experiments, it is desirable to adopt a plant species with a short life cycle and small size. For this reason, plant biologists use the *Arabidopsis thaliana* as the model plant for their experiments. In Figure 2.1 we display a summary of the life cycle characterising an *Arabidopsis thaliana* plant [2]:

- 1 week: the seed germinates;
- 4 weeks: the rosette grows and produces leaves;

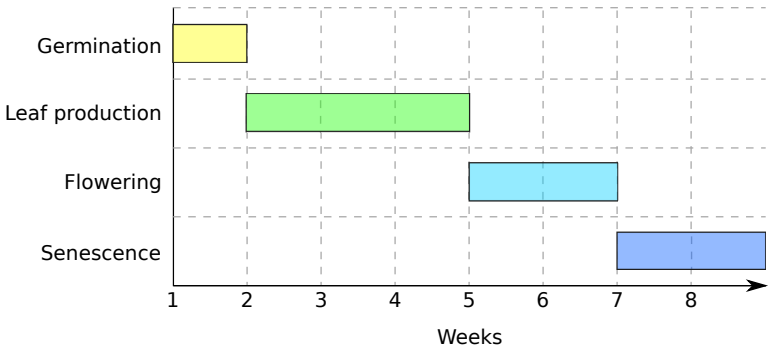


Figure 2.1: Schematic of the *Arabidopsis Thaliana* life cycle (data taken from [2]).

- 6 weeks: the plant flowers;
- 8 weeks: siliques ripen (e.g., they release the seeds) and the plant is ready to be harvested.

The other benefit of this model plant is the size. The wild type (e.g. the genotype naturally present in nature) Columbia (Col-0) develops rosettes of $\sim$ 10cm diameter [3]. This allows plant biologists to seed several plants in a tray, allowing a top-view camera to capture all the plants. In Figure 2.2(a), we display a tray containing 24 plants of *Arabidopsis thaliana* plants from five difference mutants (col-0, pgm, ctr1, and ein2.1; c.f. Figure 2.2(b)).[1] Figure 2.2(c) shows that a plant grows in approx. 5 weeks after sowing.

From the image analysis perspective, *Arabidopsis* plants have another important benefits. Morphologically, they are *rosette* plants, meaning that the leaves are circularly arranged around the centre of the plant. In addition, they do not expand much vertically, remaining mostly flat. This particular geometry allows easy and effective image acquisition from the top.

Leaf counting is a sought-after trait, because it gives an indication of the stage of a plant's development [2], together with other

[1] Further descriptions of these mutants can be found in [3]

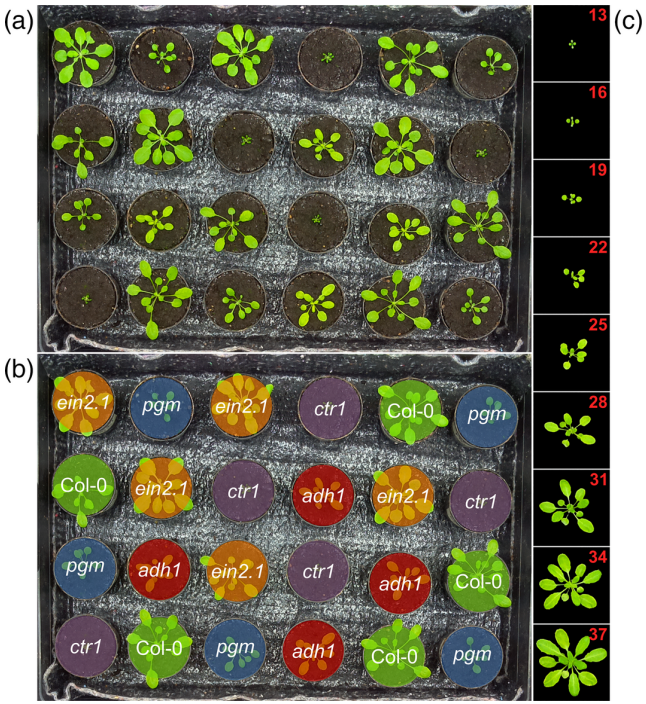


Figure 2.2: Example of a plant tray with 24 plants of *Arabidopsis Thaliana* (a) from different 5 ascensions (b). In (c), we show a time lapse of a plant subject development in 24 days (image adapted from [3]).

important information, such as the health conditions [52]. (Please refer to Chapter 1 for further details about the importance of leaf count). From the biological point of view, several structures compose a leaf. For the purposes of this thesis, we are interested in three parts, as shown in Figure 2.3. Specifically, the *lamina* (or blade) is the most important part of a leaf, as the plant mostly relies on this wide and flat part to accomplish photosynthesis [53]. The blade of the leaf is connected to the plant's stem via the *petiole*, which is also responsible for the leaf's movements [54]. At the far end of the

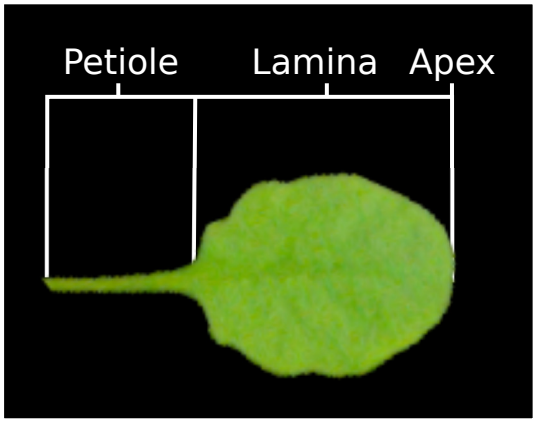


Figure 2.3: Anatomy of an Arabidopsis leaf.

blade, there is the leaf *apex* (or tip). This part of the blade is the landmark used to compute morphological plant traits, e.g. plant diameter (maximum distance between any two leaves in a rosette), and leaf length (maximum distance from stem to apex).

Even though it cannot be easily assessed by the naked eye over a short period of time, plants move in reaction to external stimuli. For instance, a leaf moving towards the sunlight is performing a *tropic* movement. *Nastic* movements are also in reaction to external stimulus, although plant response is non-directional. Such a movement can be the upward inclinations of leaves due to growth.

2.2 Machine Learning Background

Machine learning encompasses a set of techniques and algorithms that fit a mathematical model to a set of (labelled) data. Given a dataset $\{x_i, y_i\}$, $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, a machine learning algorithm finds a function φ such that $\varphi(x_i) = y_i, \forall i = 1, 2, \ldots, N$, where x_i is a data point, y_i is a target variable, and $\varphi(\cdot)$ is the task at hand. Therefore, we are interested in finding the best correlation between

data points and target variables. In general, if $\mathcal{Y}$ is a finite and discrete set, φ is called *classifier*. If $\mathcal{Y}$ is discrete or continuous range of values, then φ is called *regressor*. For sake of clarity, in this section we will assume φ as a classifier, although similar arguments hold for the regression case as well.

In computer vision, the set $\mathcal{X}$ contains images to be analysed. From the mathematical point of view, an image is represented as a 3-dimensional matrix $I(u, v, c)$, where u, v refers to the image coordinates, c refers to one of the image channels (e.g. red, green, and blue). In general, an image $x_i \in \mathcal{X}$ cannot be used in a classifier φ as it is, because raw images do not have enough discriminative power. Specifically, using an image I as a set of pixels is not enough to construct a function φ that outputs the correct target value y_i, for any $i = 1, 2, \ldots, N$. In order to reduce ambiguity and increase the discriminative power of the data, the images are transformed from the image space $\mathcal{X}$ to the feature space $\mathcal{F}$ using a function ψ that retains the most discriminative information within the image. In general, we can describe this process as follows:

$$x \xrightarrow[\text{feature extractor}]{\psi(x)} f \xrightarrow[\text{classifier}]{\varphi(f)} y \tag{2.1}$$

The task of feature extraction is a crucial problem in computer vision. In the past, machine learning was confined only in finding the best φ (e.g., classifier), where the feature extraction was done using *hand-crafted* descriptors. Specifically, a plethora of methods exploiting certain characteristics of an image (e.g., texture, presence of corners, edges, etc.) was used to describe images. (A recent review of these methods can be found in [55]). However, since the difficulty of extracting discriminative image features, machine learning has also been utilised to learn better ψ functions. Extensive efforts have been done in the past years, towards the quest of finding better image features for computer vision tasks. Examples of learned features are e.g. vocabulary learning [56], sparse coding [57], Gaussian mixture models [58], neural networks [59]. In particular, neural networks have proven to outperform hand-crafted features and other

feature learning approaches [60]. For this reason, deep learning has attracted attention in the last decade.[2]

2.3 Deep Learning Background

Neural networks have been proven to learn high discriminative image features for the task at hand. The first and most simple neural network is the perceptron [61], which solves the problem of binary classification. Specifically, this simple architecture multiplied the input data by a set of weights and the result of the classification was either '0' or '1', if the result of this operation was greater or less than a certain threshold (known as a *bias term*). Both these parameters (weights and threshold) are learnt during training. The set of weights and bias term is known as the *parameters set* (typically denoted as Θ). An extension to this model is the logistic regression [62], which solves the same problem using a non-linear function. In this case, the multiplication between input and weights is performed within the logistic function, which is bound in [0,1]. Since real data rarely exhibit a linear nature, the introduction of a non-linearity has shown better performance in neural networks.

Deep networks exploit the idea of the previous model and stack several layers, such that the output of one layer becomes the input of the next one. As the number of layers increases, the size of the parameters set increases as well. Therefore, the complexity of a deep network stems from the number of parameters to optimise. The current method used to train deep networks is the backpropagation algorithm [63], which enables an efficient way to optimise network parameters. Backpropagation allows neural networks to be more complicated and robust, stimulating the emergence of deep learning. For this reason, the functions in (2.1) are learnt jointly: a part of the network is committed to feature extraction, whereas the last part is the classification task. These architectures are usually known in the literature as *end-to-end* networks.

[2]Google Trends: `https://tinyurl.com/yczurp5z`

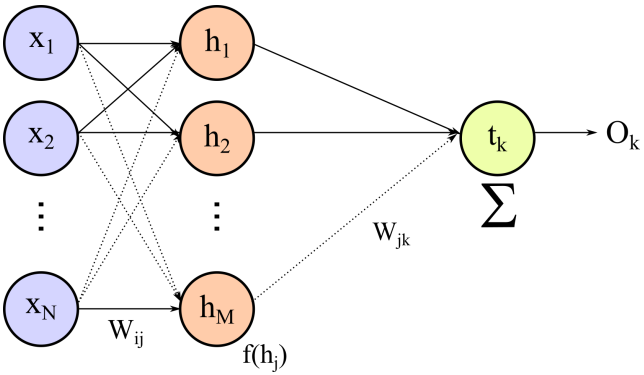


Figure 2.4: Example of neural network.

2.3.1 Under The Hood

To give an idea the way deep neural networks are trained, we show the example in Figure 2.4. An input vector $\mathbf{x} \in \mathcal{X}$ of size n is fed to the network. The hidden layer $\mathbf{h} \in \mathcal{F}$ is a vector of size m computing the following calculations:

$$\mathbf{h} = \psi(x) = \phi_1(W_1\mathbf{x} + \mathbf{b_1}), \tag{2.2}$$

where ψ is a feature extraction function, as in Equation (2.1). In this case, the operation performed by the feature extraction is a matrix multiplication between the input and a weight matrix $W_1 \in \mathbb{R}^{m \times n}$, and $b_1 \in \mathcal{F}$ is the bias term. The function ϕ_1 is called *activation function* (similar to the sigmoid in the logistic regression). The output of the network is (in this case) a scalar that is computed as follows:

$$\hat{y} = \varphi(\mathbf{h}) = \phi_2(W_2\mathbf{h} + b_2), \tag{2.3}$$

where $W_2 \in \mathbb{R}^{1 \times m}$ is another weight matrix for the output layer, with the scalar bias term $b_2 \in \mathbb{R}$. Similarly as in Equation (2.2), ϕ_2 is an activation function (it can be different than ϕ_1).

In Equations (2.2) and (2.3), the only known value during training is the input vector $\mathbf{x}$. Once the activation functions are fixed (c.f. Appendix A for further details), optimal values for the following parameters $\mathbf{\Theta} = \{W_1, \mathbf{b_1}, W_2, b_2\}$ have to be found. In order to do so, values in the parameters set $\mathbf{\Theta}$ are randomly initialised and backpropagation applied. This algorithm uses gradient descent to find optimal values for $\mathbf{\Theta}$, minimising a cost function. For the example in Figure 2.4, given a training set $X = \left\{(\mathbf{x^{(k)}}, y^{(k)})\right\}, k = 1, 2, \ldots, K$, with input vectors $\mathbf{x^{(k)}}$ and target value $y^{(k)}$, a valid loss function would minimise the distance between the prediction in Equation (2.3) and the target values in the training set. Such a loss function can be formalised as follows

$$\mathcal{J}(\mathbf{x}; \mathbf{\Theta}) = \frac{1}{2} \sum_{k=1}^{K} \left(y^{(k)} - \hat{y}^{(k)}\right)^2. \tag{2.4}$$

In Equation (2.4), the distance between predictions and target values is minimised for all the samples in the training set X. The optimisation of Equation (2.4) can be done via gradient descent, which updates each parameter $\theta \in \mathbf{\Theta}$, computing the derivative of the loss function w.r.t. θ:

$$\theta \leftarrow \theta - \alpha \frac{\partial \mathcal{J}}{\partial \theta} \tag{2.5}$$

where the computation of the derivative of Equation (2.4) w.r.t. θ involves the functions in Equations (2.2) and (2.3). Applying Equation (2.5) for a certain number of iterations (usually called epochs), optimal values for Θ will be found.

2.3.2 Convolutional Neural Network

Significant benefits were brought in deep learning, when the convolution operation was introduced as operation. In Equations (2.2) and (2.3), the information is processed with weight matrices using the dot product. This kind of layer (usually called *fully connected* or

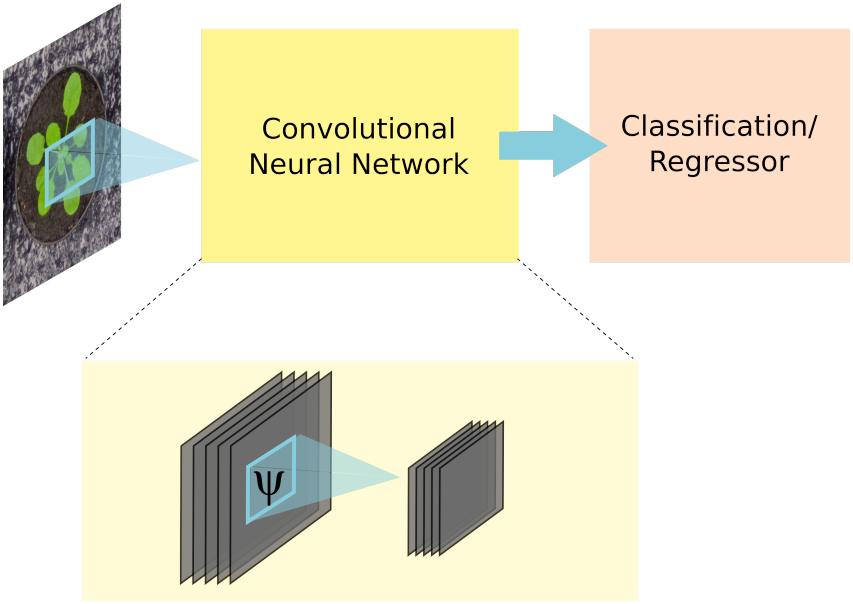


Figure 2.5: Schematic of a Convolutional Neural Network for a classification or regression task.

dense layer) cannot well accommodate 2-dimensional data, such as images. The convolution is a linear operation that can easily analyse two-dimensional data, limiting the number of parameters of the network. The 2-dimensional discrete convolution operation applied to an image I of size $w \times h$ with the kernel W of size $w' \times h'$, such that (without loss of generality) $w' \leq w$ and $h' \leq h$, is defined as

$$(I * W)(x, y) = \sum_{u=0}^{w'-1} \sum_{v=0}^{h'-1} I(x+u, v+u) W(u, v). \tag{2.6}$$

In Figure 2.6, we graphically show how the convolution operation works . Specifically, we apply it to a 5×5 image I with a kernel W of size 3×3. The weights of the kernel W are multiplied to the corresponding values of I and then the sum is computed.

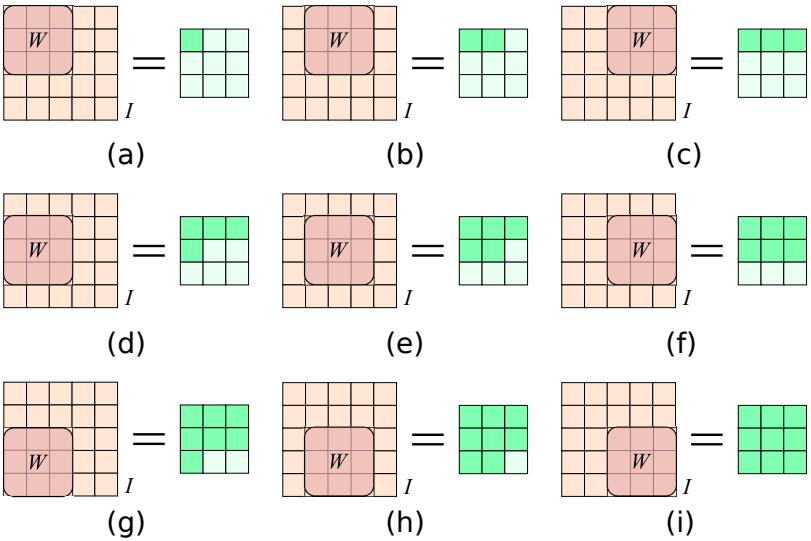


Figure 2.6: Convolution operation applied between an image I of size 5×5 and a kernel W of size 3×3.

For instance, Figure 2.6a shows the first step: all the weights of W are multiplied with the top-left values in I and the output is stored at top-left cell in the resulting matrix. As W shifts over I, the sum of products is repeated and the new value is located at its corresponding location in the resulting matrix.

Since Equation (2.6) is differentiable w.r.t. W, it can be easily plugged into a neural network, such that the backpropagation will find the optimal values for W. Network architectures with several convolutional layers are known as *Convolutional Neural Networks* [64] (shortened CNN or ConvNets), where an example is displayed in Figure 2.5. The success of ConvNets stems on their ability to extract high-level features from input images which are discriminative for the specific task (e.g. regression).

When convolutional layers are used, several hyper-parameters have to be set. Below, we report a short list of such hyper-parameters, which are relevant for this thesis.

- **Kernels size.** A convolutional layer typically applies the operation in (2.6) using different W. Therefore, the number K of kernels W_k has to be specified. Furthermore, the kernels have a finite dimension, which is another hyper-parameter. In the example in Figure 2.6, we show a 3×3 kernel. Other typical values are 5×5 and 7×7.

  Stride. In Figure 2.6, we displayed an example where the kernel W moves over the image I, covering all the locations. However, sometimes it might not be necessary to go entirely over the input image. Therefore, W can also move by steps (or strides) larger than 1. For example, if we applied the convolution in Figure 2.6 with a stride equal to 2, the output image would have been of size 2×2. For this reason, strides are typically used to downsample the resulting image.

- **Padding.** As highlighted in Figure 2.6, the resulting image of the convolution operation has a smaller size than the input. Specifically, the size of the output is $(w - w' + 1) \times (h - h' + 1)$. Sometimes, it might be necessary to have the result of the convolution of the same size as the input. To do so, padding pixels of half of the size of the kernel are added around I. Typically, these new values are 0, such that they will not have any undesired effect during the convolution.

2.3.3 Computational Demand

Deep neural networks are obtained by stacking several layers, such as convolutional and fully-connected, on top of each other. However, the learning of deep networks requires time. As an example, it is not rare to deal with deep networks with 50 layers and millions of parameters to optimise. If executed in a normal computer, the training would take weeks. For this reason, deep learning walks hand in hand with GPUs. Commonly referred to as video cards, a GPU is designed to render 3D scenarios at least 25 frames per second. In order to satisfy such a demand, a GPU is able to perform

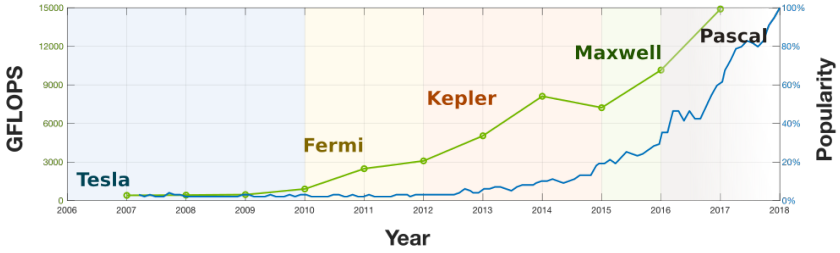


Figure 2.7: Evolution over time of Nvidia GPUs supporting CUDA vs the popularity of deep learning (Google Trend; *cf* Section 2.2).[3] (GFLOPS = Giga floating-point operations per second.) Shaded areas display the Nvidia GPUs' microarchitecture evolution. The latest microarchitecture (Volta) is not displayed as no benchmark data are available at this time.

millions of operations per second, thanks to the *parallel computing*. Since most of the operations of a neural network can be parallelised, a GPU can be used to train a neural network.

Since the introduction of CUDA in 2007, which allows programmers to run code on a GPU, deep learning has rapidly evolved. In Figure 2.7, we plot the computational power of the GPUs in GFLOPS (Giga floating-point operations per second) over time. It can be easily observed that the computational power of GPUs has increased in two orders of magnitude in the last 10 years. Also, the availability of large-scale annotated datasets has played an important role in the evolution of deep learning. For instance, ImageNet [65] is a constantly growing image dataset containing $\sim 14M$ labelled images. The training of such a quantity of data requires a lot of computational power, justifying the necessity of GPUs in deep learning and computer vision.

[3]Data source: `https://en.wikipedia.org/wiki/List_of_Nvidia_graphics_processing_units`, accessed April 2018.

2.4 Evaluation Measures

To evaluate the performance of the algorithms discussed in this thesis for the leaf count, following evaluation measures are employed:

- *Difference in count (DiC):* $\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)$;
- *Absolute Difference in count (|DiC|):* $\frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$;
- *Mean squared error (MSE):* $\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$;
- *Percentage agreement (%):* $\frac{100}{N}\sum_{i=1}^{N}\mathbf{1}\{y_i = \hat{y}_i\}$;
- *Coefficient of determination (R^2):* degree of the goodness of fitness between algorithm predictions and ground-truth [66];

where $\hat{y}_i$ is the algorithmic prediction for the input (image) x_i, and y_i is the ground-truth. The function $\mathbf{1}\{\cdot\}$ is the indicator function, which returns 1 when the predicate inside the braces is true, 0 otherwise. In fact, the *Percent Agreement* (%) indicates in how many cases the algorithmic estimation agrees with ground-truth. For this measure, values close to 100 indicates high performance. For the remaining evaluation measures, values close to 0 indicates better performance of the predictor.

For the per-leaf segmentation, a different evaluation criterion is employed: the *Symmetric Best Dice* [67], which is defined as:

$$\mathrm{SBD}(\hat{L}, L) = \min\left\{\mathrm{BD}\left(\hat{L}, L\right), \mathrm{BD}\left(L, \hat{L}\right)\right\}, \tag{2.7}$$

where L is the ground-truth per-leaf segmentation, $\hat{L}$ is the algorithmic result, and the *Best Dice* (BD) is defined as:

$$\mathrm{BD}(L^a, L^b) = \frac{1}{M}\sum_{i=1}^{M}\max_{1\leq j\leq N}\frac{2\left|L_i^a \cap L_j^b\right|}{|L_i^a| + \left|L_j^b\right|}, \tag{2.8}$$

given that the function $|\cdot|$ denotes the leaf area computed as the number of pixels, L_i^a is the *i*-th leaf in L^a (similar is L_j^b).

3

Prior Art

Counting is a computer vision task where, given an image as input, the number of specific objects is estimated. In the literature, there are two broad categories of counting algorithms, described as follows:

- *holistic approaches:* global features from images are extracted to obtain the total count;
- *local approaches:* image features are extracted in local regions of the images to obtain the count in that area. Total count is obtained aggregating local estimations;
- *hybrid approaches:* both local and global features are extracted to determine local (and global) estimations in a scene.

In addition, we dedicate an entire section to reviewing the leaf counting literature. Specifically, we discuss a particular type of local approach that counts by performing finely grained leaf segmentation. Then we also discuss holistic approaches that are based on regression models.

3.1 Overview

The estimation of the number of objects in a scene is very important in different contexts. From a security perspective, many algorithms have been developed to count people and pedestrians from the CCTV cameras [5,19,68–70]. Similarly, counting has been used for traffic analysis [71] and car counting [72,73]. In biology, counting algorithms have been employed on the problem of cell [73–75] or animal counting [76], such as penguin [77], mosquitoes [78], as well as for the leaf counting in plant images [7,20,48,51,79–82].Therefore, the literature on counting objects in an image is broad.

Intuitively, counting can be obtained by detecting or segmenting the objects [76,81,82]. However these approaches require finely grained annotations to be trained successfully. The collection of such datasets is not easy and it is time-consuming. So, if we are interested in determining *only* the number of objects in an image, this problem can be addressed as a regression task. In this case, the algorithm will output the total number of objects, rather than provide a precise annotation. In this case, not only does the task become easier to be solved, but also the data collection can be performed by non-experts. As an example, in [46,77], the authors employed the Zooniverse online platform (`https://www.zooniverse.org`) to collect annotations using non-expert citizen scientists.

In the next sections, we show the current state-of-the-art approaches on counting, addressing it as a direct regression problem.

3.2 Holistic Approaches

In [68], a global regression algorithm is presented to count the number of people in a video sequence. The foreground is extracted from the CCTV video, using a mixture of Gaussian model for the background and foreground segmentation [83]. This operation allowed the area of interest (called blobs), where pedestrians are located, to be obtained. From the detected area, two features are

extracted: histogram of orientations and histogram of blob size (in pixels). This method takes into account the scale variation caused by people's distance from the camera. To predict the number of people in the current frame, a neural network is trained.

In [69], the number of pedestrians in the UCSD dataset [4] is counted. The authors use background subtraction to identify and isolate blobs containing pedestrians. Then, the following image features are extracted: (i) number of pixels occupied by pedestrians, (ii) perimeter of the segmentation mask (using Canny edge detector [84]), (iii) edge responses in segmented blobs, (iv) *Minkowski* fractal dimension [85], (v) perimeter and area ratio, and (vi) statistical landscape features (SLF) [86]. Predictions are made by training a *semi-supervised elastic net*. While the elastic net puts together the benefits of ridge and lasso regression in a single optimisation framework, the authors proposed adding unlabelled data as a regularisation term. In order to exploit unlabelled data without biasing the regression model, the authors impose a constraint discouraging a sudden change of the pedestrian count between two neighbouring frames. This can be formalised with the following object function:

$$\hat{w} = \underset{w}{\operatorname{argmin}} \|Xw - Y\|^2 + \lambda_1 \|w\|_2^2 + \lambda_2 \|w\|_1 + \lambda_3 \sum_{(i,j)\in\Omega} (x_i w - x_j w)^2, \tag{3.1}$$

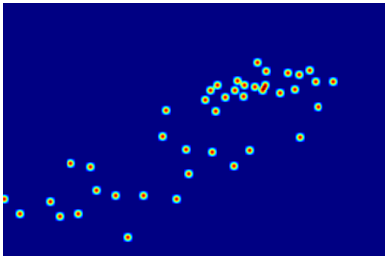
where Y is the vector containing the labels of the known frames (e.g., the number of pedestrians), X is the matrix with the global features for each image, $\|\cdot\|_n$ is the ℓ^n-norm, Ω is the set of neighbour frames, $\lambda_{1,2,3}$ are tuning parameters. The first part of the cost function is the elastic net definition [87] with ℓ^1 and ℓ^2 regularisation terms, whereas the last term penalises sudden changes in prediction between neighbouring frames.

In [19], the authors use a Bayesian regressor to infer the number of people in the UCSD video sequences, extracting features from the segmented area of the frames, via mixtures of dynamic textures [4]. Patches from the video sequences are modelled as samples

Table 3.1: Features extracted in [19].

Segment features		
Area	# of pixels in the segment	1
Perimeter	# of pixels in the segment's perimeter	1
Perimeter edge orientation	6-bin histogram of orientations	6
Perimeter-area ratio	Perimeter/Area	1
Blob count	# of connected component with at least 10 pixels	1
Internal edge features		
Edge length	# of pixels in a segment	1
Edge orientation	6-bin histogram of edge orientations	6
Minkowski dimension	fractal dimension of internal edges	1
Texture Features		
Homogeneity	texture smoothness: $g_\theta = \sum_{i,j} \frac{p(i,j\|\theta)}{1+\|i-j\|}$	4
Energy	sum-squared energy: $e_\theta = \sum_{i,j} p(i,j\|\theta)^2$	4
Entropy	randomness: $h_\theta = \sum_{i,j} p(i,j\|\theta) \log p(i,j\|\theta)$	4
	Total:	30

(a) Pedestrian annotated by placing a dot on them.



(b) Density ground truth obtained as a mixture of Gaussians.

Figure 3.1: Example of dot annotations on a frame taken from UCSD dataset [4].

drawn from a mixture model distribution, learned via expectation-maximisation (EM). Patches are assigned to the segment associated with the mixture component with the highest probability. The list of all features extracted is summarised in Table 3.1. Texture features are computed using the grey-level co-occurrence matrix [88] (GLCM), where images are quantised into eight grey levels and the joint distribution of neighbouring pixels $p(i, j|\theta)$ for orientations $\theta = \{0°, 45°, 90°, 135°\}$ is computed.

Deep learning has contributed to holistic counting methods. In fact, in [70], a deep regressor network is presented to predict the number of people in highly crowded scenes. Their architecture is a simple network with six convolutional layers, followed by two fully-connected layers. The last node outputs the count in the image.

3.3 Local Approaches

Local approaches for counting estimate the number of objects of interest in local areas of the image. Instead of extracting global features, predictions are made at local scale preserving the spatial information, which is usually not provided by holistic approaches. However, compared to global approaches, local methods require

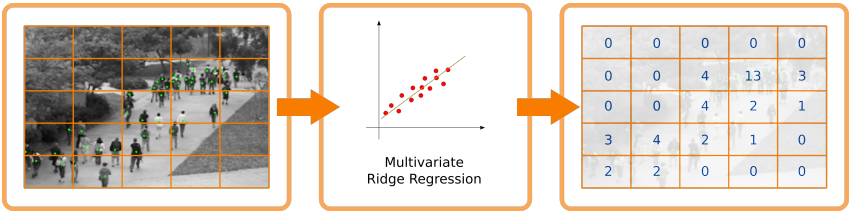


Figure 3.2: Counting pipeline of the method in [5]: (a) features are extracted in K cells, (b) features are used to train a multivariate ridge regressor, which provides (c) predictions simultaneously for all the cells.

some additional information in the data to include spatial information. Often, objects of interest are annotated by placing a dot where each one is located [74].

In [5], the authors present a local approach based on a *multivariate ridge regression*. Each training image is divided into a grid of K non-overlapping cells, where image features are extracted. Local features (refer to Table 3.1) are concatenated into a vector, which constructs a global representation of the image. The proposed multivariate ridge regression predicts the number of people for each cell, rather than providing a global count, preserving the spatial information related to the position of the people in the scene. A representation of the pipeline devised for this method is shown in Figure 3.2. This method benefits from the employed regression framework, since it has a closed-form solution and no iteration procedure is performed during training, although the solution of the linear equation coming from the closed-form solution might be computationally demanding as the number of data points increase.

An approach that has recently had a huge impact is outlined in [74]. It is used to count pedestrians in the UCSD dataset [4], as well as cells in synthetic images [89]. Specifically, the authors approximate the local density of pedestrians as the sum of Gaussian functions. Local counts are achieved by integrating (in the mathematical sense) a region of interest. The algorithm optimises the

Maximum Excess of SubArrays (MESA) function, which is a robust and computationally loss measuring the mismatch between the ground-truth and the estimated densities. Dense SIFT [90] descriptors are used as local image features. An extension of this method is in [73], where the authors present an interactive counting algorithm based on the ridge regression (similar to [5]).

Similarly, in [75], the authors propose a density estimation method based on *Random Forest* [91]. To train the regressor, the following patch-level image features are extracted: (i) pixel intensities (RGB colours), (ii) Laplacian of Gaussian, (iii) Gaussian gradient magnitude, and (iv) eigenvalues of the structure tensors. Random Forest is an ensemble of decision trees, that are trained to randomly select a set of features to split the image space. Final predictions are obtained by averaging all the single outputs provided by each decision tree in the forest.

Local density estimation has also been tackled via deep learning. In [77], the authors propose a multi-task convolutional neural network, based on [92], to count a new dataset of penguins.[1] Specifically, they collected dot annotations using *Zooniverse*, a citizen science web interface that allows millions of volunteers to support researchers by annotation the data. Many users annotated thousands of images multiple times, providing several annotations for each subject. Their network makes use of this information to predict:

- *foreground/background segmentation:* multiple annotations of the same penguin are used to generate blobs;
- *density estimation:* Gaussian functions are applied using a variable size computed by the dispersion of the annotations;
- *inter-observer agreement:* measure of uncertainty amongst the crowd-sourced annotations, using the variance of annotations as ground truth information.

[1] Dataset freely available at the following URL: `http://www.robots.ox.ac.uk/~vgg/research/penguins/`

Recently, the Count-ception network was proposed [93]. Based on an inception network [94], Count-ception is trained on density maps [73–75, 77]. Specifically, The network predicts the count on $r \times r$ patches. Due to the convolutional nature of the network, each pixel will have multiple predictions and, in order to overage out errors, the mean value per pixel is taken as actual count.

3.4 Hybrid Approaches

Deep neural networks have proved to be extremely powerful in many areas. In fact, as discussed before, many of the latest holistic and local approaches do employ deep learning. Due to its power, some deep learning methods have been proposed recently to solve the counting problem locally and globally simultaneously.

In [95], the authors propose a convolutional neural network (an architecture similar to [70], but with fewer convolutional layers), which is iteratively optimised to predict a density map and a global count. They apply their network to the problem of pedestrian counting, proposing a new dataset (*WorldExpo'10*). Differently from [74, 75], they approximate each person with two normalised Gaussians (e.g., they sum to 1): $\mathcal{N}_h$ is the head part, $\mathcal{N}_b$ is a bivariate Gaussian distribution for the body part. The learning is done iteratively between the two loss functions that minimise the ℓ_2 loss for the density estimation and global count. Similarly, in [96], the authors propose an end-to-end architecture that learns from local and global counts on crowded scenes.

In [97], the authors propose the *Contextual Map CNN* to predict the density map of a crowded scene. Specifically, the network has three branches that analyse the input images at different scales. The *Global Context Estimator* is a sub-network based on VGG-16 [98], which processes the entire input image. The *Local Context Estimator* is a sub-network that takes patches of 64×64 to extract local features from the input image. Inspired by [99], this branch helps to extract meaningful high-level features for the final task of estimating the

density map. All the features extracted by these sub-networks are concatenated to produce an output density map. Optimisation is performed using two losses: (i) minimises the ℓ_2 distance between ground truth and estimated densities; (ii) minimises the goodness of the estimated density w.r.t. the input image (adversarial loss [100]).

3.5 Leaf Counting Methods

Leaf counting is a particular instance of object counting from images and, as discussed in 1.1, is a challenging task [28]. Therefore, since leaf counting is the major focus of this thesis, we detail the state-of-the-art approaches of this topic separately.

Density-based approaches [73–75, 77, 95, 97] do not perform well in this context, due to the large variability in the appearance and scale of leaves, as new leaves appear very small and grow over time. In fact, the size of leaves should not affect the actual count. Furthermore, inter- or intra-species variability plays a big role. Clearly, leaves from two different plant species of genotypes (c.f. Figure 1.3) appear different. However, the difference in appearance between juvenile and grown leaves is also large (c.f. Figure 2.2c).

For these reasons, specifically designed approaches have been proposed for leaf counting, which can be grouped into two categories:

- *counting by segmentation*: leaves are individually segmented and the number of instances provides the total count;
- *counting by regression*: holistic methods, similar to the ones described in Section 3.2.

3.5.1 Counting by Segmentation

Leaf segmentation is a computer vision task that aims to delineate each leaf individually. In Figure 3.3, we display an example: given an RGB image (c.f., Figure 3.3a), the task is to obtain a finely-grained

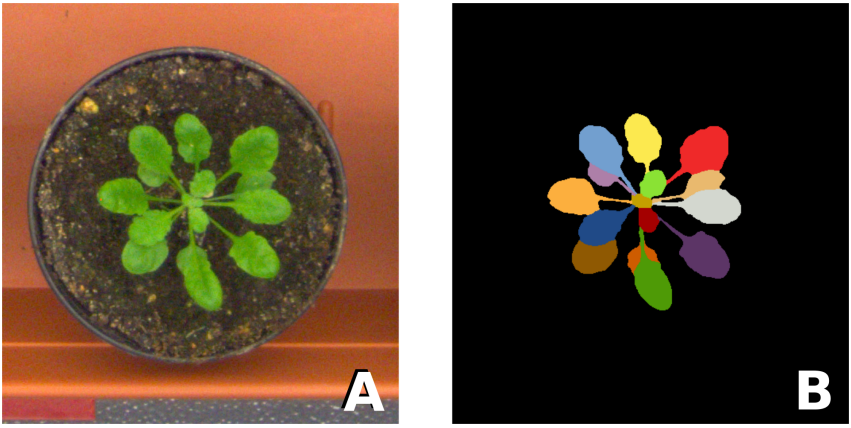


Figure 3.3: Sample of RGB Arabidopsis Thaliana Col-0 image taken from [6] (a), with its corresponding per-leaf segmentation mask (b).

per-leaf segmentation of the plant, as shown in Figure 3.3b. In the computer vision community, this is seen as a multi-instance segmentation problem [67], which is a hard task to accomplish due to the variability of plants. A first attempt to achieve this goal in (semi-)controlled conditions was done in [101], but their method assumes a destructive imaging protocol, which limits its application. In [67], four methods for leaf segmentation are proposed. All of them tackle this problem by exploiting the geometric characteristic of the rosette, whilst the utilisation of machine learning techniques is still limited. Later, the approaches in [81,82] improved the results of leaf segmentation, and thus leaf counting, by employing deep recurrent neural networks. Although their results outperformed the state-of-the-art approaches, these methods suffer from a major drawback. In order to get an image similar to the one shown in Figure 3.3b, finely grained segmented images must be paired with RGB images for training.

The manual process of leaf annotation is considered a tedious and time-consuming task: it is estimated that it requires on average approximately 20 minutes for an expert annotator to delineate

each leaf in a plant image [50]. Even though this number can be drastically reduced to a couple of minutes, by using specifically designed annotation tools [3, 50], it still requires expertise to obtain a precise annotation to be used as ground-truth for machine learning approaches. Thus, counting by segmentation needs annotated data, which are hard to obtain. In order to obtain the leaf count, this problem can be relaxed. Instead of using finely-grained per-leaf segmentation masks for training, we consider learning a model that, taking as input an RGB image of a plant, predicts the total leaf count, regardless of their spatial location. This introduces a different class of counting methods based on regression models.

3.5.2 Counting by Regression

The first batch of approaches of this kind includes [7] (detailed in Chapter 4), which employed an SVM regression model trained on holistic image descriptors, and [20], which used geometric features and compared them with different regressors provided on the machine learning software suite WEKA [102]. Recently, deep neural networks have also been employed for leaf counting. In [80], the authors propose a method to count the number of leaves, classify mutants, and estimate plant's age. They show outstanding results on the leaf counting problem, but their work suffers from a major issue: they tailored their neural network architecture for each dataset and task. Differently, in [18], the authors show that deep learning can benefit from data agglomeration from images acquired from different setups, without the need to adapt a network to a specific dataset. This approach obtained successful results in the *Computer Vision Problems in Plant Phenotyping* workshop (CVPPP 2017), held in conjunction with the ICCV conference. Another approach using deep learning for leaf counting is [79], where the authors proposed two networks working together to obtain the leaf count. Apart from the complexity of the model, this approach does not generalise well (e.g. errors in counting tobacco leaves were higher than with simpler machine learning approaches).

Part II

A Shallow Model for Leaf Counting

4

A Shallow Leaf Counting Direct Regression Model

In this chapter, we propose a learning-based approach for counting leaves in rosette-shaped plants. We relate image-based descriptors learned in an unsupervised fashion to leaf counts using a supervised regression model. To take advantage of the circular and coplanar arrangement of leaves and also to introduce scale and rotation invariance, we learn features in a log-polar representation. Image patches extracted in this log-polar domain are provided to K-means, which builds a codebook in an unsupervised manner. Feature codes are obtained by projecting patches on the codebook using the triangle encoding, introducing both sparsity and specifically designed representation. A global, per-plant image descriptor is obtained by pooling local features in specific regions of the image. Finally, we provide the global descriptors with SVR to estimate the

This chapter is based on:

- M. V. Giuffrida, M. Minervini and S. A. Tsaftaris. "Learning to Count Leaves in Rosette Plants", *Proceedings of the Computer Vision Problems in Plant Phenotyping (CVPPP)*, pages 1.1-1.13. BMVA Press, 2015.

Figure 4.1: Some examples of the A1, A2, and A3 images from the CVPPP 2015 dataset.

number of leaves in a plant.

We evaluate our method on datasets of the *Leaf Counting Challenge* (LCC), containing images of Arabidopsis and tobacco plants. Experimental results show that on average we reduce absolute counting error by 40% w.r.t. the winner of the 2014 edition of the challenge –a counting via segmentation method. When compared to state-of-the-art density-based approaches to counting, on Arabidopsis image data ∼75% less counting errors are observed. Our findings suggest that it is possible to treat leaf counting as a regression problem, requiring as input only the total leaf count per training image.

4.1 Proposed Approach

Here a global regression method to count leaves in rosette plants is described, hereafter referred to as *General Leaf Counting* (GLC). The algorithm takes as input greyscale images $I_j, \forall j = 1, \ldots, N$, showing a top-view of individual rosette plants (cf. Figure 4.1). As Figure 4.2 illustrates, the first step exploits the circular arrangement of leaves by converting the image into the log-polar domain. Then a suitable feature representation is learned from the data by training a dictionary in an unsupervised fashion on image patches

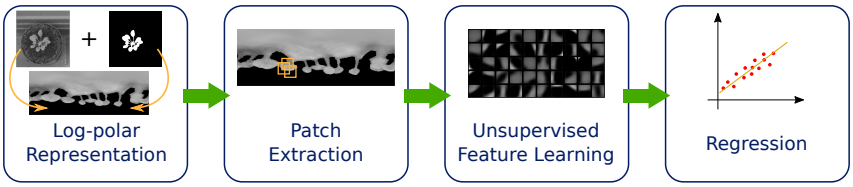


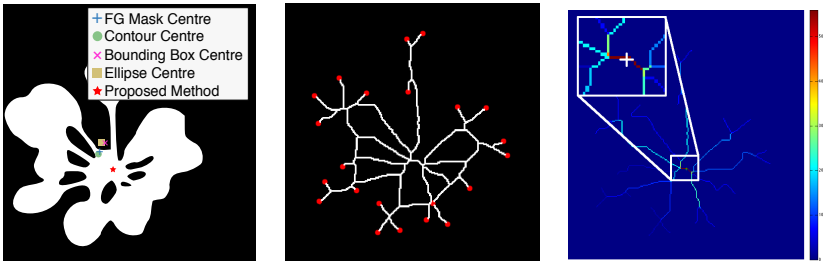
Figure 4.2: Flowchart describing the leaf counting method we proposed for the CVPPP Workshop 2015.

extracted from informative regions. A local descriptor for each patch is computed using the learned dictionary, employing the *triangle encoding* [103]. By max-pooling, such feature vectors are combined to obtain a global image descriptor, used to train a regression framework to predict the number of leaves. Each step is detailed in the following sections.

4.1.1 Log-polar Representation

Rosette plants are characterised by a radial arrangement of leaves around the centre of the plant (i.e., the stem). In order to exploit this structure, the input image I (the subscript j is omitted for brevity) is converted into the log-polar domain, obtaining a new image denoted as $\tilde{I}$. This conversion not only orients leaves w.r.t. the plant centre to appear parallel, but also ensures the same sampling and final dimensions of $\tilde{I}$ for any plant size, accounting for the problem of extracting good descriptors in the presence of large size variability within a training set.

The log-polar transformation needs the plant centre as input. Since finding the centre of a mass for the foreground mask is unreliable in a complex object (see also Figure 4.3a), here the position of the centre of the plant is estimated from the skeleton of the segmentation mask. From the skeleton, the endpoints (plotted in red in Figure 4.3b) are detected and the shortest paths along the skeleton connecting each endpoint to all other ones are computed. Aggregating all shortest paths, the segment that is traversed more frequently



(a) Plant mask (b) Skeleton (c) Centre detection

Figure 4.3: Centre detection in a complex object. Shown are: (a) plant mask available from expert annotation (together with classical calculations of a centre and proposed); (b) skeleton obtained from (a) and endpoints as red dots; and (c) most traversed segment, with detected centre marked with a white cross.

is detected. Therefore, the centre of the region containing the most traversed segment is set as the new origin (x_0, y_0) (Figure 4.3c). In Figure 4.3a, different approaches to determine the plant centre of mass are compared: (i) segmentation mask centre [104]; (ii) centre of the contour; (iii) centre of the smallest fitting bounding box; and (iv) the smallest fitting ellipse. It can be observer that our approach to find the centre of complex objects is the best among the others, since it take into the account the elaborated structure of the plant. (Quantitative results are reported in Table 4.2.)

When the centre of the plant is estimated, the input image I is converted into the log-polar domain. The conversion is computed by sampling with increments of 1° in the angular coordinate θ, thus the transformed image $\tilde{I}$ is 360 pixels wide, while the radius is adaptively chosen by computing the distance between a centre of the plant and the farthest point in the segmentation mask. In order to facilitate the next step of the algorithm, fixed zero-padding is added at the bottom-most part of the image. The result of the log-polar transformation can be seen in the second and forth column of the panel shown in Figure 4.4, where they correspond to the

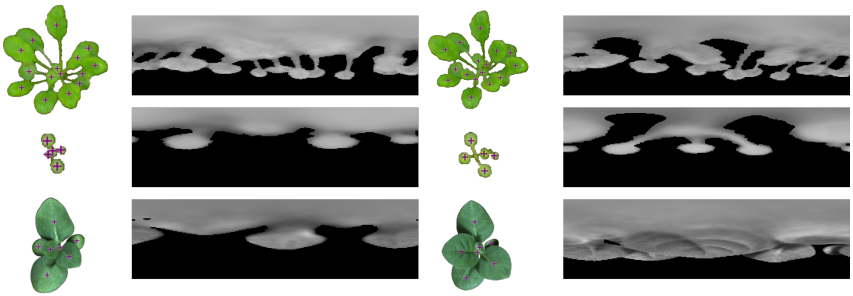


Figure 4.4: Example images (background was removed) of Arabidopsis taken from the A1 (top), and A2 (middle), datasets respectively, and tobacco taken from A3 (bottom). First and third columns show the same plant (with leaf centre annotations in purple) few days after. Second and fourth columns show the corresponding log-polar representations.

conversion of the plants placed to their left side.

4.1.2 Patch Extraction

To learn a dictionary, instead of extracting densely all possible patches from $\tilde{I}$, patch extraction is focused on regions that are most informative from a leaf counting perspective. Regions are identified with the *FG/BG ratio curve*, i.e., the ratio between the number of foreground (FG) pixels and the number of background (BG) pixels (Figure 4.5). Using a sliding window as high as $\tilde{I}$ and of fixed width W, $\tilde{I}$ is scanned along the θ-axis and the FG/BG ratio is computed. The ratio between foreground and background pixels will have high value wherever plant pixels are dominant, even when leaves are overlapping. Local maxima are detected in the curve in Figure 4.5, and use the corresponding (column) locations to define in $\tilde{I}$ regions of interest of width W' centred on the maxima. From these regions (which may also overlap), $S \times S$ sized patches are densely extracted, discarding duplicates or patches falling entirely within background. The patches are then normalised by the ℓ^2 norm to reduce photo-

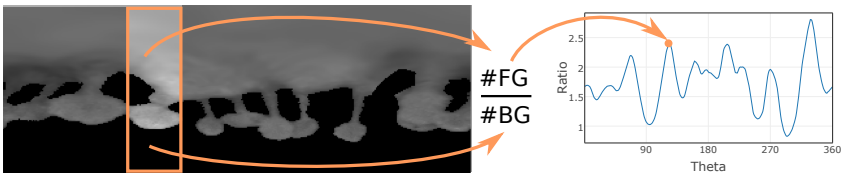


Figure 4.5: FG/BG ratio: a sliding window moves rightwards to compute the ratio between the number of foreground and background (black) pixels within it. Observe that we have local maxima even when leaves overlap.

metric variability. Patches extracted from a log-polar representation $\tilde{I}$ are denoted as vectors p_i of dimension $S^2 \times 1$, where $i = 1, \ldots, P$.

4.1.3 Unsupervised Feature Learning

Features suitable for this application are learned in an unsupervised fashion, using the patches extracted at the previous step. The patches extracted from the training images are clustered via K-means to learn a representative codebook.

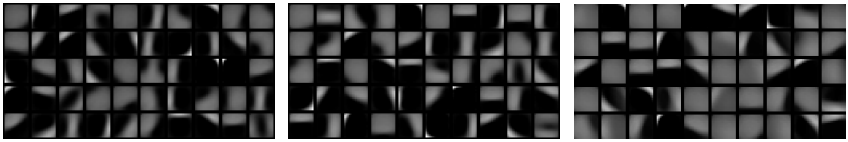
K-means is an unsupervised clustering algorithm that splits the space into K groups, which takes as input a set of data $p_i, \forall i = 1, \ldots, N$ and creates K clusters. Let U be the membership matrix assigning the point p_i to the class j. K-means is a *hard clustering* algorithm, namely a point in the space shall belong to one and only one cluster. The membership matrix encodes the set of patches belonging to a class by assigning either zero or one, that is

$$u_{ij} \in \{0, 1\}, \qquad j = 1, \ldots, K, \tag{4.1}$$

and it is required also that:

$$\sum_{j=1}^{K} u_{ij} = 1, \quad \forall i = 1, 2, \ldots, K. \tag{4.2}$$

Equation (4.2) forces the hard clustering constraint of the algorithm. K-means minimises the following objective function:



(a) Arabidopsis (A1) (b) Arabidopsis (A2) (c) Tobacco (A3)

Figure 4.6: Features learned with $K = 50$ using patches taken from the log-polar representation of the plants.

$$J(\theta, U) = \sum_{i=1}^{N} \sum_{j=1}^{K} u_{ij} \|p_i - \theta_j\|^2, \tag{4.3}$$

where θ_j is the centroid of the j-th cluster. Equation (4.3) can be optimised via an iterative algorithm, which at each step finds the closest points to θ_j's and the centroids are then updated, according to the points that are inside the j-th cluster [105]. K-means assumes that points in the space follow a Gaussian distribution and, since it is randomly initialized, the uniqueness of U is not guaranteed. In fact there could exist different combinations of u_{ij} minimising (4.3), depending on the initialization of the θ_j's. In Figure 4.6, learned codewords are shown for each dataset, that is A1, A2, and A3 (cf. Figure 4.1). Codewords in the learned codebook represent the most important parts of a plant, e.g. edges, petiole, homogeneous areas, etc. Each patch is then represented via *triangle encoding* [103], which is a non-linear mapping between p_i and all the centroids: for each patch the distances to all the θ_j's are computed $\delta_j(p_i) = \|p_i - \theta_j\|_2$ and the triangle encoding is obtained as

$$z_i = \max \left\{0, \overline{\delta}(p_i) - \delta(p_i)\right\}, \tag{4.4}$$

where $\delta(p_i) = [\delta_1(p_i), \delta_2(p_i), \ldots, \delta_K(p_i)]$ and $\overline{\delta}$ is the average of $\delta(p_i)$.

Figure 4.7: Pooling regions in the log-polar image.

4.1.4 Holistic Regression

When all vectors $\mathbf{z}_i$ are determined in an image, max-pooling is applied to compute a global descriptor which reduces the size of the descriptor and also adds invariance to small local transformations [106, 107]. The log-polar image $\tilde{I}$ is partitioned into T non-overlapping equally sized regions $\omega_t, t = 1, \ldots, T$ of the same height as $\tilde{I}$ and $D = 360/T$ pixels wide, example of which is shown in Figure 4.7. For a region ω_t, the max-pooling vector $\boldsymbol{\zeta}_t$ is built s.t.:

$$\zeta_t^{(k)} = \max_{\mathbf{z}_i \in \omega_t} z_i^{(k)}. \tag{4.5}$$

The global descriptor for I_j is obtained by concatenating all the corresponding $\boldsymbol{\zeta}_t$ in a new vector $\mathbf{x}_j$. Based on the observations $\mathbf{x}_j$, $j = 1, \ldots, N$, computed from the N training images, and y_j leaf counts, *Support Vector Regression Machine* (SVR) is employed to learn a regression model [108]. SVR shares the same principle of a support vector machine for classification, but instead of finding the best separation line maximising the margin between two classes, SVR finds the best fitting line that approximates the data, within a tolerance term ϵ. SVR minimises the amount of error outside the $\pm\epsilon$ threshold (the so-called SVR tube) [109]. The cost function being optimised is

$$\begin{array}{ll} \min_{\mathbf{w}} & \frac{1}{2}\|\mathbf{w}\|_2 + C\sum_{j=1}^{N}\left(\xi_j + \xi_j^*\right) \\ \text{subject to} & \begin{cases} y_j - \langle \mathbf{w}, \mathbf{x}_j \rangle - b & \leq \quad \epsilon + \xi_j \\ \langle \mathbf{w}, \mathbf{x}_j \rangle + b - y_j & \leq \quad \epsilon + \xi_j^* \\ \xi_j, \xi_j^* & \geq \quad 0 \end{cases} \end{array} \tag{4.6}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product, C is a positive constant, ξ_j and ξ_j^* are slack variables, and ϵ is the tolerance in the loss function [109]. Nonlinear SVR is achieved using *kernel methods*, which maps the data into a higher-dimensional feature space [110], by replacing the inner products in (4.6) with a kernel function $\phi(\cdot, \cdot)$. Here, the nonlinear relationship between image descriptors and number of leaves is modelled with the *radial basis function* (RBF), defined as:

$$\phi(\mathbf{x}, \mathbf{y}) = \exp\left(-\gamma \|\mathbf{x} - \mathbf{y}\|^2\right), \tag{4.7}$$

where $\gamma > 0$ is a model parameter.

SVR is trained using the vectors $\mathbf{x}_j$ as samples and the corresponding number of leaves y_j in I_j as the target value. The final estimation provided by the regression is a real number, which is rounded to the nearest integer. In order to predict the number of leaves in an image, it is converted into the log-polar domain and patches are extracted. Triangle encoding is computed over all the extracted patches and the resulting representations are pooled together, to obtain a global descriptor to provide to the regressor for prediction.

4.2 Experimental Results

In this section, evaluations of the leaf counting approach on image data showing rosette plants are examined. First, discussions on the experimental settings and evaluation criteria are presented. Next, results obtained on training and testing datasets are shown, comparing also to a variant of the proposed method aimed to learn

better representations for the central part of a plant. Moreover, comparisons with counting via segmentation method [20] and recent density based methods [73,74] are shown.

4.2.1 Setup

Image data. We use three datasets, that is A1, A2, and A3, consisting of images showing top views on individual plants provided by the *Leaf Counting Challenge* (LCC) CVPPP 2015 challenge organisers [12,111], examples of which are displayed in Figure 4.1. Each image in the training dataset is provided with a foreground segmentation mask (i.e., plant vs. background), leaf centre annotations, number of leaves. Testing sets include the corresponding plant foreground masks, but number of leaves are unknown, since results were evaluated by the organisers.

Choice of parameters. Only the green channel of the original RGB images was used for computational simplicity. Another choice could have been to use the grey-scale image, although it is highly correlated to the green channel. Alternatively we could opt for an illumination invariant transform such as the HSV (Hue, Saturation, Value), or colour transforms with class separation properties to obtain one or more channels [112]. The training procedure was repeated separately for each dataset and tuning parameters were chosen via cross-validation. Window width for the FG/BG ratio curve was set to $W = 20^\circ$ (see Section 4.1.2), since smaller values would result in a noisy FG/BG ratio curve, while larger ones would provide coarse results. In the patch extraction phase, we use $S = 15$ and $W' = 40^\circ$. K-means learns $K = 50$ centroids, using the *K-means++* initialisation criterion [113]. It was observed that large values of K lead to a coherent codebook with redundant clusters. Max-pooling is performed using $T = 5$ non-overlapping regions in the log-polar image (cf. Figure 4.7). Prior to the regression, global descriptors are normalised by subtracting the mean and dividing by the standard deviation (computed on all $\mathbf{x}_j$ vectors). For SVR,

a standard setup was used, that is $\gamma = 1/(TK)$, where TK is the dimension of $\mathbf{x}_j$, and loss parameter $\epsilon = 0.001$ [114].

Implementation details. The proposed method was implemented in Matlab. For training, due to the large size of the datasets, experiments were run on a CentOS 6.6 server with 4 CPUs Intel Xeon E7540 (6 cores with hyper-threading) and 64 GB of RAM. Although not necessary, the same configuration was employed for the testing phase as well. Overall, it takes approximately 20 seconds per image for training, out of which 80% is spent to learn the features, and less than 0.5 seconds to train the regressor. On the other hand testing (i.e., predicting the number of leaves in an unseen image) takes less than 3 seconds per image, since at test time we only need to extract the patches, obtain the encoding on the learned features, and apply the regressor to estimate the count.

4.2.2 The *Inner-Outer Leaf Counting* variant

In rosette plants young leaves tend to grow from the centre out and as such less mature leaves are closer to the centre. Such leaves are small, they heavily overlap, and due to low resolution are usually missed by many algorithms. In order to give more emphasis to younger leaves, a variant of the proposed method was set up and experimental results compared with the GLC. This variant, termed here *Inner-Outer Leaf Counting* (IOLC), relies on the centre coordinates of leaves to learn separately the inner part of the plant, namely the top-most in log-polar representation, and the lower part, namely the bottom-most in $\tilde{I}$. To separate the upper part from the lower one in a deterministic fashion, the log-polar image is scanned horizontally from the top downward (i.e., from the centre outwards). The separation line between the two parts is found at the vertical position where the first background pixel (from the plant mask) is found, as shown in Figure 4.8. The IOLC learns two different codebooks for the two parts respectively. In this case, max-pooling regions are $T = 2$ in the upper part and $T = 5$ in the lower one.

Figure 4.8: Graphical example how the inner and the outer part of a plant is determined: the log-polar image is scanned from the top downwards until any background pixel is found. The separation line in the log-polar representation corresponds to a circle enclosing the centre of a plant, containing the smallest leaves.

Finally, two separate SVRs are trained, where the target values y_j are chosen according to the number of annotations (leaves) inside the respective areas. The results of the two SVR predictions are added and then rounded.

4.2.3 Results

In this section, we will show the experimental results of the leaf counting algorithm (and its variant). Details on the evaluation metrics can be found in Section 2.4.

Comparing GLC and IOLC. In Table 4.1, training error for the GLC is reported, comparing it with the IOLC variant. Overall, GLC obtains better performance, reaching almost 80% agreement with the ground truth, indicating that features collected in the entire log-polar representation give satisfactory information to predict even leaves at the centre of the plant. Thus, since GLC requires only the number of leaves to train (an easier annotation problem) w.r.t. IOLC which need the leaf centres, it shows preferable behaviour.

Table 4.1: Training results of our proposed method (IOLC and GLC versions).

		A1	*A2*	*A3*	*All*
DiC	*IOLC*	-0.11(1.04)	-0.35(2.18)	-0.30(1.10)	-0.18(1.31)
	GLC	-0.13(0.88)	-0.48(2.20)	0.19(0.92)	-0.14(1.21)
|DiC|	*IOLC*	0.73(0.75)	1.45(1.65)	0.67(0.92)	0.84(1.01)
	GLC	0.48(0.74)	1.39(1.76)	0.48(0.80)	0.63(1.04)
%	*IOLC*	40.6	41.9	51.9	42.5
	GLC	77.3	74.2	92.6	79.0
MSE	*IOLC*	1.09	4.74	1.26	1.73
	GLC	0.78	4.94	0.85	1.48

Centre assessment. The evaluation of the centre of the plants, detected as described in Section 4.1.1, was assessed with respect to the goodness of the prediction on the training set. Figure 4.3a on page 46 depicts the segmentation mask and the detected centres, using:

- *Foreground centre of mass*: centroid computed over all the pixels in the plant segmentation mask;
- *Contour centre of mass*: centroid computed using only the contour pixels in the segmentation mask;
- *Bounding box centre*: centre of the smallest bounding box enclosing the plant.
- *Ellipse centre*: centre of the smallest ellipse enclosing the plant.

In Table 4.2 results of the training error w.r.t to the different approaches to detect the centre of the plant are shown. In general, the proposed method for detecting the centre of plants outperforms all the other ones, leading the regressor to a better estimation of the number of leaves.

Table 4.2: Evaluation of different methods to determine the centre of a plant. Bold values indicate best performace.

Method	*DiC*	$\|DiC\|$	*%*	*MSE*
Foreground	-0.13(1.00)	0.55(0.84)	62.5	1.01
Contour	-0.08(0.94)	0.50(0.79)	64.1	0.88
Bounding Box	-0.09(0.96)	0.54(0.79)	60.2	0.91
Ellipse	-0.04(1.03)	0.63(0.81)	54.7	1.05
Ours	**-0.13(0.88)**	**0.48(0.74)**	**77.3**	**0.78**

Augmenting the training set. The LCC datasets provide a limited amount of training images, which could penalize learning-based approaches. To explore this, the proposed algorithm was trained by varying the size of training data, whereas the remaining training part is used as a validation set. We find that the MSE in the training set reaches a plateau when we learn using 32 to 64 images, whereas the MSE in the validation set improves by $\sim$20%. This gave a motivation to augment the dataset by shifting the log-polar image, performing the full learning procedure on the *augmented* dataset. To accomplish this, 3 rightward circular shifts were applied to every training image, obtaining a 4-fold increase of each training set. The shift displacement is $D/4$, where D is the pooling region size (see Section 4.1.4). In Table 4.3 we report the training error using the augmented datasets. Comparing Tables 4.1 and 4.3 we observe that training with the augmented datasets leads to a considerable improvement in all cases, both for GLC and IOLC. Since GLC is simpler and more robust in the following only GLC is reported.

Comparison with density-based counting methods. Our global regression method GLC does not use leaf centre annotations. To compare its performance with methods that do use such topological information, density-based methods were adapted for the pur-

Table 4.3: Training results of our proposed method (IOLC and GLC versions) using the *augmented* dataset.

		A1+	*A2+*	*A3+*	*All+*
DiC	*IOLC*	0.00(0.72)	-0.16(1.42)	0.07(0.83)	-0.01(0.89)
	GLC	-0.02(0.76)	-0.29(1.36)	0.07(0.62)	**-0.05(0.87)**
|DiC|	*IOLC*	0.39(0.60)	0.87(1.12)	0.30(0.54)	0.49(0.74)
	GLC	0.41(0.65)	0.74(1.12)	0.30(0.54)	**0.45(0.74)**
%	*IOLC*	66.4	48.4	55.6	61.8
	GLC	82.8	77.4	88.8	**82.8**
MSE	*IOLC*	0.52	1.97	0.636	0.78
	GLC	0.58	1.77	0.37	**0.75**

pose [73,74]. The approach of *Lempitsky and Zisserman* [74], is used to learn a density function based on leaf centre annotations on the A1 training dataset. Dense SIFT descriptors from the green channel were extracted with bin size of 15. A codebook of $K = 800$ codewords was learned to represent data via one-hot encoding. This method exhibited lower performance compared to GLC, obtaining $\mathrm{DiC} = 0.82(1.97)$ and $|\mathrm{DiC}| = 1.59(1.42)$ (cf. Table 4.1 and Table 4.3). Moreover, the method proposed by *Arteta et al.* [73] was employed a well. The best results obtained on the A1 dataset was $\mathrm{DiC} = -0.5(10.5)$ and $|\mathrm{DiC}| = 7.3(7.4)$, confirming that the proposed approach is outperforming state-of-the-art density-based object counting methods, and reaffirming conclusions of theirs [73], that such methods are unable to accommodate object size variability.

Testing Results. To estimate leaf counts on the images in the testing set, codebooks and SVR models were learned in advance on the *augmented* training sets. Results were submitted to the organisers only for GLC: Table 4.4 summarised the performance of the proposed method, together with the counting-via-segmentation ap-

Table 4.4: Results for the testing set of our proposed GLC method with regressor(s) and features learned on the *augmented* dataset. For comparison the findings of *Pape and Klukas* [20] on the same testing set are shown (values for only two metrics were available).

		A1+	*A2+*	*A3+*	*All+*
DiC	*GLC*	-0.79(1.54)	-2.44(2.88)	-0.04(1.93)	**-0.51(2.02)**
	Ref [20]	-1.8(1.8)	-1.0(1.5)	-2.0(3.2)	-1.9(2.7)
|DiC|	*GLC*	1.27(1.15)	2.44(2.88)	1.36(1.37)	**1.43(1.51)**
	Ref [20]	2.2(1.3)	1.2(1.3)	2.8(2.5)	2.4(2.1)
%	*GLC*	27.3	44.4	19.6	24.5
	Ref [20]	-	-	-	**-**
MSE	*GLC*	2.91	13.33	3.68	4.31
	Ref [20]	-	-	-	**-**

proach proposed by *Pape and Klukas* [20], the winners of the previous leaf segmentation challenge. The proposed method outperforms the approach in [20]; in particular, significantly improvement on the accuracy on A1 and A3 datasets can be noticed. Overall, number of leaves predicted by GLC is off by at most ± 1 leaf in 57% of the cases. The A2 dataset contains several mutants and some subjects exhibit dwarfism, appearing very small in the images. When such images, or images with many small young leaves in the centre, are transformed into the log-polar domain, the effect of interpolation introduces artefacts causing performance loss. Despite A3 dataset includes very young (and relatively small) plants, the effects discussed before are compensated by increased image resolution. In fact, training and testing error in A1 and A3 are similar, even if the A3 dataset contains the least amount of training images. This motivates future investigations of specialised features for regions close to the centre.

4.3 Discussion

Significant effort was invested on counting leaves in images of rosette plants –a challenging vision problem due to variability in terms of size, appearance, and rotation of leaves. A machine learning-based approach to estimate the number of leaves from top-view images was proposed. Global features for each image were computed, using local patches extracted from the log-polar domain, which accounts for rotation and scale variability. Global features were used to train an SVR to map the feature space to the real number set. Experiments showed that with adequate training data, at testing time for an unseen image satisfactory accuracy (within the ± 1 leaf error) in counting is obtained within a few seconds (per image), opening the road to automated and reliable leaf count estimation in high throughput phenotyping applications. Using standardised datasets in the context of the Leaf Counting Challenge, samples of which are depicted in Figure 4.1, of the CVPPP 2015 workshop, this method outperforms a previous state-of-the-art method [20] on the same data. It was also compared with state-of-the-art methods for counting via density estimation, showing that our learning framework outperforms the methods in [73,74] in dataset we tested (A1). We also found that augmenting the training set, by circularly shifting the log-polar representations, increases performance.

The proposed method is simple to train. It requires input images and full plant segmentation (which for plants is easier to obtain than other applications). In terms of annotation it requires only a total leaf (object) count per image. This is much easier than centres or bounding boxes required for density or detection based methods. Experiments showed that with adequate training data, at testing time for an unseen image satisfactory accuracy in counting is obtained within a few seconds (per image), opening the road to automated and reliable leaf count estimation in high throughput phenotyping applications. Integrating such learning-based approaches to centralised cloud based analysis frameworks such as the one available at `http://www.phenotiki.com` would increase even more the

reach of automated high throughput phenotyping [28].

Despite the encouraging results obtained by this method, several limitations exist. The log-polar transformation, which helps to cope with rotation and plant size, tends to distort information at the centre of the plant, where juvenile leaves can be found. Therefore, to cope with smaller leaves in the central region, it is necessary to represent the plant images differently. Inaccurate results are due mainly to the following factors: low-discriminative features for the plant's centre and overlapping regions, rounded real-valued estimation, and bias of the regressor. One way to solve these issues is to employ a better feature representation.

In the next chapter, we will show a shallow neural network to learn rotation-invariant features from images. We devised this architecture to learn a more compact feature space to represent plant images. The rotation-invariant network will replace the K-means step from the GLC pipeline, learning features from patches extracted from the image and skipping the log-polar transformation.

5

Learning Rotation-Invariant Features

In the previous chapter, we presented a machine-learning approach for leaf counting, using a SVR trained on holistic image descriptors. Specifically, such a descriptor represents a plant image, using image features learned using K-means. Although the learned features show that the method learns anatomical parts of a plant (e.g., leaf tip, lamina, petiole, as shown in Figure 4.6 on page 49), they might occur several times at different rotations. For instance, in Figure 5.1, we display an example: the highlighted centroids represent a segment of a petiole, but they appear rotated by $90°$. Although it should not occur, as the images are transformed into the log-polar representation to address the circular arrangement of

This chapter is based on:

- M. V. Giuffrida and S. A. Tsaftaris, "Rotation-Invariant Restricted Boltzmann Machine Using Shared Gradient Filters," in International Conference on Artificial Neural Networks, 2016.
- M. V. Giuffrida and S. A. Tsaftaris, "Explicit Factorization of Rotations in Restricted Boltzmann Machines", Under review on IEEE Transactions of Image Processing, 2018.

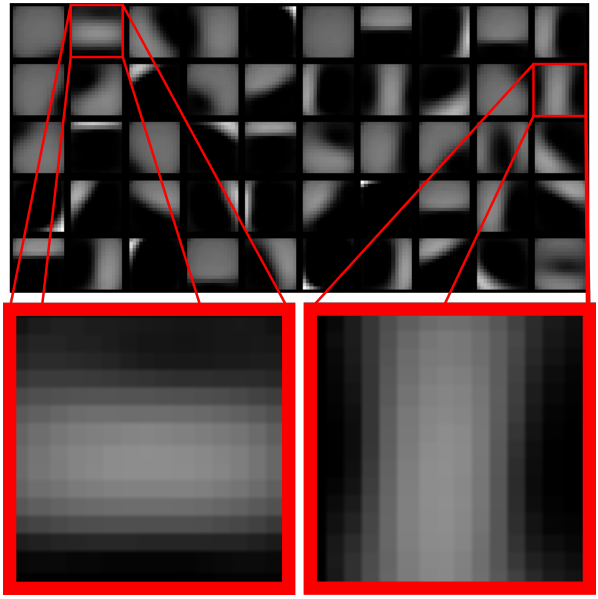


Figure 5.1: Learned features in [7] (c.f. Chapter 4) show centroids that are rotated versions of each others.

leaves, this heuristic may fail, because not all the leaves are straight w.r.t. the plant centre. Leaves can exhibit in-plant bending that can result in undesired horizontal edges in the log-polar representation.

Since neural networks proved to be state-of-the-art feature extractors, we propose a method to learn such features in an unsupervised manner with some properties. Specifically, due to the radial arrangement of the leaves in a rosette plant, we desire that our algorithm is able to extract features that are rotation-invariant – that is, a representation of a leaf (or part of it) is the same regardless of its in-plane rotation. In this chapter, we present a shallow neural network that can learn features in an unsupervised manner. We extend a well-known model in the literature known as the *Restricted Boltzmann Machine* (hereafter RBM) [23], to learn data-driven rotation-invariant features.

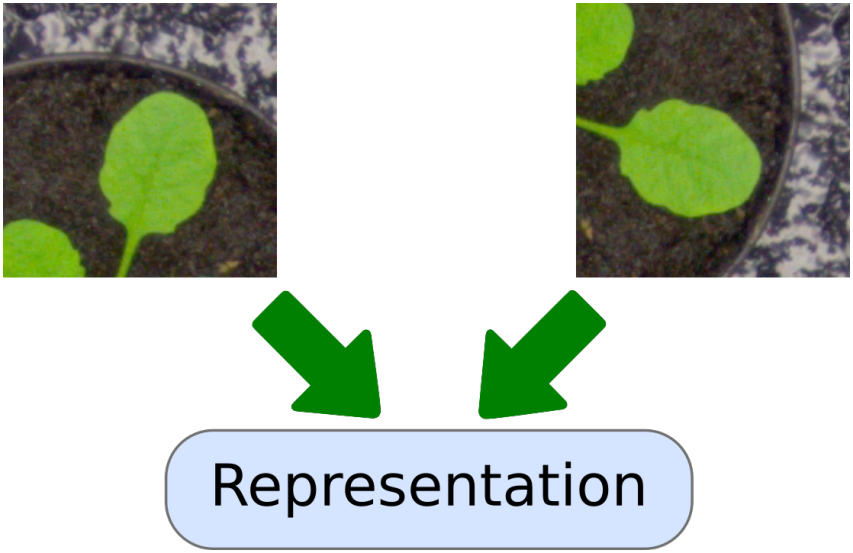


Figure 5.2: Visual example of rotation invariance. The two versions of a leaf share the same representation.

5.1 Background

In this section, we will introduce a brief background on invariant features and RBM. The topics reported in this section are essential to understand the remainder of this chapter.

5.1.1 Some Feature Properties

Invariance. This property says that a feature extractor outputs the same representation when a transformation occurs in an image. Using the definitions in Section 2.2, a feature extractor $\varphi : \mathcal{X} \to \mathcal{Y}$ is said to be invariant to a transformation $T : \mathcal{X} \to \mathcal{X}$ if:

$$\varphi(x) = \varphi(T(x)). \tag{5.1}$$

We are interested in rotation-invariance, where T is a particular

transformation matrix that performs in-plane rotations, as graphically depicted in Figure 5.2.

Equivariance. Differently, equivariance is a property of features that, when T is applied to the x, there exists a transformation $T' : \mathcal{Y} \to \mathcal{Y}$ such that

$$\varphi(T(x)) = T'(\varphi(x)). \tag{5.2}$$

Intuitively, if we know what transformation the data undergo, we also know that a known corresponding transformation to the features is also applied.

From an experimental standpoint, the Equation (5.1) cannot exactly hold, due to the numeric nature of the data and the learning process. Furthermore, a transformation T would surely introduce nuisance to the transformed image, due to the discrete representation of the images and pixel interpolation.

5.1.2 Restricted Boltzmann Machine

A *Restricted Boltzmann Machine* is a shallow network able to learn features in an unsupervised manner [23]. A RBM is a graphical model formed by a bipartite graph: one set of nodes is called the *input layer* and the other set is called the *hidden layer* (Figure 5.3). In a RBM, we are interested in maximising the following joint probability:

$$p(x, h) = \frac{e^{-E(x,h)}}{Z}, \tag{5.3}$$

where $E(x, h)$ is an energy function (c.f. Equation (5.11)), Z is the *partition function* used as normalisation factor that ensures that $p(x, h)$ is a probability (it sums to 1). The energy function is defined as follows:

$$E(x, h) = -\mathbf{h}^T W \mathbf{x} - \mathbf{c}^T \mathbf{x} - \mathbf{b}^T \mathbf{h}, \tag{5.4}$$

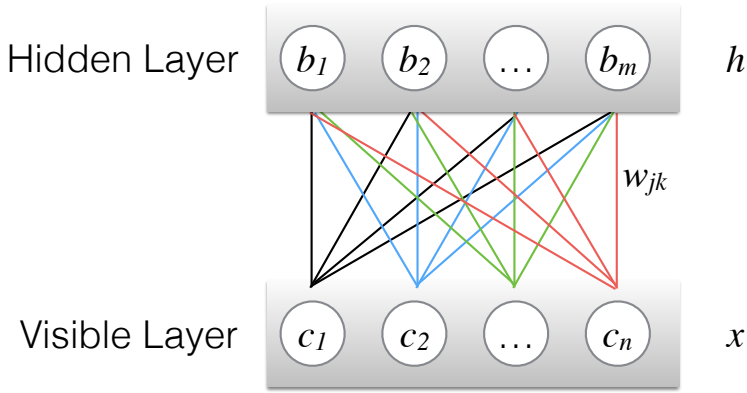


Figure 5.3: Representation of an RBM. An input vector x is given to the network to compute the probability of observing h given x. During optimisation, the best values for the weight matrix W and bias vectors $\mathbf{c}$ and $\mathbf{b}$ are found.

where $W \in \mathbb{R}^{m \times n}$ is the weight matrix, $\mathbf{c} \in \mathbb{R}^n$ is the bias vector of the input layer, and $\mathbf{b} \in \mathbb{R}^m$ is the bias term for the hidden layer. If not otherwise stated, we will use the column-vector convention.

Equation (5.11) is minimised through the *contrastive divergence* (shortened as CD) algorithm [115]. The contrastive divergence minimises the negative log-likelihood of the training set, using stochastic gradient descent. The partial derivative computed for the t-th training sample $x^{(t)}$ w.r.t. the parameter $\theta \in \Theta = \{W, \mathbf{b}, \mathbf{c}\}$ is

$$\frac{\partial - \log\left(p(x^{(t)})\right)}{\partial \theta} = \underbrace{\mathbb{E}\left[\frac{\partial E(x^{(t)}, h)}{\partial \theta} \middle| x^{(t)}\right]}_{\text{Positive phase}} - \underbrace{\mathbb{E}\left[\frac{\partial E(x, h)}{\partial \theta}\right]}_{\text{Negative phase}} . \quad (5.5)$$

The partial derivative in equation (5.5) is computed w.r.t. each parameter $\theta \in \Theta$. The computation of the negative phase is intractable, because the expected value all over the x's and h's is hard to compute. Therefore, it is approximated with $\tilde{x}$, using *Gibbs sam-*

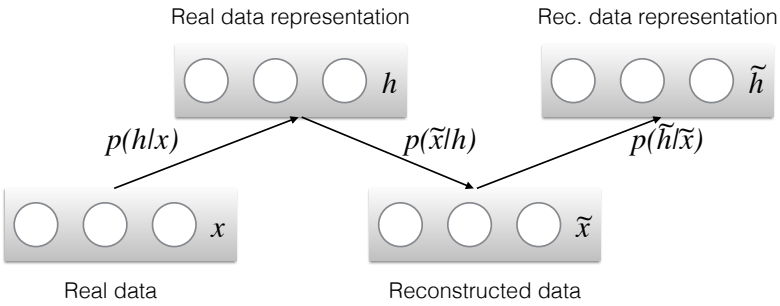


Figure 5.4: Graphical representation of Gibbs sampling. An input is provided to the visible layer to compute Equation (5.6). From this distribution, samples are drawn and a $\mathbf{h}$ vector is computed. The, this vector is used to compute the reconstructed data $\tilde{\mathbf{x}}$. The representation $\tilde{\mathbf{h}}$ is computed for the reconstructed data. These four vectors are used to determine the gradients to update the parameters in Θ.

pling [116], which computes alternately the following conditional probabilities:

$$p(h_j|x) = \sigma\left(\mathbf{W_j}.\mathbf{x} + b_j\right), \tag{5.6}$$

$$p(x_k|h) = \sigma\left(\mathbf{h}^T\mathbf{W}_{.\mathbf{k}} + c_k\right), \tag{5.7}$$

where $\mathbf{W_j}.$ and $\mathbf{W}_{.\mathbf{k}}$ denote the j-th row and k-th column in W respectively, and $\sigma(\cdot)$ denotes the logistic function (c.f. Table A.1).

Gibbs sampling. As said, the Gibbs sampling is used to approximate the negative phase in Equation (5.5). When a sample $\mathbf{x}$ is taken from the training set, Equation (5.6) is computed for all the nodes in the hidden layer. Equation (5.6) gives the success probability of a Bernoulli distribution. Then, samples of such a distribution are drawn to make a vector $\mathbf{h}$, used to compute Equation (5.12). After sampling from this last distribution, we obtain the reconstructed vector $\tilde{\mathbf{x}}$. From this reconstructed input, we compute $\tilde{\mathbf{h}}$ using Equation (5.6).

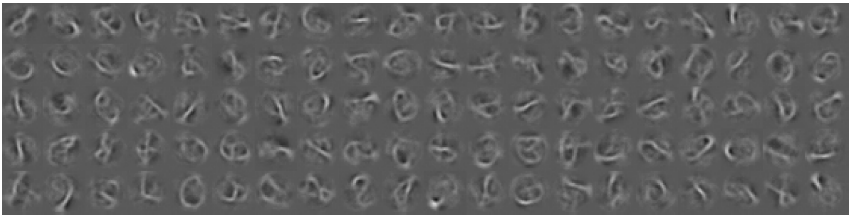


Figure 5.5: Example of learned filters using RBM on the MNIST-rot dataset [8].

Parameters update. Once we performed the Gibbs sampling successfully, we can use the obtained vectors to update the parameters in Θ as follows [115]:

$$W \leftarrow W - \eta \left(\mathbf{h}\mathbf{x}^T - \tilde{\mathbf{h}}\tilde{\mathbf{x}}^T \right), \tag{5.8}$$

$$b \leftarrow b + \eta \left(\mathbf{h} - \tilde{\mathbf{h}} \right), \tag{5.9}$$

$$c \leftarrow c + \eta \left(\mathbf{x} - \tilde{\mathbf{x}} \right), \tag{5.10}$$

where η is the learning rate of the stochastic gradient descent.

Visual representation of *W*. Each column in W represents a filter that is applied to the input data. As an example, if the size of the hidden layer is $H = 500$, the network learns 500 filters, which can be visualised as images. In Figure 5.5, a subset of the 500 filters learned by an RBM on the MNIST-rot dataset [8] is shown.

Gaussian formulation. The energy function in Equation (5.11) is used when the inputs can be modelled as binary data (e.g., black and white images). However, the energy function can be adapted to accommodate continuous data. Instead of assuming the visible layer follows a Bernoulli distribution, we will assume that samples are drawn from a Gaussian distribution. Therefore, an RBM with continuous visible units optimises the following energy function:

$$E(\mathbf{x}, \mathbf{h}) = -\mathbf{h}^T W \frac{\mathbf{x}}{\sigma^2} - \frac{\|\mathbf{x} - \mathbf{c}\|^2}{2\sigma^2} - \mathbf{b}^T \mathbf{h}, \tag{5.11}$$

where σ is the standard deviation of the Gaussian distribution. Consequently, the probability in Equation (5.12) becomes:

$$p(x_k|\mathbf{h}) = \mathcal{N}(x_k|\mathbf{h}^T \mathbf{W}_{\cdot\mathbf{k}} + c_k, \sigma_k^2). \tag{5.12}$$

Drawbacks. The original formulation of RBM is not able to capture specific structures in the data. Specifically, the determined features do not carry the invariance property. The typical approach to train an algorithm with more variability is to perform dataset augmentation. Although it is a common practice, it does not lead an algorithm to be invariant to a transformation. In fact, the algorithm will learn how to represent the rotated version of an image, but there is no guarantee such representations are equal. Thus, dataset augmentation does not ensure that Equation (5.1) holds. To this end, several modifications to the original RBM model have been proposed in literature to capture known variability in the dataset, such as rotations.

5.2 Related Works to Rotation Invariance Learning

In [21], a transformation invariant RBM is proposed, where images are subjected to a predefined set of transformations $\mathcal{T}$. Specifically, for all the transformation matrices $T \in \mathcal{T}$, each sample in the training set is transformed during the contrastive divergence. Representations in Equation (5.6) are computed for all the transformed versions and the final representation is obtained by max-pooling.

In [117] an RBM model that learns equivariant features is proposed, whereby adding a new variable to be inferred within the hidden units, this variable is then used to rotate learned weights accordingly. Specifically, using the notation in Section 5.1.2, such a

latent variable in the j-th hidden unit will apply a rotation to $\mathbf{W_{j\cdot}}$, the corresponding filter of h_j.

In [118], the authors present a general framework to inject invariance to linear transformation of data. Specifically, they also show that their method can be used to generalise *Convolutional RBM* [119] to achieve rotation invariance. Their formulation adds transformations T to data in the *product of experts* [120] and, when applied to RBM, the resulting model shares similarities with [21].

In [121], an additional step of the backpropagation algorithm used to train DBN is introduced. After the DBN has been trained, the weights of the bottom layer are transformed and the training is performed again, until all the transformations are applied.

In [122], the authors propose an RBM where input images are divided into non-overlapping blocks. Then, patches are extracted on SIFT keypoints [90] and subsequently rotated and scaled accordingly. Since patches are oriented according to the dominant orientation detected with SIFT, the RBM learns de-facto rotation-invariant features.

The aforementioned methods share the following drawbacks: either they are limited to the set of transformations considered within the model, or they involve deep networks in the hope of learning better transformation invariant features [21,117,121], albeit increasing computational demand. Furthermore, these methods alter input images with transformations, which introduce additional noise and nuisance (e.g., pixel interpolation) during training.

5.3 Explicit Rotation-Invariant Restricted Boltzmann Machine

In this section, we present the *Explicit Rotation-Invariant Restricted Boltzmann Machine* (ERI-RBM), which can model the nuisance caused by rotated versions of the same image or patch, without actually applying any transformation to the data. Our method considers a set of weight matrices (similar concept as in C-RBM [123]) and each sample

is provided to the visible layer with its dominant orientation [122]. This information is used to select a particular weight matrix during the Gibbs sampling to compute gradients of parameters. The contribution given by the new update gradients is shared among the other weight matrices, rotating the filters accordingly [121] (cf. Figure 5.6). Experiments on MNIST-rot show superior performance to several baseline benchmarks and a recent method from the literature.

Our contributions are multi-fold: (i) rotation is treated explicitly, without rotating the image patterns, in contrast to for example [21]; (ii) we adopt a shallow model using a limited amount of additional weight matrices, instead of deep architectures [119]; (iii) we share the contribution coming from a weight matrix with the other ones, rotating the learned filters by suitable angles.

5.3.1 Proposed Model

In this section, we discuss how to embed the concept of rotation-invariance explicitly in an RBM. Since input data are images of size $w \times h$, we will assume that the visible layer is arranged in a matrix of size $w \times h = d$. Each row in the weight matrix W, connecting visible units to hidden units, is a d-dimensional vector. Therefore, each row in W can also be arranged in matrix form of size $w \times h$. Henceforth, we will refer to rows in the weight matrix W as *learned filters* and rows in ∇W as *update filters*, which is the gradient computed during the Contrastive Divergence algorithm.

Mathematical Formulation

Let Φ be a set of evenly distanced angles $\Phi = \{\phi_0, \phi_1, \ldots, \phi_{S-1}\}$, such that for any $i \leq j \implies \phi_i \leq \phi_j$. In our model, we augment the number of weight matrices $W \in \mathbb{R}^{H \times V \times S}$, such that every angle ϕ_s is associated to a matrix $W^{(s)}$. Here, H is the number of hidden units, V the number of visible units, and S is the number of angles. In addition, each weight matrix has an associated bias vector $\boldsymbol{b}^{(s)}$. Hence, we rewrite the energy function characterising the standard

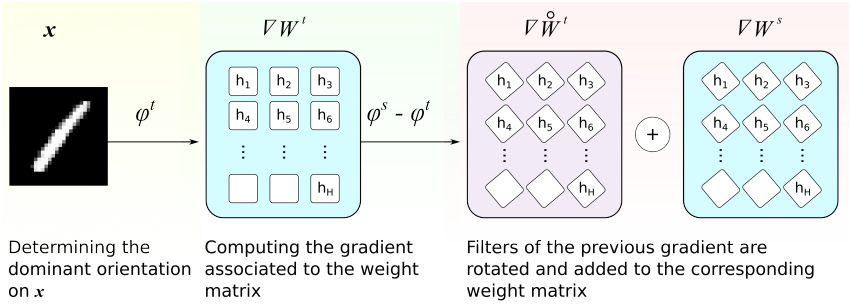


Figure 5.6: The dominant orientation ϕ^t is determined for the provided image and is used to compute the gradient $\nabla W^{(t)}$. The contribution of this gradient is shared amongst the other weight matrices $\nabla W^{(s)}$, $s = 1, 2, \ldots, S$, $t \neq s$, rotating the learned filters by the angle $\phi^s - \phi^t$ to generate the $\nabla \mathring{W}^{(t)}$ term.

Restricted Boltzmann Machine formulation as follows:

$$E(\boldsymbol{x}, \boldsymbol{h}; s) = -\boldsymbol{h}^T W^{(s)} \boldsymbol{x} - \boldsymbol{c}^T \boldsymbol{x} - \left[\boldsymbol{b}^{(s)}\right]^T \boldsymbol{h}, \tag{5.13}$$

where $W^{(s)}$ is the s-th weight matrix, $\boldsymbol{b}^{(s)}$ is the bias vector for the hidden layer associated to $W^{(s)}$, with $s = 0, 1, \ldots, S-1$, and $\boldsymbol{c}$ is the bias vector for the visible layer. The index s is uniquely determined on each input image $\boldsymbol{x}$, and will be discussed thoroughly below. Because of the modification in (5.13), all the equations involved in the CD (c.f. Section 5.1.2) algorithm have to be rewritten. Specifically, the conditional probabilities become:

$$p(h_k = 1 | \boldsymbol{x}; s) = \sigma\left(b_k^{(s)} + \boldsymbol{W}_{\boldsymbol{k}\cdot}^{(\boldsymbol{s})} \boldsymbol{x}\right), \tag{5.14}$$

$$p(v_j = 1 | \boldsymbol{h}; s) = \sigma\left(c_j + \boldsymbol{h}^T \boldsymbol{W}_{\cdot \boldsymbol{j}}^{(\boldsymbol{s})}\right). \tag{5.15}$$

During the optimisation algorithm, an image $\boldsymbol{x}$ with dominant orientation ϕ_s is provided to the Gibbs sampling. After a sufficient

number of alternating computations of (5.14) and (5.15), the gradient $\nabla W^{(s)}$ can be computed, whose contribution is shared with the remaining matrices in W. To update $\nabla W^{(t)}$, $1 \leq t < S$, $t \neq s$, we transform the update filters in $\nabla W^{(s)}$ which are then added to the t-th gradient. Specifically, since we can represent rows in $\nabla W^{(s)}$ as images, they can be rotated by an angle $\theta = \phi_t - \phi_s$. Therefore, we define a new *shared update filter* term $\nabla \mathring{W}^{(t)}$, such that

$$\nabla \mathring{W}^{(t)} = R_\theta(\nabla W^{(s)}) \equiv \begin{pmatrix} R_\theta\left(\boldsymbol{\nabla W_{1\cdot}^{(s)}}\right) \\ R_\theta\left(\boldsymbol{\nabla W_{2\cdot}^{(s)}}\right) \\ \vdots \\ R_\theta\left(\boldsymbol{\nabla W_{H\cdot}^{(s)}}\right) \end{pmatrix}. \tag{5.16}$$

where $R_\theta = [\cos\theta \ -\sin\theta; \sin\theta \ \cos\theta]$ defines the 2D rotation matrix by an angle θ. This operation may generate filters bigger than the input layers and we crop them such that the filter size remains $w \times h$. At this point, the final expression for the gradient $\nabla W^{(s)}$ is updated as follows:

$$\nabla W^{(s)} := \nabla W^{(s)} + \nabla \mathring{W}^{(s)}. \tag{5.17}$$

Note that (5.17) will be utilised within the Stochastic Gradient Descent step of the CD algorithm. Therefore, $\nabla W^{(s)}$ will be multiplied by a learning rate η that typically has values set in the order of 10^{-3} (further details are discussed in [124]). Hence, any side effects originating from pixel interpolation are minimised, precisely because of the small η. Gradients $\boldsymbol{\nabla b}^{(s)}$ are computed as described in Section 5.1.2, using samples $\boldsymbol{v}$ with the associated dominant orientation φ_s.

Determining the Dominant Orientation Using an Exogenous Process

Each image $\boldsymbol{x}$ is associated to an angle ϕ_s, determined by the histogram of oriented gradients from $\boldsymbol{x}$ [125]. Derivatives along the

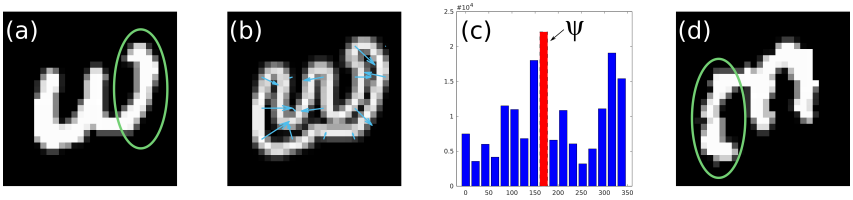


Figure 5.7: Computation of the dominant orientation for a sample image taken from the MNIST dataset: (a) original sample, (b) gradients of the image, (c) histogram of oriented gradients with highlighted mode ψ, (d) sample rotated by ψ degree. The region marked by a green ellipse corresponds to the same portion of the number 3 in the original and rotated image. Observe the differences due to image interpolation introduced during rotation.

x and y directions are computed and the angle of each gradient vector can be determined. All the vectors are accumulated into a histogram with S bins and the angle ψ with the highest frequency is found. Formally, the index $s = \text{argmax}_j \, \phi_j$, such that $\phi_j \leq \psi$, $\phi_j \in \Phi$. Figure 5.7 shows graphically those steps: from the original image pattern (a), derivatives are computed using Sobel filters (b). Subsequently, we build the weighted histogram of oriented gradients and the angle with the highest frequency ψ is selected (c). We highlight in red the 9-th bin of the histogram, hence $s = 9$ for the illustrated example. In (d) we report a rotated version of the sample image by ψ degree to show the deleterious effect of image interpolation.

Since strong edges near image boundaries may bias the estimation of the dominant gradient, the magnitude of the corresponding vectors is weighted with a Gaussian kernel, with $\sigma = \frac{min\{w,h\}}{5}$ (width and height of $\boldsymbol{x}$ respectively), such that central gradients contribute more than those at the boundaries. (We found this value covers evenly the entire image without exceeding its size.)

5.3.2 Experimental Results

Setup. We used the MNIST-rot dataset[1] [8], containing $10,000$ images for training, $2,000$ for validation, and $50,000$ for testing. This dataset is derived from the MNIST dataset, where samples were rotated by random angles. To enable comparison with other methods, for consistency, we kept this dataset splitting, and we did not perform cross-validation (that could have provided variances for statistical analysis). Since each image contains several non-zero pixels close to 0, we threshold them at a value $\tau = 0.3$. We compare ERI-RBM with several informative baselines and a recent invariant method:

i. *Classical RBM:* We trained a standard Bernoulli Restricted Boltzmann Machine and compared results with our Explicit Rotation-Invariant RBM;

ii. *Dominant RBM (D-RBM):* We built a simplified model that learns an RBM for each dominant orientation, splitting the training set into S partitions, associated to a different RBM (ie., we have S independent RBMs);

iii. *Oriented RBM (O-RBM):* We pre-process the dataset by aligning all images according to their dominant orientation to a reference orientation and train a single RBM.

iv. *TI-RBM*: We also compared with the method in [21], using the authors implementation[2]. Extracted features are provided to the following classifiers: linear and RBF SVM [126], softmax [87], and K-NN [127].

Parameters. We set the number of hidden units to $H = 100$, while we progressively increased the number of bins S, used to generate the histogram of orientations. Following the instructions in [124], we

[1] `http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/DeepVsShallowComparisonICML2007`

[2] Available at `https://github.com/kihyuks/icml2012_tirbm`

Table 5.1: Testing accuracies of standard RBM, Dominant RBM, Oriented RBM, TI-RBM [21], and our proposed ERI-RBM.

	RBF SVM	**Lin. SVM**	**Softmax**	**K-NN**
RBM	87.37%	59.27%	57.80%	82.69%
D-RBM (S=4)	83.44%	58.95%	56.80%	78.84%
D-RBM (S=9)	79.18%	53.62%	50.76%	73.56%
D-RBM (S=18)	69.84%	49.20%	46.58%	63.61%
O-RBM (S=18)	87.37%	58.99%	57.80%	82.69%
ERI-RBM (S=4)	78.49%	60.27%	58.31%	74.97%
ERI-RBM (S=9)	91.27%	74.87%	73.02%	88.48%
ERI-RBM (S=18)	**92.08**%	**77.69**%	**75.84**%	**89.34**%
TI-RBM [21] (S=18)	80.63%	69.10%	68.20%	73.60%

set the learning rate $\eta = 10^{-3}$, the Contrastive Divergence algorithm is iterated up to 200 epochs, and a constant momentum $\alpha = 0.9$ was used. The parameters for SVM were found using logarithmic grid search and best values are (i) *RBF SVM:* $C = 10, \gamma = 0.1$; (ii) *Linear SVM:* $C = 0.1$. We set arbitrary $K = 3$ for the K-NN, using the Euclidean distance as metric. For TI-RBM [21], a set of $K = S$ transformations are considered, which is each associated with an array of H hidden units, while a single weight matrix W is considered. The final representation used during inference is obtained by max-pooling. To make the comparison to ERI-RBM fair, for TI-RBM the sparsity term was disabled, and we set the number of hidden units to $H = 100$.

Discussion. We report our results in Table 5.1 and we noticed that nonlinear SVM gave the best performance in all the cases. The baseline is given by RBM with an accuracy of 87%. Tests using D-RBM show a gradual loss of accuracy as the number of dominant orientations S is increased. This behaviour can be attributed

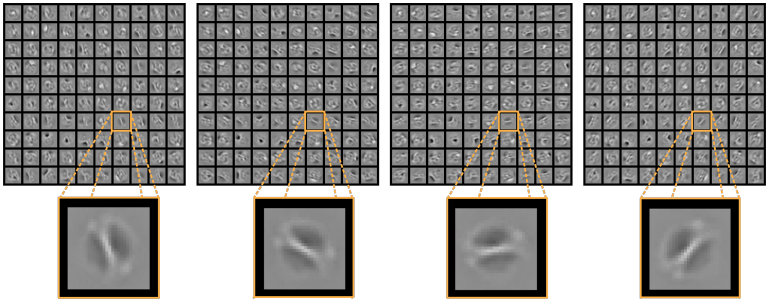


Figure 5.8: Filters learned by our ERI-RBM at $S = 9$. We highlight a filter that appears at rotations $0°$, $40°$, $80°$, and $120°$, showing that our model learns rotation-invariant filters. The remaining weight matrices are omitted for brevity.

to the lack of information sharing amongst the RBMs, since they were each trained independently with less data (per RBM). Overall, our proposed model outperforms the baseline RBM ($S \geq 9$). At $S = 4$, ERI-RBM has a loss of performance, because of the coarse quantization of the 2π space: angles $0°$, $90°$, $180°$, and $270°$ will have orthogonal rotations when shared update filters are computed for neighbour matrices, causing the propagation of sharp rotations that do not contribute much. As the number of S increases, ERI-RBM has a $+13\%$ of improvement, showing that our model is able to learn rotation-invariant features. This is also displayed in Figure 5.8, showing learned filters when $S = 9$. O-RBM shows no improvement compared to RBM, demonstrating that the contribution provided by the shared update filters increases the discriminative power of the final representation. Note that we also trained classical RBM with $H = 1000$, noticing an improvement of 2%, still lower than ERI-RBM. Finally, using the same experimental setup, ERI-RBM outperformed [21] by $+12\%$ in testing accuracy. (These results are different from those reported in [21] since sparsity is not present and we used less units.). Our approach does rely on the determination of orientation, which could be seen as a limitation. Preliminary

results (results reported in [128]), obtained by artificially perturbing the orientation estimate, show that we are tolerant to such errors up to ± 4 bins off on the original estimate. This remains to be confirmed in images with cluttered background.

5.4 Determining the Dominant Orientation with an Endogenous Process

In this section, we presented an improved method to learn rotation-invariant features with RBM, explicitly factorising for rotations. Similarly as in Section 5.3, this revised version uses a weight matrix per each orientation. As shown in Figure 5.9, during training, an input image is passed through all weight matrices. To determine the orientation of the input image, the reconstruction error (per orientation) is computed. The weight matrix that best reconstructs the input is chosen, and then the gradient for that matrix is computed to update the parameters. Then the gradients are shared across the other weight matrices, using the sharing gradient step (c.f. Section 5.3). The training process is regularised with a KL-Divergence term that enforces a prior distribution of the rotations.

Hence, we summarise the contributions of this revised method as follows:

i) the estimation of the dominant orientation is performed (here done via reconstruction rather than relying on exogenous processes);

ii) a regularisation term based on a KL-Divergence term;

iii) mathematical proofs of the invariance achieved by our model.

5.4.1 Theory

In our formulation of rotation-invariant Restricted Boltzmann Machine, we assume a set $\mathcal{S} = \{R_0, R_1, \ldots, R_{S-1}\}$ of $S = |\mathcal{S}|$ equidistant rotations with $R_i \in \mathbb{R}^{V \times V}$ being by $\phi_i = i\frac{2\pi}{S}$, for all $\phi_i \in \Phi$. Any

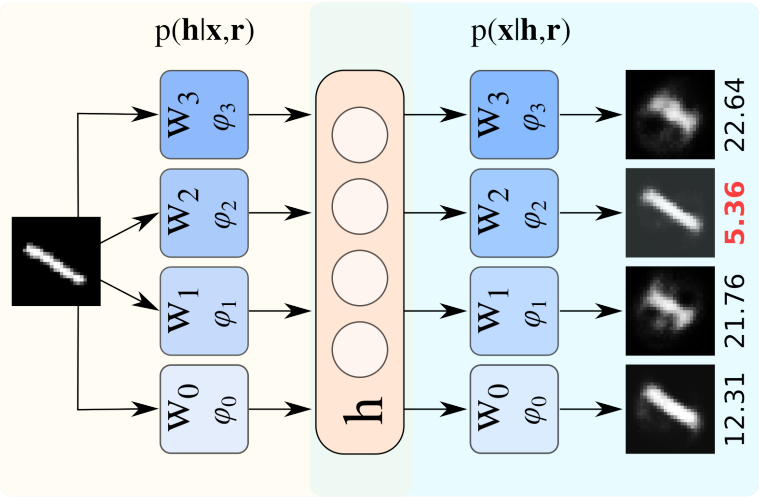


Figure 5.9: Representation of the proposed rotation-invariant RBM. In this example, we have $S = 4$ rotations, corresponding to the equidistant angles $\Phi = \{\phi_0 = 0°,\ \phi_1 = 90°, \phi_2 = 180°, \phi_3 = 270°\}$, each of those associated with a matrix W_s. When an image is provided to the network, the weight matrix minimising the reconstruction error is chosen, as highlighted in bold red. We depict the unfolded steps of the CD-1 [9].

rotation R_i can be represented by a transformation matrix satisfying the following properties:

1. *orthogonality:* $RR^T = R^T R = I$, $\det(R) = \pm 1$,
2. *linearity:* $R(x + y) = R(x) + R(y)$,
3. *closure (on the set $\mathcal{S}$):* if $R_i, R_j \in \mathcal{S}$, then $R_\kappa(x) = R_i(R_j(x)) = R_j(R_i(x))$ and $R_\kappa \in \mathcal{S}$, s.t. $\kappa = m(i, j)$.

The *modulo function* $m(i, j)$ allows for cyclical indexing over the elements in $\mathcal{S}$, such that the values of κ are always in the interval $[0, S - 1]$. The modulo function is defined as follows:

$$m(i,j) = (i+j) \bmod S, \tag{5.18}$$

where $i, j \in \{0, \pm 1, \pm 2, \ldots, \pm(S-1)\}$. The indices i, j can assume negative values, with the assumption that R_{-i} represents a rotation by $-\phi_i$. (It can be proven that $-\phi_i \in \Phi$.)

Application of a Transformation Matrix

Here we assume the column-vector convention (e.g. a vector v is of size $n \times 1$). Therefore, a transformation of $\mathbf{x}$ by a rotation matrix $R \in \mathbb{R}^{V \times V}$ is applied as $R(\mathbf{x}) = R\mathbf{x}$. Similarly, we can apply a rotation R to the column of a weight matrix $W \in \mathbb{R}^{V \times H}$ as follows:

$$R(W) = RW. \tag{5.19}$$

Revised Energy Function

We assume a similar energy function as in Equation (5.13). Specifically, we will assume the weight matrix W as a third-order tensor of dimension $V \times H \times S$. Thus, the revised function takes the form of:

$$E(\mathbf{x}, \mathbf{h}, \mathbf{r}) = \sum_{s=0}^{S-1} \sum_{j=0}^{H-1} \sum_{k=0}^{V-1} r_s \left(-x_k h_j w_{jks} - b_j h_j - c_k x_k\right). \tag{5.20}$$

In this formulation, a new binary vector $\mathbf{r} \in \{0,1\}^S$ is introduced, representing the factor corresponding to the dominant orientation of the inputs $\mathbf{x} \in \mathcal{X}$. This is encoded by setting $r_s = 1$ and $r_t = 0$, $\forall t \neq s$ (one-hot encoding). We formally express this constraint as:

$$\sum_{n=0}^{S-1} r_n = 1. \tag{5.21}$$

In addition, we will say that, if $r_s = 1$, then the dominant orientation is ϕ_s. Consequently, the conditional probabilities in Equations (5.6) and (5.12) are revised accordingly:

$$p(h_j = 1|\mathbf{x}, \mathbf{r}) = \sigma\left(\sum_{s=0}^{S-1} r_s\left(\mathbf{x}^T W_{j\cdot s} + b_j\right)\right), \tag{5.22}$$

$$p(x_k = 1|\mathbf{h}, \mathbf{r}) = \sigma\left(\sum_{s=0}^{S-1} r_s\left(W_{\cdot ks}\mathbf{h} + c_k\right)\right). \tag{5.23}$$

Sharing the Gradients

As discussed in Section 5.3.2, optimising W via Equations (5.22) and (5.23) has the drawback that inputs with a specific dominant orientation will contribute to update *only* the corresponding slice in W. This is similar to splitting the training set $\mathcal{X}$ into several non-overlapping partitions $\mathcal{X}_s$ and train a separated RBM for each of them. To overcome this problem, the contribution of gradient ∇W_s computed on $\mathcal{X}_s$ can be shared across the other slices in W. Therefore, we will apply the gradient sharing step during training as shown in Section 5.3.1 [24].

For sake of clarity, we suppose we have only two rotations, R_0 and R_1, which account for the $0°$ and $180°$ rotations respectively. Since ∇W_0 and ∇W_1 were computed on different portions of the data, namely $\mathcal{X}_0$ and $\mathcal{X}_1$, we want to transfer the contribution of ∇W_1 to ∇W_0 (and vice versa). To do so, we add a rotated version of ∇W_1 by $-180°$ (we denote such a rotation as R_{-1}) to ∇W_0. In this example, we can redefine the gradients as follows:

$$\nabla W_0 := R_0\left(\nabla W_0\right) + R_{-1}\left(\nabla W_1\right) = \nabla W_0 + R_{-1}\left(\nabla W_1\right), \tag{5.24}$$

$$\nabla W_1 := R_0\left(\nabla W_1\right) + R_{-1}\left(\nabla W_0\right) = \nabla W_1 + R_{-1}\left(\nabla W_0\right), \tag{5.25}$$

Using the examples above, the update rules for the third-tensor W can be generalised as follows:

$$\nabla W_s := \sum_{q=0}^{S-1} R_{-q}\left(\nabla W_{m(s,q)}\right). \tag{5.26}$$

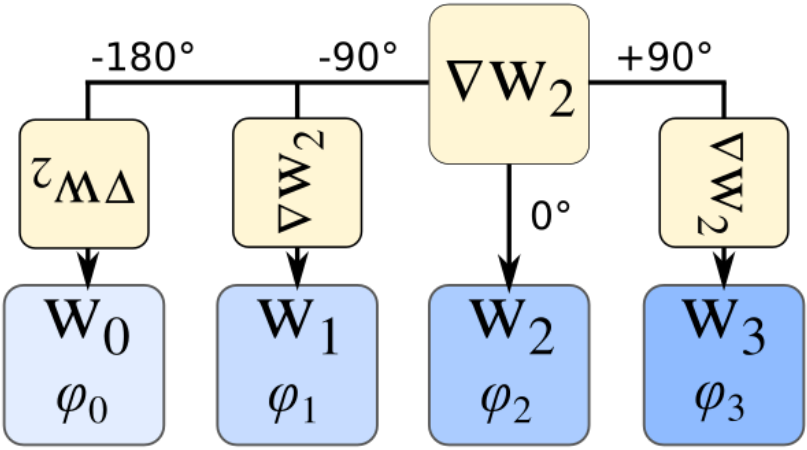


Figure 5.10: Graphic representation of the shared gradient step. Following the example in Figure 5.9, the gradient ∇W_2 for the slice W_2 is computed. By applying proper transformations, rotated versions of ∇W_2 are applied to the other slices in W, as shown above.

A visual example how the shared gradient step is applied is shown in Figure 5.10, in the case of $S = 4$ rotations. The above formulation will be used in a mathematical proof that our method learns rotation-invariant features.

Rotational Equivalence

Taking the examples in Equations (5.24) and (5.25), we can observe, by applying the closure property presented above, the following equivalence:

$$\begin{aligned} R_{-1}(\nabla W_0) &= R_{-1}(R_0(\nabla W_0)) + R_{-1}(R_{-1}(\nabla W_1)) \\ &= R_{m(-1,0)}(\nabla W_0) + R_{m(-1,-1)}(\nabla W_1). \end{aligned}$$

Since $m(-1, 0) = 1$ and $m(-1, -1) = 0$, the above relation becomes:

$$\begin{aligned} R_{-1}(\nabla W_0) &= R_1(\nabla W_0) + R_0(\nabla W_1) \\ &= R_{-1}(\nabla W_0) + R_0(\nabla W_1) = \nabla W_1, \end{aligned} \tag{5.27}$$

where $R_{-1}(\cdot) = R_1(\cdot)$ due to the closure property in Section 5.4.1 (e.g., rotating by $\pm 180^\circ$ produces the same result).

This example with $S = 2$ shows that all gradients of the form ∇W_s are rotated versions of each other. We can generalise this property for Equation (5.26) as follows:

$$R_r(\nabla W_s) := \sum_{q=0}^{S-1} R_{m(r,-q)} \left(\nabla W_{m(s,q)} \right) = \nabla W_{m(s,r)}. \tag{5.28}$$

In order to facilitate the proof of the theorem stating that our approach learns rotation-invariant features, we need the following lemma that makes use of this rotational equivalence.

Lemma 1. *Optimising a third-order tensor $W \in \mathbb{R}^{H \times V \times S}$ as described above for $t > 0$ iterations, then $W_{s'}^{(t)} = R_\kappa(W_s^{(t)})$, with:*

$$\kappa = s' - s. \tag{5.29}$$

Proof. We will proceed by induction over the iteration t. For the base case $t = 0$, we impose that:

i. $\tilde{W} \in R^{H \times V}$ is a matrix initialised with e.g., Glorot Gaussian method [129],[3]

ii. $W_s^{(0)} = R_s(\tilde{W}), \forall s \in \{0, 1, \ldots, S-1\}$.

This means that all the slices in $W^{(0)}$ are initialised as rotated versions of $\tilde{W}$, which initially can be any matrix. Now, let us suppose that the lemma is true until $t - 1$, and prove it for t. Then we have:

[3]We observed that the base case of the induction can be relaxed. Experimental evidence showed that by initialising W with random numbers drawn from a normal distribution it is still possible to have rotation-invariant features.

$$
\begin{aligned}
R_\kappa(W_s^{(t)}) &= \underbrace{R_\kappa\left(W_s^{(t-1)} + \eta\nabla W_s^{(t-1)}\right)}_{\text{From Equation (2.5)}} \\
&= \underbrace{R_\kappa\left(W_s^{(t-1)}\right) + \eta R_\kappa\left(\nabla W_s^{(t-1)}\right)}_{\text{Linearity property (c.f. Section 5.4.1)}} \\
&= \underbrace{W_{s'}^{(t-1)}}_{\text{By induction}} + \eta\underbrace{\nabla W_{m(s,\kappa)}^{(t-1)}}_{\text{From (5.26)}}.
\end{aligned}
\tag{5.30}
$$

At this point, we need to expand the modulo function. Applying (5.29), we obtain that $m(s,\kappa) = (s+\kappa) \bmod S = (s+s'-s) \bmod S = s'$. Thus, to conclude, Equation (5.30) becomes:

$$
R_\kappa(W_s^{(t)}) = W_{s'}^{(t-1)} + \eta\nabla W_{m(s,\kappa)}^{(t-1)} = W_{s'}^{(t-1)} + \eta\nabla W_{s'}^{(t-1)} = W_{s'}^{(t)}.
$$

□

This lemma states that all the slices in the tensor W are rotated versions of each other.

Measuring the Invariance

We adopt the γ-score proposed in [130] to measure invariance. Considering a set of transformations $\mathcal{S}$ and a dataset $\mathcal{X}$, the mean activation of the j-th hidden unit h_j over all the transformations $T \in S$ is computed as:

$$
\mu_j(\mathbf{x}) = \frac{1}{S}\sum_{T\in\mathcal{S}} h_j(T(\mathbf{x})), \tag{5.31}
$$

where $h_j(x) \equiv p(h_j = 1|\mathbf{x},\mathbf{r})$. It is important to note that $\mathbf{r}$ is function of $\mathbf{x}$, hence when the transformation $T(\mathbf{x})$ is applied, the vector $\mathbf{r}$ has to be recomputed accordingly. Then, the γ-score is defined as:

$$\gamma_j = \frac{\text{var}\left\{\mu_j(\mathbf{x})\right\}_{\mathbf{x}\in\mathcal{X}}}{\text{var}\left\{h_j(\mathbf{x})\right\}_{\mathbf{x}\in\mathcal{X}}}. \tag{5.32}$$

We employed the γ-score because it is bounded to the interval $[0, 1]$, where values close to 1 indicate features invariant to the set of transformations $\mathcal{S}$. The γ-score is closely related to the autocorrelation [130] and does not require parameters to be computed, as e.g. the firing threshold in [131].

Proving Rotation Invariance

In this section, we will prove that our model can learn rotation invariant features. Our theorem is based on the hypothesis that the model is trained using the revised energy model and further adaptions showed in Section 5.4.1. The proof shows that the γ-score reaches the highest value (numerator and denominator in Equation (5.32) are equal).

Theorem 1. *Under the hypotheses of Lemma 1 and given a support set $\mathcal{S}$ of S rotations, $\gamma = 1$ for our revised rotation-invariant RBM model.*

Proof. We have to prove that $\gamma_j = 1$, $j = 1, 2, \ldots, H$. From Equation (5.32), we will show that the numerator and denominator coincide. Now, starting from the definition of μ_j showed in Equation (5.31), we obtain:

$$\begin{aligned}
\mu_j(\mathbf{x}) &= \frac{1}{S}\sum_{q=0}^{S-1} h_j(R_q(\mathbf{x})) \\
&= \frac{1}{S}\sum_{q=0}^{S-1} \underbrace{\sigma\left(\sum_{t=0}^{S-1} r_t\left(b_j + R_q(\mathbf{x})^T W_{j,\cdot,t}\right)\right)}_{\text{From Equation (5.22)}} \\
&= \frac{1}{S}\sum_{q=0}^{S-1} \underbrace{\sigma\left(b_j + R_q(\mathbf{x})^T W_{j,\cdot,s'}\right)}_{\text{From (5.21), } \exists s' \text{ s.t. } r_{s'}=1}
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{S}\sum_{q=0}^{S-1} \underbrace{\sigma\left(b_j + R_q(\mathbf{x})^T R_q(W_{j,\cdot,s})\right)}_{\text{From Lemma 1, } \exists s\,:\, R_q(W_s) = W_{s'}} \\
&= \frac{1}{S}\sum_{q=0}^{S-1} \underbrace{\sigma\left(b_j + [R_q\mathbf{x}]^T R_q W_{j,\cdot,s}\right)}_{\text{c.f. Equation (5.19)}} \\
&= \frac{1}{S}\sum_{q=0}^{S-1} \underbrace{\sigma\left(b_j + \mathbf{x}^T R_q^T R_q W_{j,\cdot,s}\right)}_{(AB)^T = B^T A^T} \\
&= \frac{1}{S}\sum_{q=0}^{S-1} \underbrace{\sigma\left(b_j + W_{j,\cdot,s}\mathbf{x}\right)}_{R_q^T R_q = R_q R_q^T = I} \\
&= \frac{1}{S}\sum_{q=0}^{S-1} \sigma\left(\sum_{t=0}^{S-1} r_t\left(b_j + \mathbf{x}^T W_{j,\cdot,t}\right)\right) \\
&= \frac{1}{S}\sum_{q=0}^{S-1} h_j(\mathbf{x}) = h_j(\mathbf{x}).
\end{aligned}
$$

Since $\mu_j(\mathbf{x}) = h_j(\mathbf{x})$, then also their variance over all the samples in the training set is equal. Therefore $\gamma_j = 1$. □

The proof of this theorem shows that our method achieves full invariance. We can make the following remarks about the theorem:

Remark 1. *Theorem 1 does not assume whether* $\mathbf{x}$ *belongs to the training or testing set. In fact, it is valid for any input.*

Remark 2. *Theorem 1 does not make any assumptions how the* $\mathbf{r}$ *vector is computed, as long as Equation* (5.21) *holds.*

Remark 3. *Theorem 1 is a theoretical result and does not account for artefacts due to the discrete nature of inputs and rotations. Empirical computations of the* γ*-score might result in slightly lower values.*

Remark 4. *Theorem 1 is compatible with any typical additional terms that can be added in Equation* (2.5)*, such as momentum and L_2 regulariser [123].*

Remark 5. *Optimising* (5.20) *as in Section 5.4.1, the negative log-likelihood $-\ln p(\mathbf{x}|\Theta)$ is minimised as well.*

5.4.2 Inference of the Dominant Orientation

Here, we describe how to infer the optimal $\mathbf{r}$ vector for an input $\mathbf{x}$ in the dataset. We propose an approach that exploits the intrinsic information learned by the network during training, using the reconstruction error. In our formulation, we can define the reconstruction function $\varphi(\mathbf{x}, \mathbf{r})$ as:

$$
\begin{aligned}
h(\mathbf{x}, \mathbf{r}) &= p(\mathbf{h}|\mathbf{x}, \mathbf{r}), \\
v(\mathbf{h}, \mathbf{r}) &= p(\mathbf{x}|\mathbf{h}, \mathbf{r}),
\end{aligned}
$$

$$
\varphi(\mathbf{x}, \mathbf{r}) = v(h(\mathbf{x}, \mathbf{r}), \mathbf{r}), \tag{5.33}
$$

We define the dominant orientation for an input $\mathbf{x}$ as the one that minimises the following function:

$$
\begin{aligned}
&\hat{s} = \operatorname{argmin}_s \|\varphi(\mathbf{x}, \mathbf{r}) - \mathbf{x}\|_2^2, \\
&\text{such that} \quad r_t = 0 \text{ and } r_s = 1, s \neq t.
\end{aligned} \tag{5.34}
$$

Thus, the corresponding $\mathbf{r}$ for the input $\mathbf{x}$ is $r_{\hat{s}} = 1$ and $r_t = 0, t \neq \hat{s}$. This satisfies the one-hot encoding constraint in Equation (5.21). The optimization of Equation (5.34) can be easily computed for all the possible values of $\mathbf{r}$, as it comes automatically during the forward pass of the training process.

Implementation Details

Training. Our training algorithm is an extended version of the *Contrastive Divergence* [23]. As discussed in the previous section, the core part of our architecture is the inference of the dominant orientation. For each minibatch $\mathcal{B}$, we compute the reconstructed input $\tilde{\mathbf{x}}$ using all weight matrices in W. For each image in $\mathcal{B}$, the dominant orientation is inferred, by selecting the weight matrix that better reconstruct the input (c.f. Figure 5.9). Then, the gradients to update the parameters Θ are computed and the contribution of all the $\nabla W^{(s)}$ is shared with the other weight matrices. Other gradients coming from e.g. sparsity regulariser [123], the KL-Divergence in Equation (5.35), or momentum are also used to update the parameters of the network. Details are shown in Algorithm 1.

Testing. At inference time, an input image is provided to the network to obtain activations using all the weight matrices (one per each orientation). The hidden layer activations produced by the weight matrix minimising the reconstruction error are selected. The chosen activations represent the features of the input image. This is what is also shown in Figure 5.9.

KL-Divergence Regularisation Term to Improve Dominant Orientation Inference

The rotation estimation approach may potentially assign most inputs to one dominant orientation. To avoid this, we opted to regularise the training process, by forcing a prior on the distribution of orientations across the dataset. We achieve this by minimising the following Kullback-Leibler divergence:

$$D_{KL}(\mathbf{p}||\bar{\mathbf{r}}) = \lambda_r \sum_{s=0}^{S-1} p_s \ln \frac{p_s}{\bar{r}_s}, \tag{5.35}$$

where $\mathbf{p}$ is a prior distribution, $\bar{\mathbf{r}}$ is the average assignment of the dominant orientation of the images in the training set, as discussed

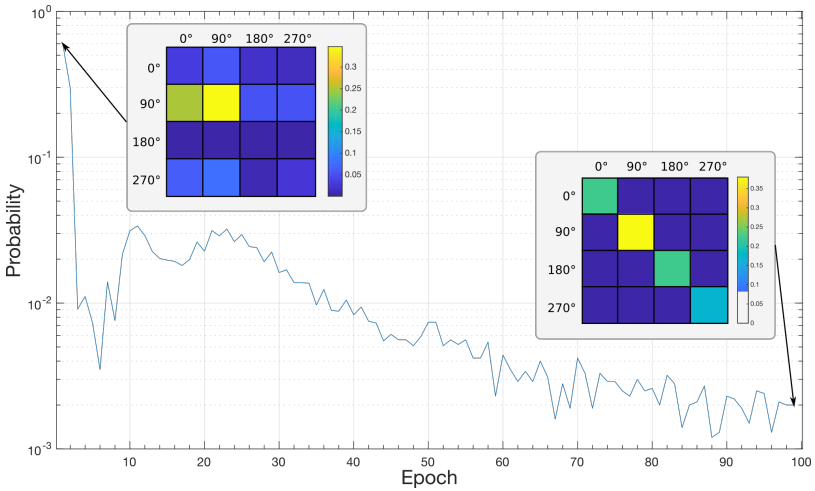


Figure 5.11: Probability (in log-scale) of a prediction change of the dominant orientation occurring during training on the *mnist-rot* dataset (*change probability*). The stability of our inference method increases exponentially over time. We also show two inset transition matrices: the left-hand side inset shows the amount of misclassification between the first two epochs, whereas the right-hand side for the last two epochs.

in Section 5.4.2, and λ_r is a positive constant weighing the strength of the regulariser. Following [132], we compute the average prediction vector $\bar{\mathbf{r}}$ over a mini-batch, rather than the whole training set.

Consistency Analysis

To evaluate the stability of our inference process, we instead measure the consistency of predictions, computing what we define the *change probability*. During training, we tracked the predictions made by our algorithm to infer the dominant orientation of each image in the training set. Then, we analysed how many times each image has been assigned to a dominant orientation over time. We computed

the probability at each epoch that an assignment change occurs and we plotted the result in Figure 5.11.

Algorithm 1: Training procedure of our proposed rotation-invariant Restricted Boltzmann Machine.

Data: Training set $\mathcal{X}$, parameters $\Theta = \{b_j, c_k, w_{jks}\}$
Result: Updated parameters Θ

begin
  for *e:= 1 to MaxEpochs* **do**
    foreach *mini-batch* $\mathcal{B} \subset \mathcal{X}$ **do**
      Perform a forward step with $\mathcal{B}$ of the network and find $\hat{s}$ by minimising Equation (5.34).
      Compute the gradient for the slice $\nabla W^{(\hat{s})}$, as well as ∇c, and ∇b as in Equations (5.8) to (5.10).
      foreach $t \in \{0, \ldots, S-1\} \setminus \{\hat{s}\}$ **do**
        Share the gradient of $\nabla W^{(\hat{s})}$ to $\nabla W^{(t)}$.
      end
      Computes the gradient ∇D_{KL} from the regulariser Equation (5.35) w.r.t. $W^{(\hat{s})}$.
      Apply the update rule in Equation (2.5) to all the parameters in Θ using $\nabla W^{(\hat{s})}$, the shared gradients from all $\nabla W^{(t)}$.
      Update all the parameters in Θ with any other gradients coming from momentum or regulariser(s) (e.g., Equation (5.35)).
    end
  end
end

It can be observed that our inference method stabilises in less than 10 epochs, becoming very consistent in ≈ 40 epochs (probability of a reassignment is very close to 0). Furthermore, Figure 5.11 contains two inset transition matrices, where we show how assignments are redistributed between two epochs. Specifically, the left-

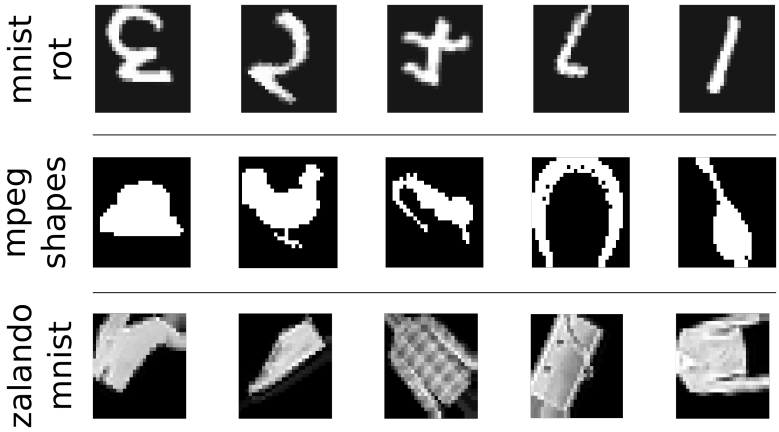


Figure 5.12: Sample images of the employed datasets. *Top row*: mnist-rot [8]. *Middle row:* MPEG-7 Shape Silhouette database [10]. *Bottom row:* rotated version of the zalando fashion mnist dataset (original from [11]).

hand side transition matrix show reassignments occurring between the first two epochs, whereas the right-hand side inset shows the changes occurring in the last two epochs. These two plots show that, although the initial predictions of the dominant orientation are very unstable, the network is able to automatically assign each image to the proper class of orientation.

5.4.3 Experimental Results

We demonstrate our model on the following datasets: *mnist-rot* [8], the *MPEG-7 Shape Silhouette database* [10], and a rotated version of the *zalando fashion-mnist* dataset [11] (c.f. Figure 5.12). We compared our performance with the following approaches:

- *Support Vector Machine [22]:* SVM trained directly on the data, without preprocessing.

- *RBM [23]:* the original model will provide a baseline result for our experiments.
- *TI-RBM [21]:* state-of-the-art method for learning transformation-invariant features. Specifically, we only used rotations as transformations.
- *ERI-RBM [24]* (c.f. Section 5.3).

Parameters

If not otherwise stated, we run our experiments using the following parameters, which were set the same for all methods. We trained RBMs with $H = 500$ hidden units for 100 epochs, using a learning rate $\eta = 0.003$. We also adopted a sparsity regulariser target $p = 0.1$ with regularisation constant $\lambda = 0.003$ [123].[4] Furthermore, for the regulariser in Equation (5.35), we set the constant $\lambda_r = 100$ and the prior probability distribution $\mathbf{p} = \mathcal{U}(0, S-1)$. We used a momentum $\alpha = 0.5$ for the first 5 epochs, then we increased it to $\alpha = 0.9$. To avoid nuisance due to pixel interpolation, we run our experiment setting the number of rotations $S = 4$, as the angle set Φ contains only multiples of $90°$. We initialised the weight matrices using the *Glorot* method [129], using random numbers sampled from a Gaussian distribution with zero mean and standard deviation of $\sqrt{2/(V+H)}$, where bias terms are initialised with 0s. For classification, we followed the same protocol as in [24], adopting SVM [22] with an RBF kernel. For the classifier, we set the spread parameter $\sigma = 0.002$. The loss parameter C is set accordingly for each dataset. Experiments were repeated 5 times, with different initialisation, and mean and standard deviation were computed.

Experiments

Tests on mnist-rot [8]. This well-known benchmark dataset contains 10,000 images for training and 50,000 for testing of hand-

[4] As reported in [123], we only update the bias of the hidden units $\mathbf{b}$.

Table 5.2: Test accuracy of our method compared with SVM [22], the original formulation of RBM [23], the transformation-invariant RBM [21], and the explicit rotation-invariant RBM [24] on three different datasets. We report *best result (mean ± std)* [25] of 5 random initialisations.

Method	mnist-rot [8]	MPEG-7 [10]	zalando mnist-rot
SVM [22]	89.36%	82.00%	74.71%
RBM [23]	80.18% (79.91% ± 1.96)	78.57% (78.28% ± 0.27)	62.99% (62.92% ± 0.10)
TI-RBM [21]	89.90% (89.75% ± 0.16)	76.14% (75.14% ± 0.57)	76.54% (76.42% ± 0.12)
ERI-RBM [24]	90.61% (90.48% ± 0.13)	73.00% (72.52% ± 0.25)	74.84% (74.49% ± 0.18)
Ours	**92.12%** (91.81% ± 0.30)	**85.71**% (83.43% ± 1.38)	**77.14**% (76.94% ± 0.24)

written digits (c.f. top row in Figure 5.12). For training, we adopted the parameters discussed in page 92, setting $C = 10$ for the classifier.

Table 5.2 shows the results of our experiments. It can be observed that TI-RBM and ERI-RBM perform similarly on this setup, outperforming the baseline by $\approx 10\%$. Our proposed method obtains the best performance, achieving more than 92% test accuracy. This results shows that our method of inferring rotations is more reliable than e.g., max-pooling across all rotations [21], or relying on exogenous methods [24]. Then, we empirically computed the γ-score as described in Equation (5.32) and our method scored $\gamma = 0.98$, as expected from Theorem 1. In Figure 5.8, we show a subset of the filters learned by our method and, as it can be observed, filters are rotated versions of each other, providing experimental evidence for Lemma 1.

We compared also with the *Contractive Autoencoder* on the same dataset [133]. Their method minimises a regularisation term based on the Jacobian matrix of the encoder step of the network, which learns invariant features. Their results with 1,000 hidden units showed a smaller test accuracy of 90.34%. Therefore, our method learns high discriminative features with half of the hidden units, thus learning a more compact representation.

Tests on a small training set. In this experiment, we want to demonstrate that our method learns robust features even when trained on a small dataset. We used the *MPEG-7 Shape Silhouette database* [10], containing 1,400 images belonging to 70 categories (samples depicted in Figure 5.12). Since the images have a variable size, we cropped and resized them to 28×28 pixels. We randomly split the dataset into 700 images for training and 700 for testing, maintaining class balance. In this case, we set the loss parameter for SVM $C = 100$.

Results on this dataset are also reported in Table 5.2. Our method outperforms all other approaches, reducing the testing error by $\approx 10\%$. Specifically, we can observe that TI-RBM [21] and ERI-RBM [24] suffer from lack of data in the training set, obtaining a testing error even higher than the RBM baseline. On the other hand, RBM is not able to accommodate the rotational variance in this dataset, causing it to perform poorly compared with our approach. Therefore, our method can learn better rotation-invariant features also in the case of a reduced training set.

Tests on the rotated zalando fashion mnist dataset. We also tested our approach on a customised version of the *zalando fashion mnist* dataset [11]. Specifically, the original dataset contains images of 10 categories of clothes. Images are grayscale and 28×28 pixels size.[5] For these experiments, we generated a rotated version of the dataset, using uniformly distributed random rotations. To create this customized dataset, we adopted the original code from [8] used to generate *mnist-rot*.[6] We generated 10,000 images for training and 50,000 for testing [8]. To the best of our knowledge, an equivalent *mnist-rot* for the *zalando fashion mnist* has not been created yet. We refer to such a generated dataset as *zalando fashion mnist-rot*. Results

[5]Further details on the *zalando fashion mnist* at `https://github.com/zalandoresearch/fashion-mnist`.

[6]Available at `http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/DeepVsShallowComparisonICML2007`.

Table 5.3: Ablation results showing that our method benefits from shared gradients and the regulariser presented in Equation (5.35). We also report the upper bound (ours trained with actual rotations) values for all the applicable datasets. (We could not perform this on the *MPEG-7 database* [10], as the images are already transformed and their rotations are unknown).

Method	mnist-rot [8]	MPEG-7 [10]	zalando mnist-rot
RBM [23] (reference)	80.18%	78.57%	62.99%
Ours without sharing gradients and Equation (5.35)	53.80%	68.71%	56.76%
Ours without Equation (5.35)	91.96%	82.14%	76.21%
Ours (proposed method)	92.12%	85.71%	77.14%
Upper bound (ours with actual rotations)	93.96%	N/A	79.38%

on this version of the *zalando* dataset are also shown in Table 5.2.[7] Overall, we can observe that our method outperforms all the other approaches on this dataset as well.

Discussion. From our experiments, it appears that it is better to rely on the intrinsic information encoded in the network to infer the dominant orientation. In this way, our model explicitly cancels out the nuisance given by rotations, producing fully rotation-invariant image representations. We demonstrated that our approach is better than marginalising across all possible rotations, as it happens in TI-RBM [21], or using an exogenous process to estimate orientations, as in ERI-RBM [24]. In addition, we showed that our method works particularly well in datasets with small size.

[7] We also trained our model on the original *zalando fashion mnist* dataset [11] and it performed similarly to an RBM (84.55% vs 85.99% for ours and RBM respectively). Furthermore, our method had a higher accuracy compared with TI-RBM and ERI-RBM. Thus, our method works well when there are no rotations present in the training set.

Ablation Experiments

We want to assess how our approach benefits from the gradient sharing step [24] and the KL-Divergence based regulariser shown in Equation (5.35). Experiments were performed using the same protocol as discussed in the previous sections. We show the result of our experiments in Table 5.3.

To establish a reference baseline, we trained the original RBM model [23]. Next, we trained our model disabling sharing gradients and Equation (5.35). In this case, it can be observed that our model has lower performance compared with the baseline. Training our network without shared gradients is similar to training S different RBMs, such as the s-th model is trained with only the $\mathcal{X}_s$ partition of the data. This means that the training set $\mathcal{X}$ is split and each RBM is trained independently on a smaller portion of the data. This procedure is closely related to the *Oriented RBM* baseline method described in [24]. By enabling the shared gradients, the performance of our approach improves by 20% (even $\approx 40\%$ on *mnist-rot*), showing that this technique is effectively improving the training. The gradient sharing step also promotes the learning or rotation-invariant features (see Theorem 1), thanks to the rotation equivalence property in Equation (5.28) . When the regulariser in Equation (5.35) is also enabled during training, performance improves further, as the inference of the dominant orientation becomes more robust and reliable.

Robustness of Orientation Estimation

We want to assess whether errors in estimating rotation during training can 'self-correct' and still lead to reasonable weight matrices (i.e. achieve good test accuracy).

We train our architecture but every 20 epochs we randomly perturb a portion of the rotations of the minibatch. To see if they can 'self-correct', we computed the *change probability* also in this case. Figure 5.13 shows that perturbing the orientation estimate

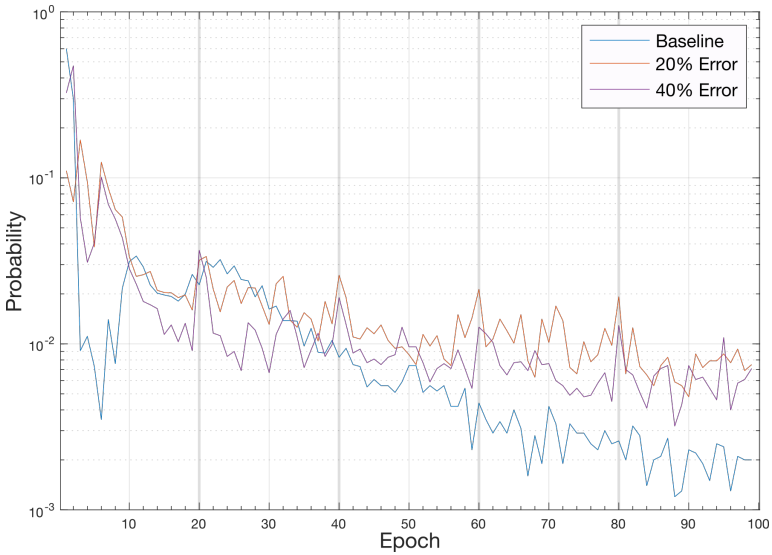


Figure 5.13: Probability (in log-scale) that a change in inferring the dominant orientation occurs during training. Epochs 20, 40, 60, and 80 are marked with a thicker grey line, highlighting the times which random errors are introduced in the training. Overall, the injection of noise at regular interval does not affect the learning process. The probability of a change in inferring the dominant orientation decreases asymptotically (c.f. Figure 5.11). We plot baseline (no changes), 20%, and 40% for brevity.

has minimal effect on the learning of the orientations (and weight matrices) as they quickly recover and converge to the same low probability values.

To see whether orientation perturbation during training affects the performance of the final weight matrices, we evaluate their testing accuracy on *mnist-rot*. As Table 5.4 shows, there is minimal performance decrease even at high portions of perturbation (different initialisations were used). As an example, when 50% of the images are affected by wrong dominant orientations, the loss of

Table 5.4: Testing accuracy on *mnist-rot* dataset of our architecture trained by altering the inference of the dominant orientation at gradually increasing portions of the training set.

	0%	10%	20%	30%	40%	50%
Accuracy	92.12%	91.76%	91.93%	91.50%	91.96%	90.96%

testing accuracy is $\approx 1\%$ w.r.t. the baseline (c.f. Table 5.4). Thus our method of inferring dominant orientation is robust and consistent even in the case of noisy rotation estimates (in line with recent literature in learning with noisy labels [132, 134]).

5.5 Experiments with Plant Data

In this section, we apply the rotation-invariant Restricted Boltzmann Machine on plant data of the CVPPP dataset. We follow the pipeline presented in Chapter 4, replacing the K-means step.

Specifically, we keep the original image, instead of transforming it into the log-polar representation. We extract patches from the Cartesian space and train the ERI-RBM discussed in Section 5.4. A holistic per-image descriptor is obtained with a sum-pooling. Image features are then used to train the SVR model for the leaf prediction.

5.5.1 Implementation Details

We used the A1 images from the CVPPP 2017 dataset, rescaling them to 100×100 pixels in size after removing the background. Then, we extracted 15×15 patches from each image in the dataset, randomly sampling only the 25% of them [103]. Before training the rotation-invariant RBM, we normalised the set of patches by subtracting the mean and dividing by the standard deviation.

We trained the ERI-RBM in Section 5.3 for 100 epochs, with 500 hidden units, $S = 4$ rotations, and a learning rate $\eta = 0.001$. For

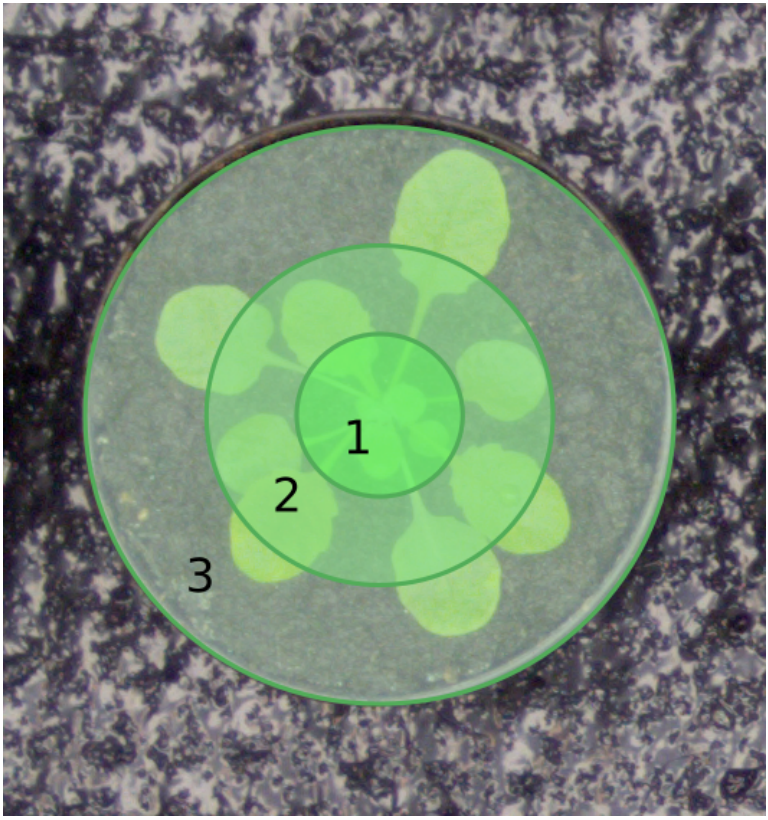


Figure 5.14: Three concentric pooling regions

this experiment, we used the Gaussian-Bernoulli formulation of the RBM to learn from real-valued data [135].

After training the ERI-RBM, we encoded images from the training and testing set with an holistic descriptor. Specifically, patches were extracted from the images and normalised. Then, each patch was provided to the trained ERI-RBM model to obtain a local descriptor. Similarly to Section 4.1.4, we split the plant image into three concentric circular pooling regions, as depicted in Figure 5.14. Patch features within each pooling region were summed (sum-pooling)

and the three resulting pooled features were concatenated together to obtain a holistic plant descriptor.

5.5.2 Results and Discussion

This proof-of-concept test on real data showed a higher MSE w.r.t to the results in Table 4.4. Specifically, the GLC (c.f. Chapter 4) has an MSE $= 2.9$ (c.f. Section 2.4) on A1, whereas this setup has an MSE $= 3.1$, showing slightly worse performance, compared to the approach with log-polar representation and K-means.

This experiment shows a lack of potential for shallow networks, particularly RBMs, to learn a good representation from real-valued data. Therefore, this inspired us to employ deep learning as a next step. However, deep neural networks require the learning of many parameters, requiring large labelled datasets. In the next part of this thesis, strategies to collect more labelled plant datasets are shown through three different approaches.

Part III

Strategies to Obtain More Labelled Data

6

Annotation Tool to Assist Experts

In the last chapter, we showed that shallow models cannot well accommodate continuous data, performing (in the case of plant phenotyping) slightly worse than our previous approach in [7]. This result motivated us to employ deep neural networks for plant phenotyping. Compared with shallow models, deep architectures require more data and computational power, as discussed in Section 2.3.3.

Although the power of deep learning is undeniable, the employment of such a tool is not easy in plant phenotyping, due to the lack of training data. For instance, the availability of labelled datasets for leaf counting is rather limited. In Table 6.1 we list the current publicly available datasets for plant phenotyping, showing the number of labelled images for leaf counting. Furthermore, the

This chapter is based on:

- M. Minervini, M. V. Giuffrida, and S. A. Tsaftaris. "An interactive tool for semi-automated leaf annotation", *Proceedings of the Computer Vision Problems in Plant Phenotyping (CVPPP)*, pages 6.1-6.13. BMVA Press, 2015.
- M. Minervini, M. V. Giuffrida, P. Perata, and S. A. Tsaftaris. "Phenotiki: an open software and hardware platform for affordable and easy image-based phenotyping of rosette-shaped plants". *Plant Journal*, 2017.

Table 6.1: Currently publicly available labelled datasets for plant phenotyping (with the number of labelled images).

Dataset	Type of plants	Modalities	Labelled Images
CVPPP 2017† [6,12,16]			
A1	*Arabidopsis thaliana* Col-0	RGB	128
A2	*Arabidopsis thaliana* Col-0, ctr, pgm, ein2.1, adh1	RGB	31
A3	*Nicotiana tabacum*	RGB	27
A4	*Arabidopsis thaliana* Col-0	RGB	624
		Total:	*810*
Multi-modal imagery database for plant phenotyping [15]			
	Arabidopsis thaliana Col-0	RGB, FMP, NIR	576
Komatsuna [17]			
	Komatsuna plants	RGB	300

†A separated testing set with undisclosed labels is also available.

multi-modal [15] and Komatsuna dataset [17] do not come with a testing set. This means that the numbers reported in Table 6.1 have to be further decreased to generate validation and testing sets. We can say that we have less than 1.5k images to learn a robust model for leaf counting using deep learning, and trying to avoid overfitting.

However, the acquisition of new (plant) datasets is not easy. Overall, the creation of a training set can be split into three main parts: i) data collection; ii) data processing ; and ii) data annotation. Data collection involves setting up of a plant experiment and constant monitoring of the growth process. Once data are collected, they need to be processed (e.g., each plant should be individually cropped). Lastly, data need annotations. In this chapter (and in the following), we will focus on the last task: how to annotate plant im-

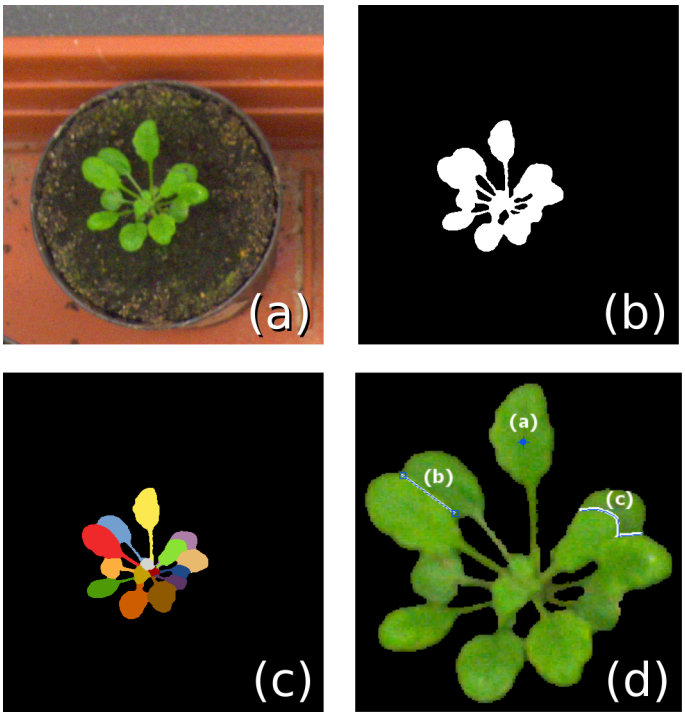


Figure 6.1: Example of a rosette plant (*A. thaliana*) and annotations: (a) Original image, (b) plant segmentation mask, (c) leaf segmentation ground truth, and (d) Zoom-in of (a), showing examples of different types of scribbles that can be used as input to our tool.

ages. Specifically, we will show here an annotation tool developed to assist the plant experts during the annotation process. This tool was then bundled into the *Phenotiki Analysis Software* [3], together with other tools, such as the leaf counting algorithm in [7].

Specifically, the tool we present can be used to semi-automatically segment leaves in images of rosette plants, using scribble annotations (cf. Figure 6.1). A scribble (or even just a dot) per each leaf and a scribble for the background are necessary (although the tool also allows users to include plant masks and exclude background).

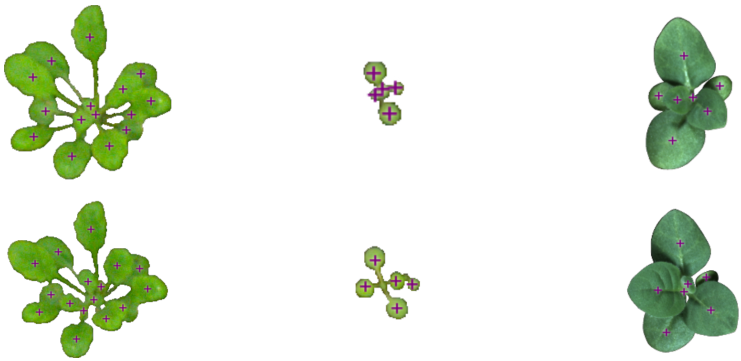


Figure 6.2: Example images (background was removed) of Arabidopsis taken from the A1 (left), and A2 (middle), datasets respectively, and tobacco taken from A3 (right). Rows show the same plant (with leaf center annotations in purple) at different stages of development.

A graph-based interactive segmentation algorithm is employed to spread those seeds across the image, converging towards accurate leaf boundaries [26]. Our implementation permits the user to revisit the segmentation and redraw scribbles until satisfactory results are obtained. We used the CVPPP 2015 dataset [6, 12], achieving an average leaf segmentation accuracy of almost 97% using scribbles as annotations. This tool can be used to annotate data directly for the study of plant and leaf growth or to provide annotated datasets for learning-based approaches to extracting phenotypes from images.

6.1 Proposed Method

The problem of segmenting leaves in an image is summarised in Figure 6.1. Given an input image (Figure 6.1(a)), and possibly a foreground mask (Figure 6.1(b)), the goal is to obtain a segmentation as close as to Figure 6.1(c) with as little interaction as possible. Typically, in a semi-automatic segmentation context user input con-

sists of dots, lines, or scribbles, examples of which are shown in Figure 6.1(d).

Interactive segmentation is an active field of research, with solutions rooted in region growing methods, level set and live-wire approaches [136–138]. The goal is to segment object(s) of interest receiving some guidance by the user. For example, active contour models rely on an initial contour estimate drawn by the user, and the algorithm will aim to converge to an accurate delineation of the object boundaries. However, the performance of such methods depends on the accuracy of the initial contour and typically several parameters need to be tuned, which may complicate user interaction.

Recently, also graph-based approaches have emerged [26, 139]. They are particularly useful since they permit the simultaneous segmentation of multiple overlapping objects in a natural fashion –instead, other approaches usually solve each segmentation individually. In this formulation, users can place annotations (e.g., a few pixels known as scribbles, hints, or seeds) and the algorithm jointly finds a multi-label segmentation. According to choices of energy functionals for constructing the graph and choices of parameters, trade-offs to accuracy, speed, and less user iterations (to correct the annotations and outcome) can be obtained. Interactive segmentation has been used in several contexts of biology and medicine, particularly in the latter for aiding diagnosis [26, 139].

In plant applications most annotation tools are limited towards annotating whole plants (hence, not individual leaves) [40, 111, 140], leaves in isolation (e.g., excised from the actual plant) [141, 142], or under predefined experimental conditions (e.g., using depth or flow information) [143].

Our goal instead is to develop a tool that permits users to delineate multiple leaves with as little interaction as possible. Hence, our tool allows the user to add scribbles inside a plant image, which are then provided to the segmentation algorithm [26] as seeds, i.e., manually labelled pixels. We require to add at least a seed pixel (i.e., a

dot) per leaf, to ensure that the algorithm is aware of the exact number of objects present in an image. In addition, our tool also permits the user to input a plant mask, which delineates plant from background given recent progress in plant segmentation [40, 111, 140]. Giving such mask as input, background pixels are used as an additional annotation and facilitate the algorithm to not over-segment towards non-plant material (such as the soil, the pot, or other external objects).

In the next two subsections we first review the graph-based algorithm we use in our implementation, and then we present the workflow and user interface of the proposed tool. We test its performance using the datasets provided for the *Leaf Segmentation Challenge* (LSC) and *Leaf Counting Challenge* (LCC). In the first row in Figure 6.2 we present three specimens drawn from A1, A2, and A3 dataset respectively. In the second row we report the same subjects after a few days of growth, showing the variability in terms of leaf shape and arrangement.

6.1.1 Interactive Segmentation with Random Walks

The segmentation algorithm adopted here relies on a graph-based representation of the image, where each pixel is a node and neighboring pixels are connected by weighted edges. Additionally, nodes may be marked as *seeds* and are associated with pixel labels, representing user-provided prior knowledge (i.e., the scribbles). To propagate object labels to unseeded pixels we use the random walks algorithm proposed in [26].

The concept of random walks on graphs has been developed in probability theory [144], and applied to several scientific fields, such as biology, physics, chemistry, or the social sciences [145]. In its basic formulation, a so-called *random walker* can move in a 1-D discrete space one step either to the right of its current position with probability p or to the left with probability $q = 1 - p$ (Figure 6.3). One-dimensional random walks have been studied extensively, also due to their interpretation in terms of discrete-space and discrete-

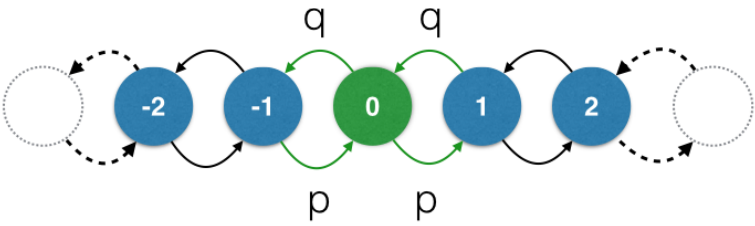


Figure 6.3: Example of 1-dimensional discrete random walk. The probability p defines the likelihood of a random walker to go right ($q = 1 - p$ is for leftwards steps).

time Markov chains [146]. Extensions to higher dimensions are based on graph representations [147].

The segmentation algorithm proposed by Grady [26] aims at partitioning a given image into a set of K labels. It relies on labelled pixels (seeds) for each of the K partitions of the image. The input image I is represented as a graph $G = (V, E)$, where a node $v \in V$ corresponds to a pixel in I, and $e_{ij} \in E \subseteq V \times V$ denotes an edge in the graph connecting two nodes. The goal is to compute the probability x_i^s that a random walker starting from $v_i \in V$ reaches the seed s, where $s = 1, \ldots, K$. A segmentation is obtained by assigning to the node (pixel) v_i the label $s^* = \text{argmax}_s(x_j^s)$ associated with the highest probability.

In this context, G is an undirected weighted graph, and a weight $w_{ij} > 0$ corresponds to the likelihood that a random walker will move from v_i to v_j. Following the original formulation of Grady, we set the weights using a function that evaluates difference of intensities among pixels in the image:

$$w_{ij} = \exp\left(-\beta \|g_i - g_j\|_2^2\right), \tag{6.1}$$

where g_i defines the (possibly vector-valued) pixel intensity of v_i. Parameter β controls the sensitivity to small differences in intensity. Note that other functions are possible, which need not rely solely on gradients of local intensity. Additionally, spatial coherence can

also be enforced by including in Equation (6.1) a term accounting for distance among pixel locations.

Simulating the random walker to compute the probabilities x_i^s would be computationally inefficient. Thanks to known connections between random walks and circuit theory, probabilities are instead obtained by solving a *combinatorial Dirichlet problem* [148]. Let L be the combinatorial Laplacian matrix [149],

$$L_{ij} = \begin{cases} d_i & \text{if } i = j, \\ -w_{ij} & \text{if } e_{ij} \in E, \\ 0 & \text{otherwise,} \end{cases} \tag{6.2}$$

where $d_i = \sum_{k \in V} w_{ik}$ is the degree of v_i. Also, let A be the incidence matrix, defined as:

$$A_{e_{ij} v_k} = \begin{cases} 1 & \text{if } i = k, \\ -1 & \text{if } j = k, \\ 0 & \text{otherwise,} \end{cases} \tag{6.3}$$

where A has size $|E| \times |V|$ and is indexed for each edge in E and each node in V. The constitutive matrix C is an $|E| \times |E|$ diagonal matrix with the weights w_{ij} along the main diagonal. The combinatorial formulation of the corresponding Dirichlet problem is:

$$D[x] = \frac{1}{2}(Ax)^{\mathsf{T}} C(Ax) = \frac{1}{2} x^{\mathsf{T}} L x. \tag{6.4}$$

The set of vertices V can be partitioned as $V = V_M \cup V_U$, with V_M containing seed nodes, and V_U containing the remaining unmarked nodes. Assuming that nodes in L are sorted such that seed vertices are before unmarked vertices, Equation (6.4) can be rewritten as follows:

$$\begin{aligned} D[x_U] &= \frac{1}{2} \begin{bmatrix} x_M^{\mathsf{T}} x_U^{\mathsf{T}} \end{bmatrix} \begin{bmatrix} L_M & B \\ B^{\mathsf{T}} & L_U \end{bmatrix} \begin{bmatrix} x_M \\ x_U \end{bmatrix} \\ &= \frac{1}{2} \left(x_M^{\mathsf{T}} L_M x_M + 2 x_U^{\mathsf{T}} B^{\mathsf{T}} x_M + x_U^{\mathsf{T}} L_U x_U \right). \end{aligned} \tag{6.5}$$

Figure 6.4: Graphical user interface of our annotation tool. In this example, we add as many seeds as the number of leaves and the underlying random walks segmentation algorithm is able to propagate those seeds and provide an accurate leaf delineation.

Because of its construction (cf. Equation (6.2)), the combinatorial Laplacian matrix L is positive semidefinite and the critical points in Equation (6.5) are local minima. Differentiating (6.5) by x_U, the solution that minimizes the Dirichlet problem has to be sought in the following system of linear equations with $|V_U|$ unknowns:

$$L_U x_U = -B^{\mathsf{T}} x_M, \tag{6.6}$$

which is guaranteed to have a solution if every connected component contains a node in V_M, namely a seed vertex.

6.2 The Annotation Tool

Our software tool provides a user-friendly graphical user interface (GUI) to support interactively experts during the leaf annotation process. Figure 6.4 shows an example of leaf segmentation obtained using our tool.

The typical workflow of the annotation process is described in the following paragraphs. First, we load an RGB image of a plant,

which is internally converted to the CIE *L*a*b** color space. The software offers the possibility to eliminate background by applying a plant segmentation mask loaded from file (e.g., obtained by manual delineation or via automated segmentation approaches [111]). Next, expert annotation is performed using the available *Annotation tools*. We permit four types of pixel annotations (examples are shown in Figure 6.1(d)): (i) dot, (ii) line, (iii) freehand, and (iv) annotations loaded from file. The *dot tool* allows to mark a single pixel as seed. The *line tool* allows to annotate a set of pixels laying on a line segment. The *freehand tool* allows to draw free-form curves, useful e.g. to annotate highly overlapping leaves. Our tool also allows to read annotations from file, in the form of a binary image where leaf center coordinates are denoted by '1' and '0' otherwise, or a binary image containing as many 'disconnected' scribbles as pixel drawings with value '1' and '0' otherwise. Zooming and panning facilitate the annotation of complicated regions. We assign to each annotation a different label from 1 to K, where K is the total number of objects (leaves in our case plus background) to segment. When the annotations are loaded, we press on the *Segment* button and the final result is displayed in the right-hand side of the interface (see Figure 6.4). A slider allows to set the parameter β (Equation (6.1)) to control the sensitivity of the segmentation algorithm to variations in intensity between neighbouring pixels. Notice that this represents the only free parameter tunable by the user.

The GUI offers two visualisation modalities, showing either the leaf labelling (cf. Figure 6.4) or the segmentation outline overlaid on the original RGB image (not shown for brevity). The leaf segmentation mask can be exported as an indices PNG image file, where '0' denotes background and all subsequent indices denote leaf labels. A colour palette is embedded in the PNG file to map indexes to displayed colour, maximising contrast and improving legibility. The GUI allows also to load a sequence of images and can split the visualisation screen to see results of prior labelling of the sequence.

In case of unsatisfactory results, the *Cancel* button cleans the image from all the scribbles, and the user can repeat the annotation

procedure. Optionally, annotations can be moved after they are placed and the last one can be deleted by pressing the `Ctrl+Z` key combination, to enable iterative segmentation and refinement.

6.3 Experimental Results

In the following we show some results and evaluations using the proposed tool. We run the software on Matlab R2014b and a PC equipped with Intel Core i7-4710HQ CPU 2.50 GHz, 16 GB memory, and running 64-bit GNU/Linux. To quantitatively evaluate our tool we use training datasets from Arabidopsis (A1, A2) and tobacco (A3) [12], in the context of the *Leaf Segmentation and Counting Challenges* (LSC & LCC), held in conjunction with the *Computer Vision Problems in Plant Phenotyping* (CVPPP 2015) workshop.[1] Example images and leaf centre annotations are shown in Figure 6.2.

We use datasets and available ground truth (dot annotations available in LCC and actual leaf segmentations) to test the performance of our tool using a variety of automatically computed input seeds and evaluate the segmentation outcome w.r.t. ground truth manual segmentation available in LSC. To simulate user interaction (scribbles), we devise three types of annotations per each leaf: *dot*, as available in the LCC training set; *line*, a single line obtained by finding the principal direction of each leaf mask; and *skeleton*, obtained by skeletonising each leaf mask. It is obvious that they simulate scenarios of less to more complex (and laborious) scribble annotations. Examples of automatically obtained annotations and segmentations are shown in Figure 6.5.

As an evaluation criterion we adopt the *Symmetric Best Dice* (%), as defined in Section 2.4 on page 28, which measures average Dice overlap between leaf objects in ground truth and algorithmic result. We use 128, 31, and 27 images from datasets A1, A2, and A3, respectively. In the random walk segmentation algorithm, we use $\beta = 35$ for A1, $\beta = 10$ for A2, and $\beta = 70$ for A3.

[1] `http://www.plant-phenotyping.org/CVPPP2015`

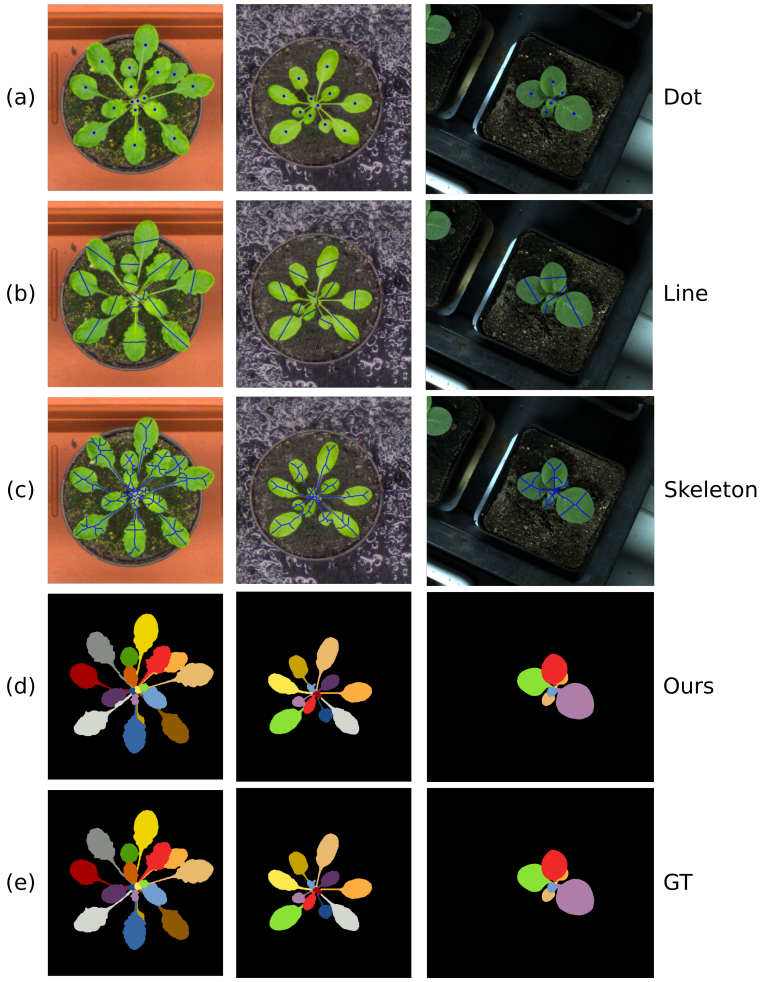


Figure 6.5: (a)-(c) Example annotations used to evaluate the random walks segmentation algorithm. (d) Leaf segmentation obtained using the annotations in (c), which matches almost perfectly the (e) ground truth leaf mask (obtained by manually labelling each pixel [12]).

Table 6.2: Leaf segmentation accuracy (*SymmetricBestDice*, expressed as %) obtained by the random walks segmentation approach [26] adopted here, using different types of leaf annotations. Ref. [20] is adopted as baseline. Results are shown as *mean* $\pm$ *standard deviation*.

		Proposed approach		
	Ref. [20]	Dot	Line	Skeleton
A1	74.2±7.7	89.9±2.8	89.7±2.6	96.8±0.9
A2	80.6±8.7	87.6±8.0	89.0±4.7	96.8±1.5
A3	61.8±19.1	75.9±22.4	88.7±5.9	96.4±2.0
All	73.5±11.5	84.5±11.1	89.1±4.4	96.7±1.5

Table 6.2 shows quantitative findings obtained by evaluating the random walks segmentation algorithm on Arabidopsis and tobacco image data. Overall, we observe that by increasing the level of detail in the annotations, leaf segmentation accuracy increases accordingly. Even when relying only on the simplest form of leaf annotation (dots), our approach outperforms the state-of-the-art automated leaf segmentation obtained by Pape and Klukas [20]. When using a straight line as an annotation, performance increases slightly on overage but consistency improves significantly (lower standard deviation w.r.t. using the dot annotation). Using skeleton-like scribbles, our tool obtains almost perfect and consistent leaf segmentation (*SymmetricBestDice* $\approx$ 97%). Our tool can be used to annotate leaves for extracting phenotyping information, since currently state-of-the-art performance in automated leaf segmentation is rather limited, but also to create annotations for obtaining ground truth segmentations.

Annotating a plant image by simply clicking on leaves (i.e., using dot annotations) is a simple task that typically requires less than a minute (based on our own albeit biased experience). Once the

seeds are placed, the leaf segmentation is calculated and displayed within few seconds. On average it takes 0.3 seconds to process an image from A1 or A2 (0.25 megapixels) and 6.2 seconds for the higher-resolution images in A3 (5 megapixels). On the other hand, annotating leaves on the same plant images adopting a completely manual approach is considerably more time consuming. We measure the performance of three different experts (i.e., already familiar with the task and the image data), observing that it requires 20 minutes or more, depending on plant complexity, to complete the leaf annotation of a single plant image, even assuming a plant mask.

6.4 Integration on Phenotiki

The development of this semi-automatic tool inspired us to gather all the image analysis algorithms that our research group has developed (some of them discussed in this thesis) into a single software. Therefore, we designed a software that allowed to: a) segment arabidopsis plant and extract geometrical information, using the level-set algorithm in [28]; b) annotate leaves using the annotation tool just described; c) count leaves using the algorithm in Chapter 4; and d) export the collected data.

However, Phenotiki is not just an analysis software. As displayed in Figure 6.6, it also includes an affordable system for plant phenotyping that, relying on off-the-shelf parts, provides an easy to install and maintain platform, offering an out-of-box experience for a well-established phenotyping need: imaging rosette plants. The analysis software processes data originating from our device seamlessly and automatically. Our affordable device ($\sim$ 200€) can be deployed in growth chambers or greenhouse to acquire optical 2D images of approximately up to 60 adult Arabidopsis rosettes concurrently.

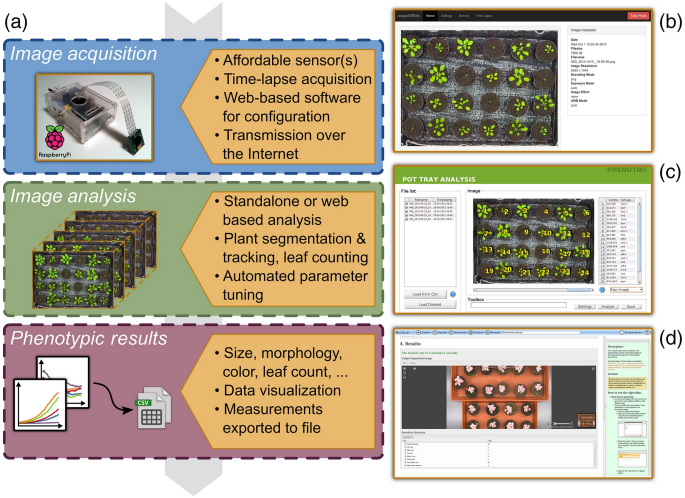


Figure 6.6: Overview of the Phenotiki system and screen captures showing the graphical user interfaces to operate its hardware and software components. (a) Schematic of the proposed distributed sensing and analysis framework illustrating the main components of our phenotyping platform. (b) Web interface to configure and operate the Phenotiki device from the browser. (c) Stand-alone version of the image-analysis software. (d) Cloud-based version of the image analysis software that runs on a web browser.

6.4.1 Sensor Specifications

The Phenotiki sensor consists of a Raspberry Pi embedded computer (`http://www.raspberrypi.org`) operating the RaspiCam fixed-focus (and fixed-zoom) imaging sensor. As Figure 6.7 shows, the device is small ($10 \times 6.5 \times 3.5$ cm) and lightweight (115 g) and could wirelessly connect to the Internet after it was set up. We devised graphical software (c.f. Figure 6.6(b)) for ease of interaction with the device, which allowed us to define an acquisition schedule and parameters for time-lapse 2D optical imaging of the scene and data transmission.

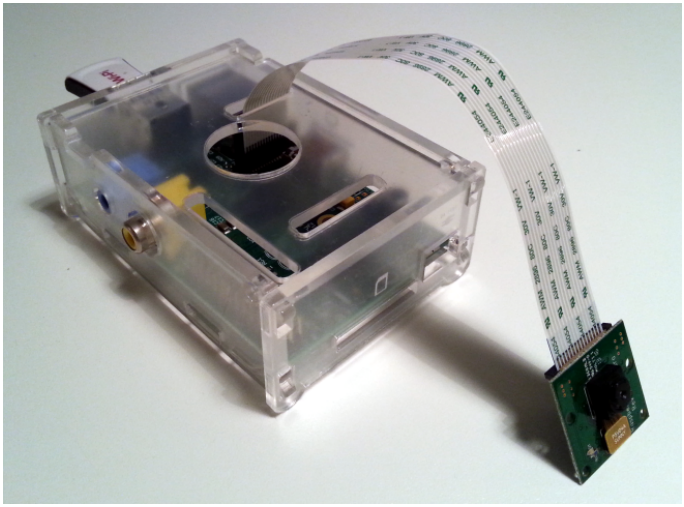


Figure 6.7: Pictures of the proposed affordable Phenotiki device.

Data storage and processing were decoupled from acquisition. Image data can be transmitted over the local network or the Internet to a centralised repository (on site or remote) for analysis. Our device can also directly connect to CyVerse (formerly iPlant Collaborative, `http://www.cyverse.org`) to upload data, and using our modules built upon the BisQue framework [150] can offer a Cloud-based application to store and analyse the images for higher throughput potential.

6.4.2 Analysis Software

The same software base is used in both the Cloud application and the stand-alone version (screenshots shown in Figure 6.6(c,d) and Figure 6.8). Robust (and validated) image processing algorithms have been efficiently implemented to enable annotation, detection, tracking and segmenting plants from the background [111], and also counting leaves automatically [7]. Furthermore, we integrated the semi-automated leaf segmentation tool presented in this section,

example of which is displayed in Figure 6.8(c).

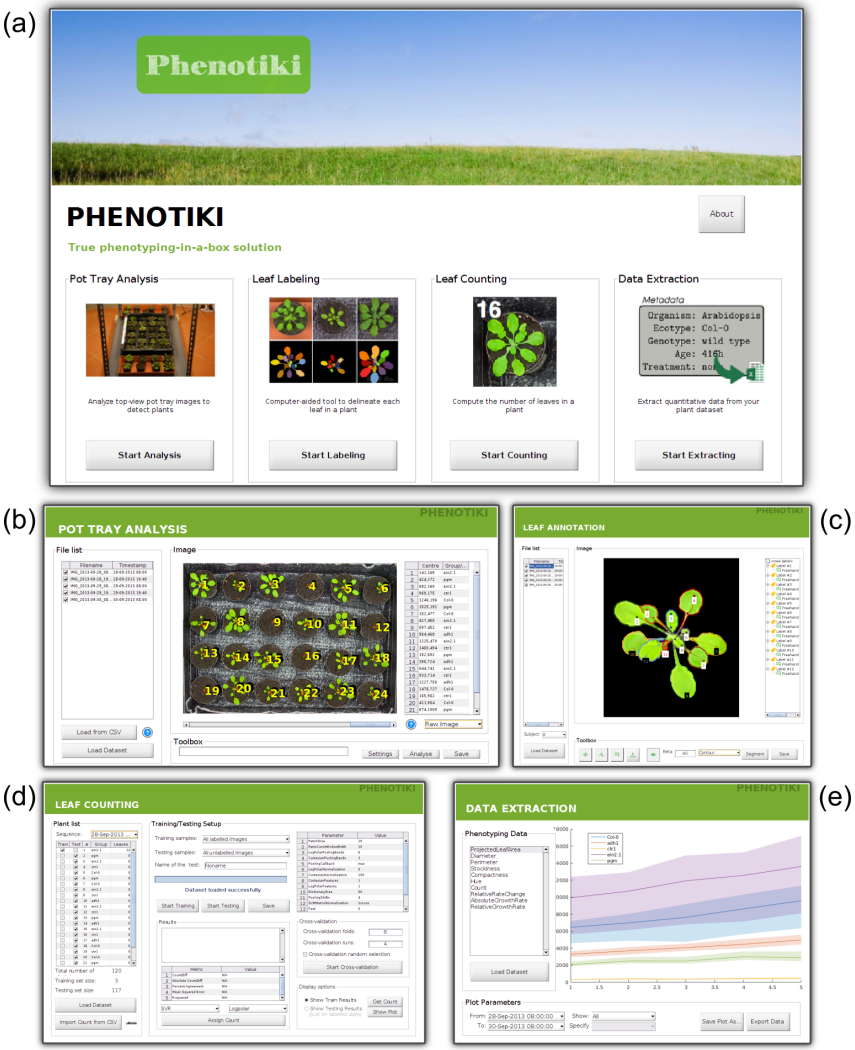


Figure 6.8: Screen captures showing the user interface of our standalone plant image analysis software.

Specifically, the *Phenotiki Analysis Software* offers four tools for the processing of top-view rosette plants, which are listed below.

Pot Tray Analysis. This tool implements the plant segmentation algorithm in [111]. The user can provide top-view images of trays, showing multiple (Arabidopsis) plants. Images can be acquired by the *Phenotiki Sensor* (the software automatically recognises them) or from custom setups. The user can adapt several parameters (c.f. [111] for further details) before starting the plant detection and segmentation. Once plants are detected, plant morphological traits can be extracted. The user can also group plants w.r.t. treatment or genotype. This information will improve data visualisation.

Leaf Labelling. This tool implements the leaf annotation tool discussed in this chapter. We improved the user interface and usability. The underlying segmentation algorithm is the same as described in Section 6.1.1 [26]. The user can add, relocate, and delete multiple scribbles to annotate each leaf. In order to use the annotation tools, plants have to be segmented using the previous plant segmentation tool. Since each leaf has to be individually segmented, the total leaf count can be used as ground-truth to train the leaf counting algorithm (c.f. Chapter 4).

Leaf Counting. This tool embeds the leaf counting algorithm presented in Chapter 4 [7]. Training labels can be prompted manually, obtained from the leaf annotation tool, or imported from a CSV file.

Data Extraction. This tool visualises and exports data and plots. The user can plot growth trends per group (showing the standard deviation), or per each individual plant. Data can be exported in a CSV file, such that the user can use external software for statistical analysis, such as *R*.

6.4.3 Software Validation

In this section, we assessed whether our software can phenotype. Specifically for this thesis, we focused for the leaf counting. The experiment involved 24 *A. thaliana* plants, including the wild type (ecotype *Col-0*) and four different mutants, all in the *Col-0* background. The *constitutive triple response 1* (*ctr1* [151]) and *ethylene insensitive 2* (*ein2.1* [152]) are defective in ethylene signaling. The *pgm* mutant is unable to accumulate transitory starch as a consequence of a mutation in the plastidic isoform of the phosphoglucomutase (*PGM*), which is required for starch synthesis [153]. The *adh1* mutant is defective in alcohol dehydrogenase activity, an enzyme playing an essential role in plant tolerance to hypoxia [154]. Although *pgm* and *ctr1* are well known to dis- play reduced growth, *ein2.1* and *adh1* mutations do not have a major impact on growth, at least based on the original reports describing these mutants. The *ctr1* mutant constitutively displays phenotypes associated with ethylene signaling, the consequences of which include extreme dwarfism [151]. The ein2.1 mutant, which is insensitive to ethylene, instead displays minor phenotypic differences when compared with the wild type, although it has been reported to grow slightly bigger [152]. The pgm mutant is smaller than the wild type [153]. Interestingly, the growth of a similar mutant (starch-free 1; *stf1*) was recently studied by digital imaging, providing an interesting benchmark for our study [155].

Growth conditions

Plants were grown in individual pots under a 12-h light/12-h dark regime; artificial daylight illumination was provided by cool-white fluorescent lamps (100 μmol photons m^{-2} s^{-h} light intensity). Temperature was on average $\sim$ 22°C (daytime) and $\sim$ 16°C (night-time). Watering was provided twice a week by subirrigation. Pots were spaced out in the tray to prevent adult plants from touching. The arrangement of genotypes in the tray was randomised to eliminate

possible bias in the results caused by variations in watering or lighting conditions. No treatments were performed.

Leaf Count Validation

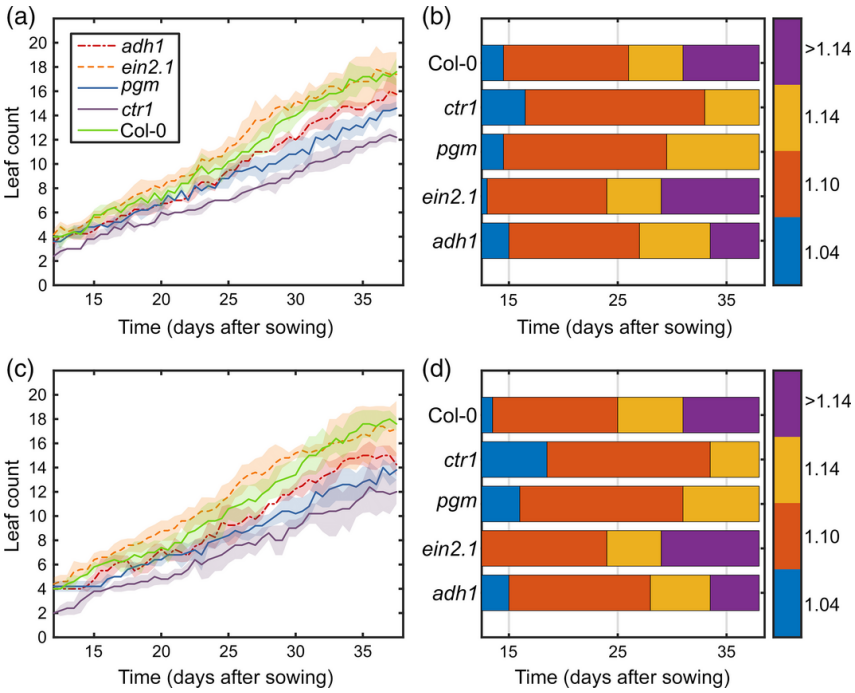


Figure 6.9: Leaf-counting data (a, b), estimated by our automated leaf-counting algorithm and (c, d) derived from the expert annotations. Results are shown as (a, c) time-series plots (mean and standard deviation) and (b, d) growth progression bars [2]. The learning-based counting algorithm was trained on a subset of plant images and then applied to the entire dataset.

For leaf count, we extended the method in Chapter 4 [7]. As the original algorithm was designed to be agnostic to scale (in order to accommodate the variable distance between sensor and camera of

the images in the challenge data set, by design), and was tested on a challenge data set that did not provide genotype information, we added two extra features: plant genotype (categorical variable) and projected leaf area (PLA, continuous variable). These properties provide information related to the typical temporal growth behaviour, or more generally speaking the dose-response characteristics of each plant, to the algorithm [156]. The categorical genotype variable was encoded as five separate dummy variables, using one-hot encoding. The new features vector is then standardised by subtracting the mean and dividing by the standard deviation.

We also compared leaf-counting progression (cf. Figure 6.9a) and developmental growth stages among genotypes, identified by the number of leaves, based on the scale discussed in [2]. In Figure 6.9(b) we highlight which day after sowing a group of plants (i.e. genotype) developed four leaves (1.04), 10 leaves (1.10), 14 leaves (1.14) and later leaf-related stages (>1.14), respectively. In accordance with the previous analysis based on plant size, we observe that *ein2.1* and *Col-0* reached successive growth stages more rapidly than the other genotypes, with *pgm* and *ctr1* producing new leaves at a markedly slower pace than the wild type. A pairwise Tukey–Kramer comparison (following a significant repeated-measures ANOVA) on leaf count data, as plotted in Figure 6.9(a), confirmed that *adh1*, *pgm* and *ctr1* differed from the wild type ($P < 0.05$).

For the purpose of the validation of the leaf count algorithm, all image data were labelled by a human expert (with the use of the annotation tool) to associate the number of leaves in each of the $1,248$ plant images in our data set, which was then used to train and evaluate the method. Figure 6.9 shows the time series of the number of leaves for each genotype (Figure 6.9c) and growth progression bar (Figure 6.9d), as derived from the expert annotations. One can readily observe that growth trends are in agreement between predicted and ground-truth (expert) counts (Figure 6.9a,c). This is also evident when visualized with growth progression bars [2] of the predicted (Figure 6.9b) and expert-derived data (Figure 6.9d),

Table 6.3: Quantitative performance of the leaf counting algorithm in Phenotiki.

	Phenotiki		**GLC [7]**	
	Training	*Testing*[a]	*Training*	*Testing*
DiC	**0.032 ± 0.772**	**0.186 ± 0.995**	0.186 ± 0.995	0.247 ± 0.143
|DiC|	**0.580 ± 0.509**	**0.702 ± 0.728**	0.880 ± 0.779	1.048 ± 1.000
MSE	**0.596**	**1.022**	1.380	2.096
R^2	**0.967**	**0.939**	0.926	0.876

We compared the original algorithm described in Chapter 4 and the extended version proposed in this chapter. Difference in count (DiC), absolute difference in count (|DiC|), mean squared error (MSE) and coefficient of determination (R2). Lower DiC, |DiC| and MSE are better, whereas higher R2 is better. The best results are highlighted in bold.

These data, following typical practice in machine-learning literature, reflect performance under a random sampling of the sets that we train on and test on. We follow a strict subject-out 50% split of the complete data. The data set includes 24 plants imaged for 26 days. The data set is split into two halves, randomly selecting 12 plants each time (and all the pictures of a plant across time) as a training set and the remaining 12 plants as a testing set (used to assess generalization error), ensuring that both subsets include examples of all genotypes (Col-0, adh1, ctr1, ein2.1 and pgm). Hence the values of DiC and |DiC| reflect the average and the standard deviation on each set, whereas MSE and R2 are, by definition, aggregates.

[a] If we are to repeat this random split many times (in machine learning this is a form of cross-validation) we see that performance remains the same, with average and standard deviations of the same measurements as 0.041 ± 0.154 (DiC), 0.841 ± 0.067 (|DiC|), 1.335 ± 0.180 (MSE), 0.923 ± 0.008 (R2). This indicates stability with respect to the set that we train on.

demonstrating that our algorithm can detect the specific growth stages of a plant (principal growth stage 1, [2]).

Quantitative analysis is shown in Table 6.3, reporting four evaluation metrics [7, 67]. With respect to the algorithm presented in [7],

Phenotiki adopts an extended version that relies on image features and also plant genotype and projected leaf-area variables to estimate the number of leaves. The results produced by the algorithm agree with leaf counts made by expert inspectors ($R^2 = 0.94$ on the testing set), with mean and standard deviations of less than 1 in absolute count ($|\mathrm{DiC}|$). Automated leaf counts differed from an expert's manual count by not more than one leaf in 83% of examples.

As a further validation, we evaluated whether interchanging the expert data with the automated predictions had any effect in statistical comparison testing. Table 6.4 compares the results of the pairwise Tukey–Kramer comparison (following a significant repeated-measures ANOVA) between count data of different genotypes obtained with the expert data and the automated counting, respectively. Observe that the P values are close to each other and at the 0.05 significance level the phenotypic conclusions are the same.

6.5 Impact of Phenotiki

Since the publication of the accompanying manuscript [3],[2] Phenotiki has been employed in different countries and research groups. We also tracked the people who downloaded the analysis software presented in this chapter, asking them to fill in a form on the website (`http://phenotiki.com`), before downloading the software. As of now, the software has reached 71 unique downloads from all over the world. In Figure 6.10, we plot the number of unique downloads per country of the *Phenotiki Analysis Software*. Mostly of the requests come from academic institutions (universities or research centres). However, some commercial stakeholders have also requested the software, showing that our system has raised interest outside the academic environment as well.

[2]The paper has been cited 21 times (data collected the 08$^{\text{th}}$ October 2018 from Google Scholar).

Table 6.4: Statistical significance of differences in leaf count between genotypes is not affected by using the leaf count algorithm. Shown is the pairwise post-hoc comparison with the Tukey-Kramer method between leaf count data of Col-0 and the other genotypes, following a two-way repeated measure ANOVA testing once on RaspiCam derived data analysed with Phenotiki and once on manual annotation by an expert, separately. No difference in the significance of the tests is observed between results obtained with Phenotiki and manual annotation, respectively. Sample size (24 subjects, 52 time points) is equal among the two tests.

Genotype	Genotype	Mean	Std	P-Value	95% Conf. Interval	
(I)	(J)	Diff	Error		L. Bound	U. Bound
Phenotiki						
Col-0	*adh1*	1.1494*	0.3033	0.0096	0.2373	2.0616
	ctr1	3.3240*	0.2860	<0.001	2.4640	4.1839
	ein2.1	-0.6217	0.2860	0.2315	-1.4817	0.2383
	pgm	1.8529*	0.2860	<0.001	0.9929	2.7128
Manual						
Col-0	*adh1*	1.3471*	0.3932	0.0209	0.1647	2.5295
	ctr1	3.5962*	0.3707	<0.001	2.4813	4.7110
	ein2.1	-0.7731	0.3707	0.2664	-1.8879	0.3417
	pgm	2.2423*	0.3707	<0.001	1.1275	3.3571

*The mean difference is statistically significant at the 0.05 level.

6.6 Extending Phenotiki

Currently, we are extending the *Phenotiki Sensor* and the *Phenotiki Analysis Software* to cope with the plant root analysis.[3] Specifically for this project, we analyse images of the root system architecture (RSA) of the chickpea plant (*Cicerum arietinum L.*).

Plants are grown inside an affordable rhizotron (called *mesocosm*)

[3]BBSRC project #BB/P023487/1 "*Improving root system architecture for enhanced drought tolerance and nutrient use efficiency in semi-arid agriculture of chickpea*". Principal Investigator: Peter Doerner.

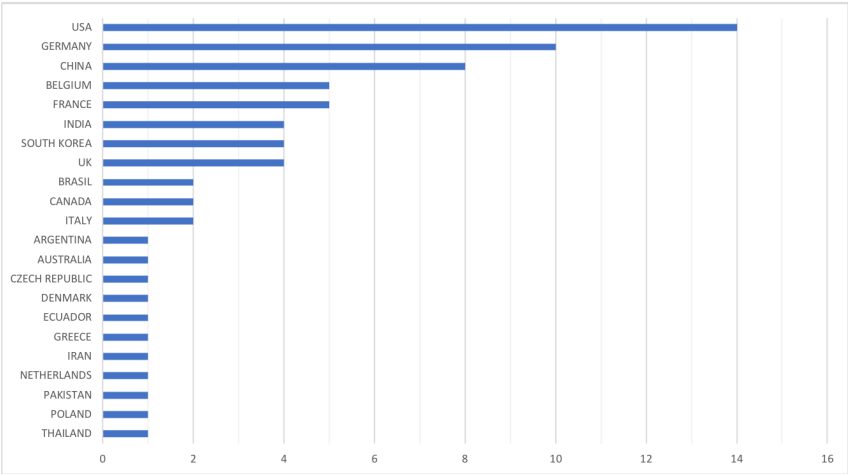


Figure 6.10: Number of downloads of the *Phenotiki Analysis Software* per country.

of size $1500 \times 450 \times 6$ mm, using real soil. A mesocosm is fitted with a pane to allow root visualisation and imaging. In order to acquire images to the whole height of the mesocosm, we used five Phenotiki Sensors placed at ~ 0.8m distance from the mesocosm, installed in an imaging station that allows picture acquisitions under controlled conditions (c.f. Figure 6.11). Cameras are placed ~ 30cm distance, such that the field of view of neighbouring sensors overlaps. To reduce glass reflection and refraction, the station is fully covered with black velvet, blocking undesirable ambient light from interfering with the acquisition process. Since no light enters, lights are necessary to illuminate the scene. The imaging station is equipped with two LED strips (colour temperature of $6,000^{\circ}$K), as shown in Figure 6.11(B).

Since images have to be taken simultaneously, we implemented a multi-sensor image acquisition protocol, where one sensor is set as *master*, and the others as *slaves*. The master sensor enables a hot-spot WiFi network to allow the slaves to connect to it. Users can still in-

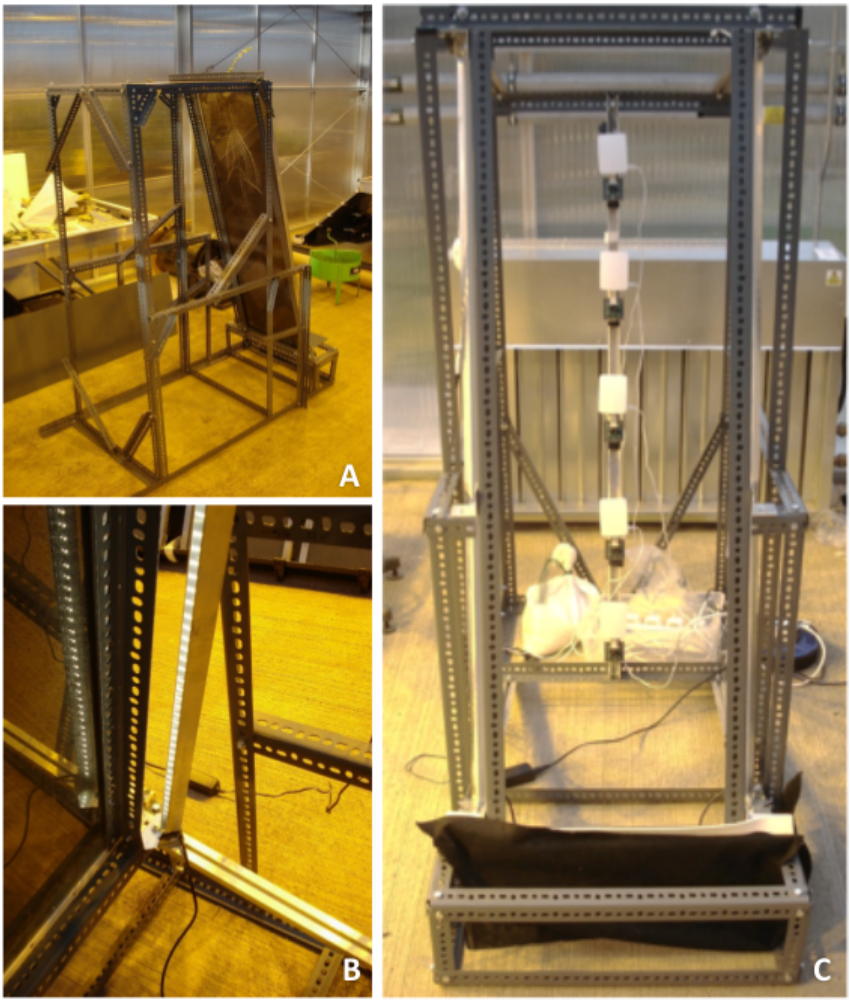


Figure 6.11: Camera station setup to acquire images of chickpea RSA. (A) Camera station; (B) LED strip close-up; (C) Phenotiki sensors arrangement. The image station is covered by black velvet during acquisition.

teract with the sensors through the web interface (c.f. Figure 6.6(b)). When a sequence is being acquired, the master triggers all the connected slave sensors to acquire an image. When all the slaves have finished, the master is notified and collects the images from all the slave devices. This distributed image acquisition protocol increases the field of view to image fully the mesocosms.

Differently from the original Phenotiki setup [3], we used a camera sensor with adjustable focus.[4] In order to compensate for the lens distortion, we calibrated all the cameras with a *ChAruco* board [157]. We acquired ~ 20 pictures per camera to compute the homography matrix necessary to remove the distortion [158].

When a sequence is acquired, images have to be *glued* together (stitched). To successfully stitch the images, it is necessary to extract robust points from the overlapping regions. To assist the stitching process, we placed a set of fiducial markers [157] alongside the field of view of the sensors. Once markers were detected, they provide robust points for the stitching. However, due to the non-linear nature of the lens distortion, fiducial points were not enough to obtain a suitable image stitching. We augmented the number of robust points, by extracting scale-invariant key points from the images [13], as shown in Figure 6.12(A1). We perform pairwise stitching (c.f. Figure 6.12(A2)) to obtain the entire image (Figure 6.12(B)).

When the images were stitched together, we applied a simple, yet effective algorithm for root segmentation to remove the background. Due to the controlled acquisition conditions, roots appear very bright (high pixel intensity) w.r.t. the soil (low pixel intensity). Firstly, we converted the image into the CIELAB colour space and we retained the Luma channel for segmentation. We extracted 160×160 non-overlapping patches from the image and we found the most suitable threshold value τ, using Otsu's method [159]. Although this algorithm found the most suitable τ from a bimodal colour intensity distribution, a patch could not contain root segments (e.g.

[4] Additional information available at `https://www.waveshare.com/rpi-camera-b.htm`.

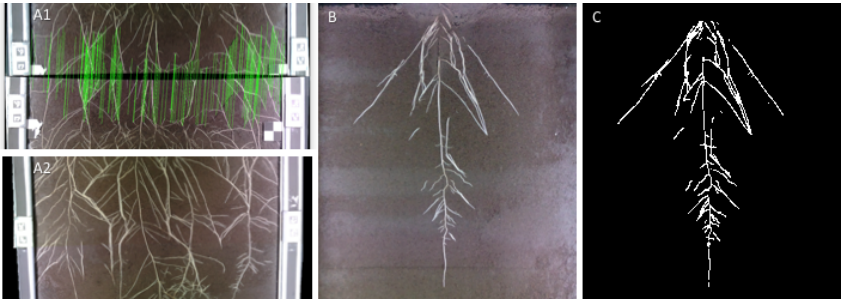


Figure 6.12: Chickpea images stitching. (A1) Robust keypoint detection and matching [13]; (A2) Pairwise image stitching; (B) Full image stitching result; (C) Root segmentation.

all soil). Therefore, a patch was segmented if τ was within the range $[a, b]$.[5] This method guarantees that (almost) all the white pixels in the segmented images belonged to root. However, the resulting image is undersegmented (e.g. some roots are missing from the segmented image). To solve this, we iteratively extract 160×160 patches from the points where roots were detected in the previous step. We find τ using Otsu's method, but the threshold value was accepted if $\tau \in [a + 20, b]$ (a more restrictive constraint). This procedure is repeated for each root pixel, until no more roots was found. Lastly, we applied morphological opening and closing operators to remove the noise. An example of the output of this segmentation method is displayed in Figure 6.12(C).

We are currently planning to include this new functionality and others within the *Phenotiki Analysis Software*, together with other algorithms to, for example, extract root traits.

[5] In our experiments, we used the range $a = 125$, $b = 170$.

6.7 Discussion

In this chapter, we presented a tool to allow experts in plant phenotyping to annotate plant images. Using the graphical user interface, users can place a *scribble* on each leaf, such as dots or lines. Using a segmentation algorithm based on random walk [26], our semi-automatic tool can generate a per-leaf plant segmentation. Later, this annotation tool was bundled within the *Phenotiki Analysis Software* [3]. Although the algorithm underneath the segmentation tool remained unchanged, we improved user interface and front-end functionality. In the future, we are planning to extend our software to analyse root system architectures.

Our tool helped expert users to reduce the annotation time to ~ 2 minutes. However, with an increasing amount of unlabelled plant images, experts need help in the annotation process to ease the burden. For this reason, we want to investigate the possibility of *employing* non-experts in the annotation task. Recently, several platforms have been made available to involve the internet community in scientific projects.

In the next chapter, we discuss how crowdsourcing can be used to train volunteers to annotate plant images. Specifically, we used the *Zooniverse* web platform, designed to involve citizen scientists in annotating the images used in research projects. We measured the inter-observer variability within experts and non-experts when annotating plant images with the annotation tool and we compared this variability with the non-expert inter-observer variability of the citizen scientist.

7

Crowd Sourcing and Inter-Observer Variability

The annotation tool presented in the last chapter allowed us to annotate and segment leaves from plants, reducing drastically the time required to perform such a task. However, there are many images that have not been annotated yet and experts cannot to the entire job alone. Therefore, it is necessary to look for annotators that can do this job, possibly remotely. In this chapter, we present our work on how we collected annotation data from the crowd sourcing platform *Zooniverse* (`https://www.zooniverse.org`).

The question that arises at this point is: *are citizen scientists reliable?* We will also study the intra- and inter-observer variability amongst experts and not-experts. In the medical community, variability among observers is known to exist and has been accepted [160]. Also experts in plant breeding, diseases, and taxonomy

This chapter is based on:

- M. V. Giuffrida , F. Chen, H. Scharr, and S. A. Tsaftaris. "Citizen crowds and experts: Observer variability in image-based plant phenotyping". *The Plant Methods*, 2018.

agree that variability exists [161–164]. For example, several studies [165–167] have been used as de-facto references for discussing rater disagreement when visually scoring leaf diseases on the basis of scales. At the same time they have become motivating references advocating that image analysis systems can help reduce rater variation [168]. They have been also perused in advocating the use of digital imaging itself as opposed to on site surveys with rating scales [169]. Even the image-based phenotyping literature has been perusing these works [170, 171]. However, an extensive literature review has not found a comparison of raters on visually *quantifiable* traits or phenotypes.

7.1 Motivation

Estimating observer variability is crucial because it primarily allows us to put bounds on effect sizes and devise annotation strategies that minimise annotation effort (e.g. by splitting annotation effort among many observers). At the same time, by evaluating agreement comparing experienced (expert) and non-experienced (non-expert) observers, we can evaluate the potential of using non-experts for simple well-defined annotation tasks. It also allows us to put the performance of algorithms in comparison to intra- or inter-observer variation and assess how close we are to achieving human performance. It may even permit us to devise different algorithmic approaches that learn despite the presence of disagreement [172, 173].

Equally exciting is the potential to explore how the use of common citizens can be used to not only annotate data for machine learning but as being part of a phenotyping experimental pipeline. The introduction of Amazon Mechanical Turk (AMT, `https://www.mturk.com/`) that permits the use of humans (via fee) in solving computer based microtasks in combination with annotation frameworks (e.g. LabelMe [174]) has led to an explosion of the potential use of crowdsourcing – a term coined by Jeff Howe in 2006 [175]. It has been used for a variety of tasks already, even

for plant research e.g. `http://photonynq.org`. However, there have been ongoing debates as to how one can control the quality of outcomes because in principle, crowdsourcing allows 'anyone' to contribute. More recently, citizen-powered platforms, where volunteers participate to help with a task, as opposed to receiving a reward (a payment in real [AMT] or virtual money [Gamification]), have received particular attention by many researchers.

In this chapter, we will use *Zooniverse*, which allows researchers to build projects to collect data from thousands of people around the world, in order to support corresponding research. Several exciting projects have used the platform already: for example, in [77] used the data from a penguin watch project to automatically count penguins in the wild. We aim to estimate observer agreement with a simple, yet expertly designed, image-based observational study. We select images of *Arabidopsis thaliana* (taken from a dataset in the public domain [6]) and ask several observers to count leaves using a variety of setups in a controlled fashion. At the same time, we included the same images within a larger citizen-powered research project that runs on Zooniverse. Specifically, we aim to assess whether:

i. variations exist between the same observer (intra-observer);

ii. computer-aided counting, using a specifically designed annotation tool, helps to reduce variability compared to straightforward visual observation;

iii. observers differ from each other (inter-observer);

iv. higher resolution reduced observer variability;

v. observer variability has any statistical influence in separating a cultivar of known different leaf growth w.r.t. wild-type;

vi. time needed for annotations depends on expertise;

vii. we can simulate the effects of randomly sampling from an observer population on statistical inference;

viii. counts from a citizen-powered study can be used for phenotyping; and

ix. a recent ML algorithm that predicts leaf count from plant images performs within the variation of observers.

7.2 Methods

We recruited 10 annotators: 5 who have experience with image-based plant phenotyping (shorthanded below as ExP) and 5 who do not have experience with phenotyping but yet have experience with images (shorthanded hereafter as NExP) to annotate a subset of the *Arabidopsis* dataset in [6]. Specifically, each annotator had a set of different tasks to accomplish using visual tools or simple observation designed to assess the influence of the factors considered in this study. Details of the approach taken are provided below.

7.2.1 Employed Data

The data used in this study have been collected using the Phenitiki sensor [3]. Images of two cultivars were selected (the wild-type *col-0* and *pgm*), 5 replicates each every other day at 8am (i.e. every 48 hours). *pgm* is known not to be able to accumulate transitory starch due to a mutation in the plastidic isoform of the phosphoglucomutase, which is required for starch synthesis and overall is known to be smaller than the wild-type [176]. Furthermore, *pgm* was recently shown to produce new leaves at a pace lower than wild-type [3]. Thus, we knew a priori that these cultivars should show differences in a longitudinal assessment of leaf count. The sampling frequency chosen (every 48 hours) results in 13 time points per each plant, providing 130 images overall for annotation. This sampling frequency was chosen after statistical power analysis on the sample size of an ANOVA experiment [177] drawing effect sizes reported in [3].

Images were cropped such that a plant appears centred in the field of view. Plant images from the Raspberry Pi camera had an

effective resolution of 300×300 pixels (hereafter shorthanded as *RPi*), whereas the ones from the camera with movable optics had 470×470 pixels (shorthanded as *Canon*). In addition, to properly test intra-observer variability eliminating as much as possible effects of visual memory, a copy of all images was created, where images were artificially transformed by random $90°$, $180°$, $270°$ rotation or horizontal/vertical flip. These transformed datasets are shorthanded as *RPi'* and *Canon'*. Data within each set were randomized to break temporal consistency and within genotype associations and to satisfy an identically independently distributed (IID) data source design.[1] Dataset names were obscured as A (RPi), B (Canon), C (RPi'), and D (Canon') such that observers were blinded to what the sets meant and reduce possible bias in ratings.

7.2.2 Study Design

A customised graphical user interface, based on the annotation tool in Phenotiki (c.f., Section 6.4.2) was developed. The tool prompted the user to select a dataset for annotation (from A, B, C, D) and the selected list of images was automatically loaded. For each image, the observer could place dot annotations marking every leaf they could identify. Critically dots remained visible throughout a plant annotation helping the annotator keep track of visited leaves. When the observer was done, they could proceed to the next plant. Zoom and pan functionality were available to help observers visualise scenarios such as small emerging leaves and occlusions. Annotation timing was recorded but observers were not aware of this fact. Annotation timing (per plant) was calculated as the time elapsed from the first and last leaf annotation for a given plant.

Experienced (with image-based plant phenotyping) and non-experienced observers were recruited to participate in this observational study. They were provided with a description of the purpose

[1] This more closely emulates how experts rate data with visual scales in the field since there is an inherent assumption that previous ratings and images of the scene are not used as reference.

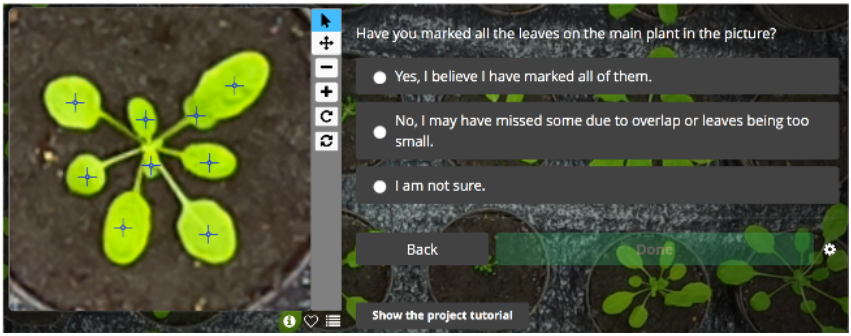


Figure 7.1: Screenshot of the Zooniverse site used here showing annotations and the confidence question.

of the study, and were asked to consent to participate in the study. They were shown a guide and an introduction to the annotation tool to ensure a common baseline. Specifically, we showed them examples of good plant annotations, where they were asked to mark leaves at the centre of the leaf blade (or the most visible area in case of severe overlap). Each observer was assigned two or more of the datasets to rate and count leaves. The order of the datasets shown was randomised and never of the same orientation (e.g. if one was shown A the next dataset would be C or D) to minimise effects of memory. To further reduce memory effects a 10 minute break was enforced between annotation tasks.

Some observers were asked to rate the images also without the use of the tool, but recorded leaf counts in a spreadsheet after shown an image.

Time to complete each set was recorded in addition to the times recorded by the tool itself (see annotation timing above).

7.2.3 Citizen-Powered Study

The A data (RPi) were included as part of a larger citizen-powered study ("Leaf Targeting", available at `https://www.zooniverse.`

org/projects/venchen/leaf-targeting) built on the online platform Zooniverse (https://www.zooniverse.org/). Using the Zooniverse application programming interface (API), an annotation work-flow was designed that showed an image to a user via a web browser. The users (random visitors) were asked to view a tutorial on how to annotate leaves. The task essentially involved placing a dot annotation on each leaf, thus retaining the characteristics of the interface used in the fully controlled study described previously. Users could as well zoom in and out and delete dot annotations. Users were also asked to answer a question after each plant was annotated as to their confidence in having annotated all leaves (encoded as Yes: 3, Not sure: 2, Missed leaves: 1). An example of an annotated image along with the interface and questions seen by the users are shown in Figure 7.1. We note that the users have the option to log in to the platform and also to comment about images where they can discuss issues related to the image or the task in general. We set the work-flow to repeat the same image 8 times after at least all images have been annotated 3 times; images for annotation are shown at random and thus annotations can be treated as IID and the same image is not rated by the same user. The system exports complete information for each annotated image such as image ID, user name (or unique IP), time, the locations and number of dots, and the response to the confidence question

7.2.4 Statistics and Evaluation Metrics

A variety of descriptive and summary statistics as well as several statistical methods were used to evaluate agreement in the controlled experiment. We note that in the case of discrete counts and heavily zero inflated differences (when comparing counts between observers) many of the common statistics and visualisation methods can lead to misinterpretations. Thus, between a reference observer (X_R) and one of the other observers (X_o), we adopted the evaluation metrics discussed in Section 2.4. In addition, we also use the *Krippendorff's alpha*, which is an index of inter-observer agreement [178].

(We used the mALPHAK implementation in Matlab [179] treating counts as a ratio scale variable comparing X_R and X_o).

To visualise agreement between pairs of observers we used a modified version of the Bland-Altman (BA) plot [180] in conjunction with the histogram of count differences. For the BA plot, we plot colour labelled squares with square colour varying according to how many points agree on the same coordinates. This is necessary since we observed that in scatter plots of discrete quantities, points will overlap misrepresenting the true distribution of the data.

Finally, while evaluating agreement is interesting on its own, we also considered an application-driven measure of agreement by estimating a mixed effect repeated measure two way ANOVA on count data as employed in [3] for the two cultivars. By this, essentially we test whether any observable differences exist in between cultivar longitudinal trends obtaining average counts using a different set of observers. We treated subject ID (i.e. the replicate) as a random effect whilst all other as fixed effects. To not over-inflate degrees of freedom we treated time as a continuous predictor. Of particular interest is the interaction term between time and cultivar (cultivar*time hereafter), since this is the term that tests longitudinal differences between the cultivars.

7.3 Results

7.3.1 Intra-Observer Variability

We assessed this via a second reading from the same observer using the tool. In Figure 7.2A we plot histograms and Bland-Altman (BA) plots for two observers on the datasets A, C (ie. same as A but with geometric changes). Considering also the corresponding rows in Table 7.1, we can see that intra-observer agreement overall is excellent, with the NExP observer showing slightly higher variation (higher standard deviation) and decreased agreement (alpha) compared to ExP.

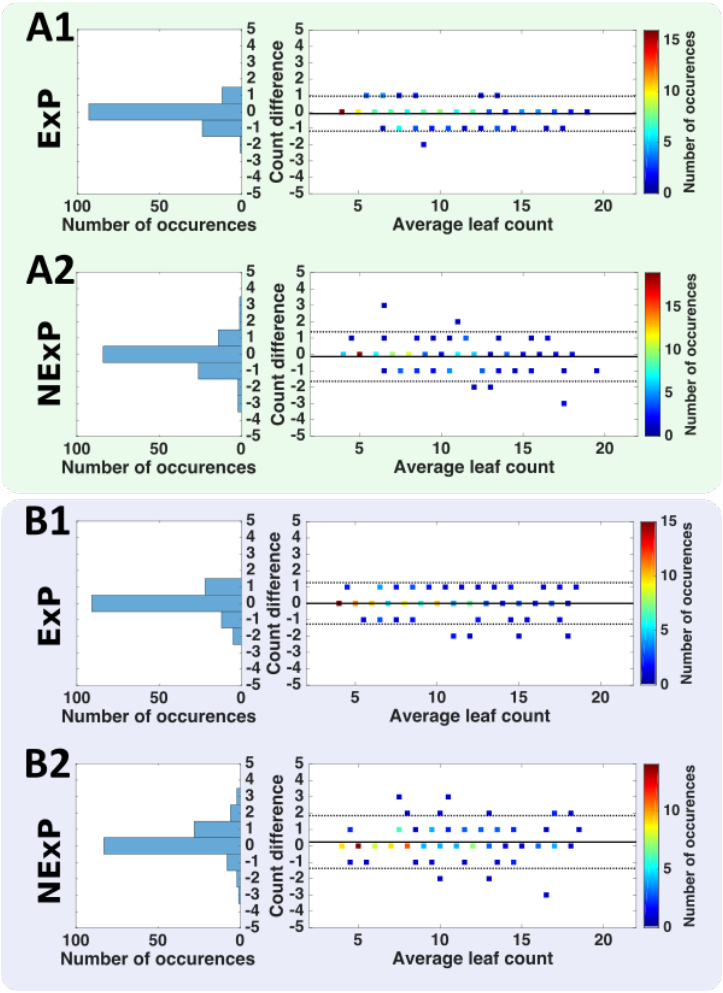


Figure 7.2: A Intra-observer variability of experienced (A1) or non-experienced (A2) observers in RPi. B Influence of the tool in intra-observer measurements in experienced (B1) or non-experienced (B2) observers in RPi

7.3.2 Variability Between Tool and Spreadsheet-Based Counting

To assess whether the tool contributes to lower variability in intra-observer measurements, in Figure 7.2(b) we show histograms and BA plots comparing counts obtained via the tool or spreadsheet measurements using the same, ExP or NExP, observer, shown respectively left and right. Note that deviation is higher when compared to the intra-observer findings using the tool alone (previous paragraph). It appears that the tool has less effect (smaller deviation) to an ExP, whereas it seems to help reduce variability for NExP. This adheres to comments of NExP observers stating that when leaf numbers are high, and plant structure appears complex, it is hard to keep counting the leaves manually without visual reference resulting in frequent restarts of counting (even 3 times). We note that the tool retains the placed dots visible to precisely help visual memory. The same conclusions can be drawn from the statistical numbers shown in Table 7.1, however with slightly decreased agreement in the NExP observer.

All the results presented in the following refer to tool-based annotations.

7.3.3 Inter-Observer Variability

To assess inter-observer variability we selected one experienced observer as a reference and compared against other ExP and NExP observers (a total of 9), which allows us to be concise (e.g. by showing representative comparison pairs instead of all possible combinations). Although this approach does not take into account observation error of the reference observer, the chosen observer had the smallest intra-observer variation (see entry marked with a '*' in Table 7.1).

Figure 7.3(a) and Figure 7.3(b) visualise inter-observer agreement in the case of RPi and Canon, whereas Table 7.1 offers statistics. Overall we see that agreement is excellent independent of experi-

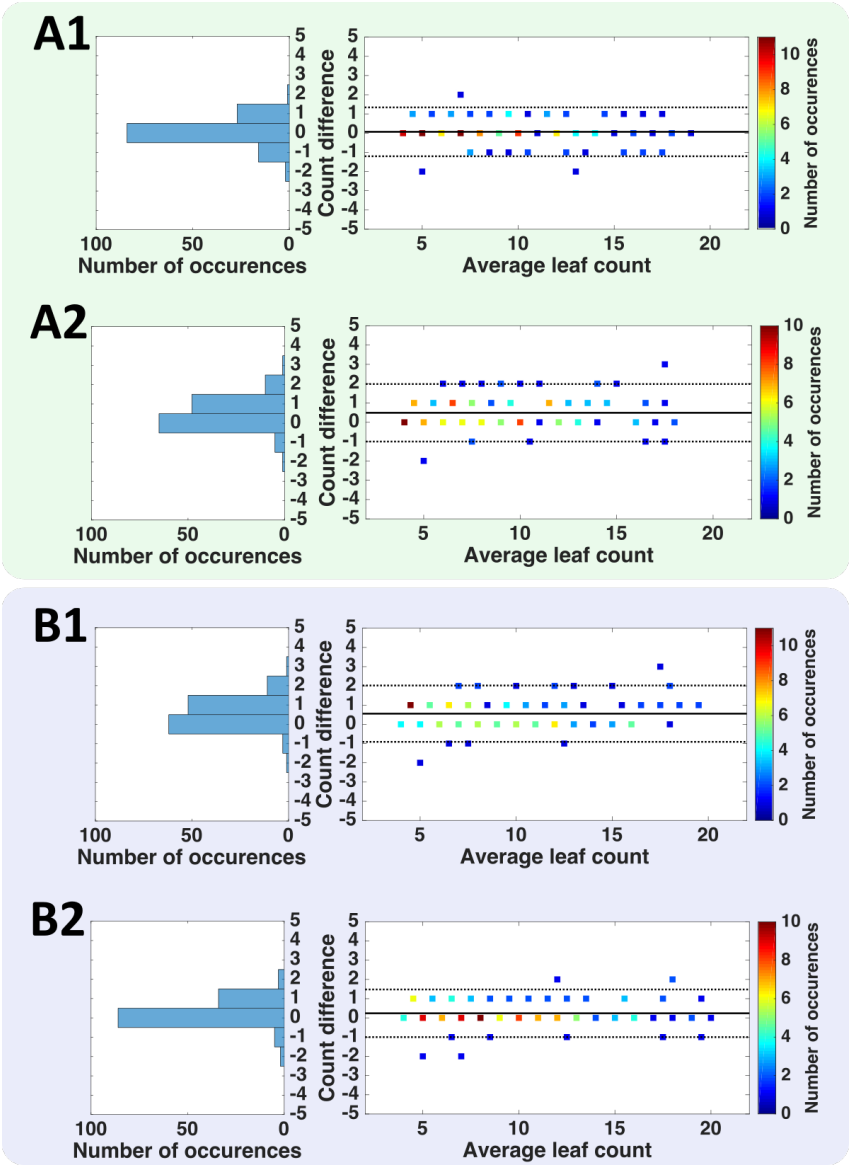


Figure 7.3: Inter-observer and influence of resolution. A: Inter-observer variability among experienced (A1) or non-experienced (A2) observers in RPI; B: same as in A but in Canon data. (*This figure continues on the next page*)

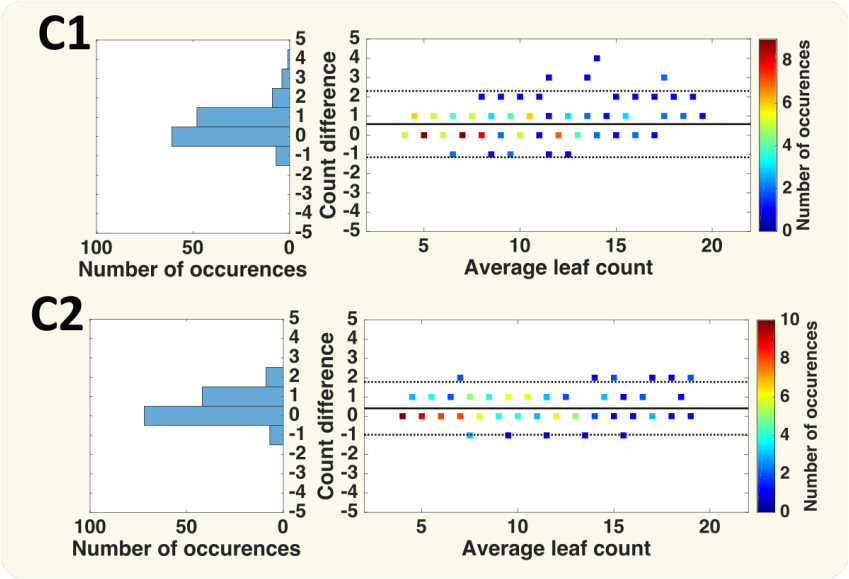


Figure 7.3: (*continue from the previous page*) C: Variability of experienced (C1) or non-experienced (C2) observers when comparing counts of the same observer in RPi and Canon data.

Table 7.1: Measurement of agreement between experienced and non-experienced observers. For shorthand definitions see text. For DiC and |DiC| average and standard deviation are reported. Note that these correspond also to bias and limits of agreement (when standard deviation is multiplied by 1.96) of the Bland-Altman plots reported. ↓ means lower is better, whereas ↑ means higher is better.

	DiC ↓	|DiC| ↓	MSE ↓	R^2 ↑	alpha ↑
Intra-observer (RPi) tool					
Experienced [*]	0.10 (0.54)	0.29 (0.47)	0.307	0.980	0.987
Non-experienced	0.13(0.77)	0.42 (0.65)	0.600	0.960	0.981
Tool-based vs. Spreadsheet-based (RPi)					
Experienced	0.00 (0.64)	0.33 (0.55)	0.415	0.970	0.986
Non-experienced	0.23 (0.82)	0.46 (0.71)	0.730	0.950	0.977
Inter-observer (RPi) tool					
Experienced	0.07 (0.65)	0.37 (0.53)	0.423	0.974	0.980
Non-experienced	0.49 (0.76)	0.60 (0.67)	0.815	0.962	0.962
Inter-observer (Canon) tool					
Experienced	0.55 (0.74)	0.63 (0.68)	0.861	0.969	0.959
Non-experienced	0.23 (0.63)	0.37 (0.56)	0.450	0.977	0.976
Intra-observer across resolution (RPi and Canon) tool					
Experienced	0.57 (0.87)	0.68 (0.79)	1.100	0.950	0.965
Non-experienced	0.40 (0.70)	0.51 (0.62)	0.650	0.973	0.977
Citizens inter-observer (RPi) Zooniverse					
Exp vs Consensus (avg)	0.53 (0.77)	0.62 (0.69)	0.869	0.962	0.960
Exp vs Consensus (max)	0.08 (0.82)	0.45 (0.69)	0.684	0.957	0.971
Consensus (avg) vs sing. random	0.00 (0.78)	0.42 (0.65)	0.607	0.960	0.970

Notes: * This observer is noted as the reference observer for the remaining analysis.

ence. At times experienced observers appear to disagree more particularly when resolution is higher. This is likely attributed to how experienced observers appreciate new leaf emergence and particularly if they are trained to see it or not.

7.3.4 Influence of Resolution on Intra-Observer Variability

This variation among experienced observers becomes also evident when comparing the same observer and their annotations when resolution alters (Figure 7.3(c) on page 143). The ExP observer (who is also the reference) tends to underestimate when resolution is lower. Whereas the NExP observer shows less under-estimation and higher agreement. It appears that NExP observers may miss young leaves independent of resolution (as they are not trained to see them) whereas the ExP observer misses them only on lower resolution.

7.3.5 Influence of Observer Variation in Longitudinal Analysis

In Figure 7.4, we show per-day average leaf count for each cultivar (i.e. averaging across replicates) when using annotations from different sets (and numbers) of observers for the RPi data. The top row refers to using a single ExP or NExP observer i.e. averaging within the population of each cultivar (panel A); whereas the middle row refers to a group of observers within their expertise, averaging first across observer annotations, and then across replicates (panel B). Panel C is similar to B but averages across all observers. The plots show average leaf count (within the population of each cultivar) and 1 standard deviation (shading) from the mean of the population. It is evident that given the effect size of the chosen cultivars, trends of average leaf count are expected even when using a single observer, albeit the ExP observer shows less variation. When combining observations across a group of observers trends still show clearly and

Table 7.2: F and p-values for the ANOVA tests corresponding to the plots in Figure 7.4. Only time*cultivar interaction is shown corresponding to the factor of interest (longitudinal trend). Results with 'All' and Consensus citizen average (or max) across per-plant observations.

	Sum Sq.	F	p-value
A single ExP	47.816	43.775	0.000167
A single NExP	47.170	30.017	0.000588
All ExP	56.264	34.661	0.000367
All NExP	49.533	29.116	0.000649
All observers	53.219	32.280	0.000464
Consensus Citizen (average)	66.923	19.044	0.0024
Consensus Citizen (max)	76.855	23.713	0.0012

one may even argue that averaging across NExP tends to perform even better than a single NExP observer (compare panel B and A).

In Table 7.2 the results of the statistical ANOVA experiment are shown focusing only on the interaction term of interest (time*cultivar). We can see that in all cases the interaction is significant ($p \leq 0.05$) confirming the visual findings of Figure 7.4 and analyzed above. Note that although the smoothing effect is evident in the plots, when using more observers slightly increases the p-value (decrease of the F score). This could be attributed to the fact that when using a single observer their behaviour (e.g. tendency to under-estimate) may be considered a fixed effect which is captured in the intercept, whereas using a population of observers (even of the same expertise) this may not be captured by the specification of the ANOVA model.

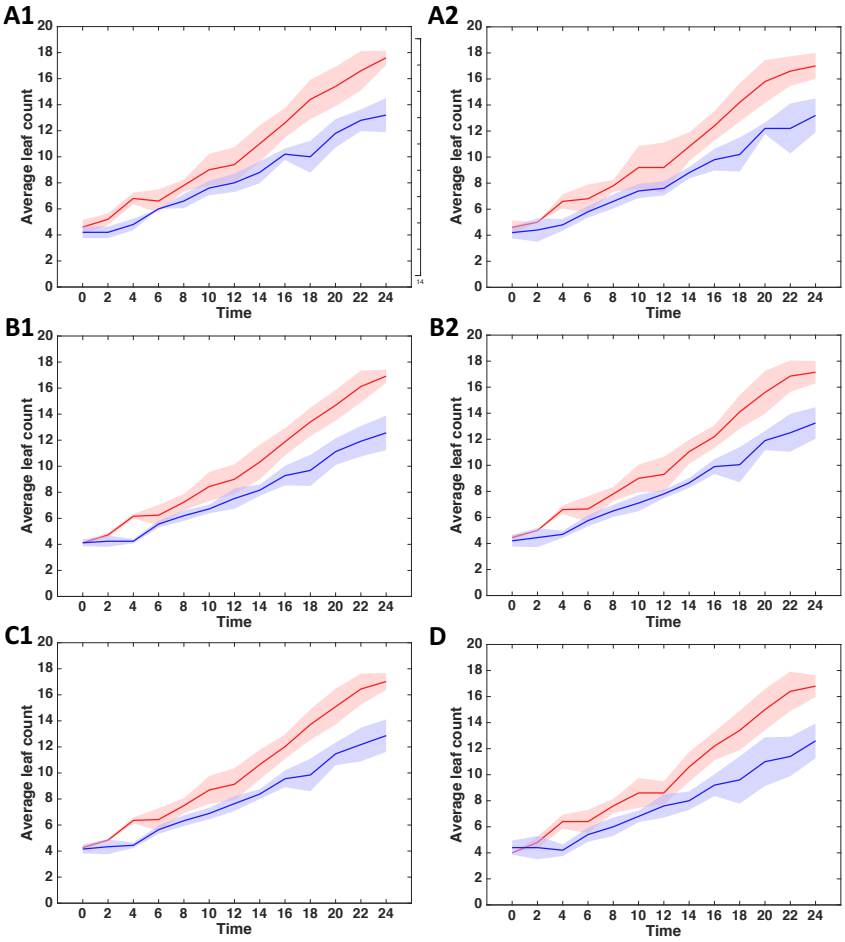


Figure 7.4: Average longitudinal count curves (solid) of the two cultivars [red: *col-0*; blue: *pgm*] and 1 standard deviation (shaded area), shown in A: relying on a single experienced (left: A1) or non-experienced observer (right: B1); B: relying on all experienced (left: B1) or non-experienced (right: B2) observers; C: relying on all together; and in D: relying on the consensus citizen

7.3.6 Time Results

Overall, we find that on average observers using the tool spent 48 minutes to annotate 130 plants for an average of 21 seconds per plant. Observers using the spreadsheet took on average 42 minutes. These findings were obtained by recording start and stop times of 5 observers in a controlled setting and provide aggregate timing information across an annotation task.

On the other hand, by keeping track of time when annotations were placed using the tool, more precise per leaf timing annotations were obtained (see Methods). Since this approach assumes that observers continuously label leaves, which may not hold if they take a break whilst labelling a plant, times greater than 200 secs were considered outliers and were excluded from analysis.

Recording the time required to annotate a plant, we found that there is no statistical difference between experienced and non-experienced observers (p-value 0.245). On average, within the 21s required to annotate a plant, only 8.5s were used to actually complete the task. (In general, an annotator takes 1.10 ± 2.15 seconds per-leaf). We argue that annotators use the remaining time to assess how to annotate a plant and evaluate the quality of their own work. In fact, several annotators were double-checking their work after they finished to annotate all the leaves. We found this by analysing the timestamps recorded for each annotation. For some plants, the last annotation was placed after 40 minutes from the first one on the same image. Moreover, we also found no correlation between errors and time. Specifically, comparing the leaf count with the reference expert, the DiC is not affected over time.

7.3.7 Simulating a Citizen-Powered Study

Given the number of available observers on RPi (9 observers) and the *a priori* knowledge of their experience, it is of interest to explore: (i) the effects of using multiple observers for phenotyping by reducing their load (i.e. not having to annotate all images but a fraction of

them) and consequently; (ii) the potential of using citizen-powered research platforms for phenotyping (where experience could be an unknown factor).

At first instance we wanted to simulate how many annotations we need to still maintain the phenotyping findings of the previous section: i.e. that there is an effect between time and genotype in the ANOVA setup. For this purpose, we set up a Monte Carlo simulation study that at each trial randomly draws a sampling matrix with K observations per time point. For example, for two observations per time point, this matrix has $K = 2$ ones per row (a row is an observation) for a total of 260 ones (the rest being zeros). The placement of ones select from which annotator an observation is obtained for this time point. For more than 1 annotation per time point (i.e. plant image), annotations across observers are averaged.

We varied $K = 1, 2, 3$ drawing from all available annotators ($n = 9$) or only from experienced ($n = 5$) or non-experienced observers ($n = 4$) to inspect the influence of mixing experience in annotations in the overall result. At each trial we run the ANOVA experiment and record the p-value of the interaction term (time*cultivar). We draw 500 trials for each variation of setup (K and the observer groups) and finally obtain summary statistics of the distribution of the p-values among the 500 trials, namely minimum, maximum, mean, standard deviation, and kurtosis (a notion of symmetry and normality).

Table 7.3 reports the findings of this study. Overall it can observed that, independently of the number of annotations used or the experience of observers, the p-value is not statistically significant (the max p-value is always below the significance threshold). This is telling since even 1 annotation is enough for the effect size observed in these cultivars. With 1 annotation per time point, with 9 observers this would have an effect of reducing annotation effort per-observer to 11.1% of the dataset (i.e. 14-15 plants per each observer). As expected the more observers the better; but sampling only from experienced observers did not necessarily outperform

Table 7.3: A simulated citizen-powered experiment. P-values corresponding to an ANOVA test randomizing the number of observations available per each plant at a specific time point. Process is repeated sampling from any of the observers (i.e. the sampling may contain a mix of experienced and non-experienced observers) or only from experienced (ExP) or non-experienced (i.e. NExP) ones.

	K	min	max	mean	std	kurtosis
any	1	0.00003	0.00819	0.00124	0.00113	10.34
any	2	0.00002	0.00729	0.00120	0.00112	8.98
any	3	0.00010	0.00235	0.00061	0.00032	6.49
ExP only	1	0.00000	0.00726	0.00102	0.00103	9.58
ExP only	2	0.00004	0.00306	0.00057	0.00040	9.29
ExP only	3	0.00008	0.00150	0.00047	0.00021	5.35
NExP only	1	0.00008	0.00378	0.00100	0.00065	5.71
NExP only	2	0.00023	0.00174	0.00078	0.00028	3.49
NExP only	3	0.00033	0.00124	0.00069	0.00015	3.19

sampling only from non-experienced ones. Given the leptokurtic characteristic of these distributions (high kurtosis), the distributions are highly peaked around the mean with values concentrating around these. Overall, while the max indicates the worst expected result, results around the mean are to be expected as more typical.

7.3.8 Results from the Citizen-Powered Study

The study was launched on May 1st 2017, and by June 1st, approximately 5000 user annotations were available on a dataset of 1248 images, including the 130 RPi images used in this paper, with each image having at least 3 user annotations. Data were extracted from the Zooniverse database and a similar statistical analysis as to the one outlined above was carried out.

Of the 5000 annotations 4 Zooniverse users were responsible for annotating close to 10% of the data, as we can see in Figure 7.5(a).

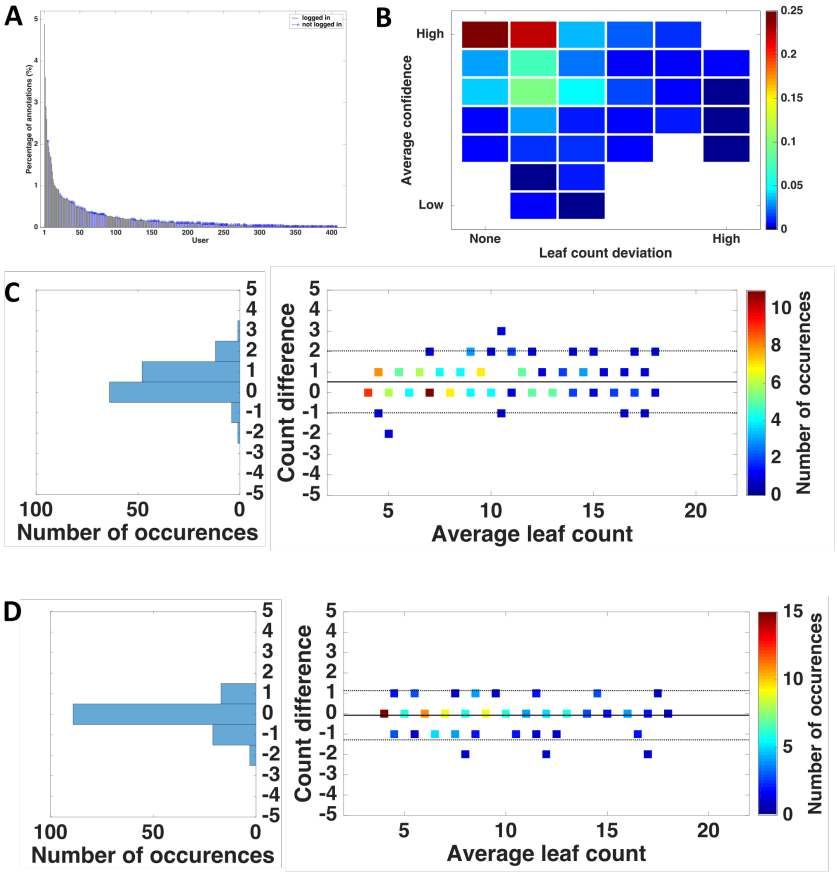


Figure 7.5: Citizen distribution and variability. A) Number of images annotated per user (citizen); B) Relationship between leaf count variation and average user confidence per plant; C) Variability between the consensus citizen and the reference observer; D) Variability between the consensus citizen and a random selection of counts (from the 3 available per-plant).

Most users contribute few annotations (long tail to the right), and not surprisingly most of the users are logged in (shown as black stem line without a marker in Figure 7.5(a)), which implies that they are frequent contributors to the platform.

Of particular interest is to explore if the self-reported confidence (answering the question on whether they believe they have annotated all leaves) relates to the spread of leaf counts among users for each plant. Figure 7.5(b) shows a two dimensional histogram of the per-plant standard deviation of the reported leaf count among the users with none referring to 0 standard deviation (i.e. annotations agree fully) and the average confidence (averaging the confidence question) for each plant of the 130 used in this study. An average of 3 shows high confidence (y-axis) vs. an average of 1 low confidence (y-axis). Colour encodes probability of occurrence. Users tend to agree with each other and their self reporting of confidence appears to be consistent with their spread in counting leaves, since the upper left quadrant sums to approximately 70% of occurrences.

We then estimated a consensus citizen by averaging counts across the annotated counts for each plant. We compared this consensus against the reference observer (from our controlled study) and a random single selection of counts, which can be seen as selecting one count per plant out of the 3 citizen provided counts (shorthanded as sing. random in Table 7.1). The results of this analysis are shown in Figure 7.5(c,d) respectively. We see what there is some variability among the reference observer and consensus citizen (Figure 7.5(c)), with the latter underestimating counts (see also related entries of DiC in Table 7.1). On the other hand variability appears to be smaller within citizens (c.f. Figure 7.5(d) and entries in Table 7.1).

Admittedly of most interest is to understand if citizens can be used for actual phenotyping. We use the counts of the consensus citizen and plot as previously averaged (and one standard deviation) per cultivar counts as a function of time in Figure 7.4(d). We can see that this plot closely resembles the others and particularly

the one of using only non-experienced observers in our controlled study. Equally the corresponding ANOVA experiment (last row in Table 7.2) shows exactly the same findings since using the consensus citizen counts yields a p-value still statistically significant, albeit larger compared to the one of the controlled experiment. However, a key difference between the two exists: in our controlled study all observers rated all images, so perhaps fixed effects of each observer may be captured in the intercept. Instead in the citizen experiment all counts come from a large pool of observers. In fact, when we compare the p-value of the consensus citizen ($p = 0.0024$) it is within the min-max bounds we find in our simulated study reported in Table 7.3.

Post-hoc, i.e. knowing that citizens underestimate, underestimation reaches 0 if we use the maximum across annotated counts (instead of average), and several other metrics improve including the p-value of the ANOVA. In Table 7.1 and Table 7.2 this is shown as consensus (max).

7.3.9 Variability Between Algorithmic Leaf Count and Experts

In addition to manual counting, we also tested a well-known leaf counting algorithm [3,7] to assess whether algorithm error is within (or outside) human variation.

For this experiment, we used the plant images used in [3], with annotations performed by experts not involved in other aspects of this study. Overall, this dataset contains 1,248 individual images of plants, taken from five different cultivars (*col-0*, *pgm*, *ein2.1*, *ctr*, and *adh1*). Specifically, images of *ctr*, *adh1*, and *ein2.1* cultivars were used as training set (728 images in total), whereas the images of *pgm* and *col-0* cultivars, which were also used in this study, were employed as testing set (130 images in total). From the training images, we learned a plant descriptor that derives image features and the projected leaf area to learn a non-linear model to predict the leaf count. It is noteworthy that the training set contains cultivars

Table 7.4: Algorithmic leaf counting results obtained using the method in [7]. Four metrics are reported. We first compare between the algorithm and the 728 images in the training set (ie. how well the algorithm learns). Then we compare how well the algorithm predicts counts on a testing set of 130 images (also used in this study) comparing the algorithm with the counts of the annotator (that also was involved in deriving annotations for the training set). Lastly we compare the annotator (the data of which we used to train the algorithm and was not involved in this study) with the reference observer used throughout in this study.

	Algorithm vs. annotator Training error	**Algorithm vs. annotator Testing error**	**Annotator vs reference Inter-observer error**
DiC ↓	0.00 (1.07)	-0.04 (1.31)	0.21 (0.75)
|DiC| ↓	0.61 (0.88)	0.88 (0.96)	0.46 (0.62)
MSE ↓	1.163	1.700	0.600
R^2 ↑	0.933	0.895	0.964

not included in the testing set, which makes this learning protocol the most stringent condition as the algorithm has never seen the mutants. After the model was trained, we calculated the evaluation metrics in [3] in the training (728 images) and testing sets (130 images). In addition, since the expert observer that labelled the images used to train the algorithm was not part of this study, we also computed the disagreement between this expert and the reference observer used throughout this study.

As shown in Table 7.4, the algorithm learns well (agreement between algorithm and annotator on the 728 training images the algorithm was trained on). When predicting counts on the 130 test images, the algorithm performs slightly worse when compared with the same annotator involved in labelling the training set (middle column). However, we can see that the algorithm is within inter-observer variability which compares two expert annotators (last column in Table 7.4). While on average the algorithm predicts the correct leaf count on some images (mean close to zero) it appears

that it is over or under-estimating counts on some, which explains the high standard deviation and high MSE. We note that here the algorithm carries two sources of variation (error): one of the annotator and one of the learning process itself. The latter can be minimised, but the former unfortunately is harder to do so unless a mixture of annotators are used.

7.4 Discussion

We showed that intra-observer variability remains low with experienced observers but non-experienced ones tend to vary more in their second repeat reading using a visualisation tool. Our annotation tool helps to retain mental memory and to reduce fatigue overall lessening the potential for errors when plants become larger and have more leaves. At the same time we showed that higher image resolution helps, but not always with the same effect: higher resolution aids the experienced user to find more of the smaller leaves but non-experienced ones missed them more often independently of resolution. Inter-observer variability is not significantly greater than intra-observer variability. Overall observers tend to be within plus/minus one leaf almost 80% of the time.

This agreement seems appealing but it might be random in nature and we explored if it affects the use of observers in actually identifying group differences in longitudinal counts. Repeat statistical tests showed that when we use one or more experienced or non-experienced observers we still come to the same statistical conclusion using an ANOVA test on the same longitudinal cultivar comparison: we find, as expected, differences in trends between *col-0* and *pgm* as reported previously on the same data [3] (c.f. Figure 7.4 on page 147). Whether we use only experienced or non-experienced observers has minimal effects on the statistical inference of the test.

Encouraging are the investigations using simulated and real data from citizen-powered experiments. In real experiments we cannot ensure the composition (in expertise) of the participating users and

neither can we assume that the same user will annotate all the data. However, our analysis on simulated data (where we can control the composition) showed that having even 1 annotation per plant can be sufficient to arrive to the same statistical conclusion (differences in cultivar trends) but of course having more is better, reducing variation. These findings held also in the real citizen-powered experiment based on the Zooniverse platform. Leaf counting based on algorithms while showing promise and progress does not yet meet human performance necessitating further investigation in the area; thankfully, collation studies [67] and challenges (e.g. the counting challenge of the CVPPP workshop series `https://www.plant-phenotyping.org/CVPPP2017-challenge`) on open data [6] will help advance the state-of-the-art.

In this study, consensus was obtained through averaging across annotations and treating time points independently, but alternative mechanisms can be used to establish more consistent longitudinal counts. For example, one can adopt several other consensus approaches that are data-agnostic [181] or if we assume that leaves always emerge or remain the same in a succession of images but cannot disappear, consensus can be derived using a dynamic filtering approach. Alternatively, machine learning algorithms can be used to learn directly from such repeated and imprecise (in machine learning speak: noisy) annotations potentially also obtaining consensus estimates which should also help eliminate observer bias. However, in machine learning much effort has been devoted to noisy annotations in classification tasks [172, 173] but regression is a yet unexplored area. A more radical approach, is to alter the design of the annotation task completely: for example, users can be shown pairs of images and can be asked to identify only 'new' leaves (if any at all). Irrespective of the design of the annotation task, minimising the amount of data requiring annotation by selectively displaying (to the observers/annotators) only images that do need annotation is always desirable. This has strong links to active (machine) learning [182] which displays images that are the most informative from a machine learning perspective. Integrating this may be possible

within a controlled lab annotation platform (as for example with the CellProfiler [182] software) but doing so in Zooniverse is not straightforward as images used in the work-flow cannot be altered on the fly and a customised platform would be required.

Considering all these findings, we can conclusively argue that while there is some variability among observers it is minimal when evaluating quantitative traits like counting objects, even of very different sizes. For the group (cultivar) effect sizes observed here this variability had no effect in statistical inference. Clearly, not all of these findings generalise to all (possible) human annotation tasks. Findings on 'negative effects', i.e. factors increasing annotator variability, like fatigue, lack of suitable annotation tools etc. can be expected to be also present for harder annotation tasks being more challenging for humans. They are expected to generalize well. However, 'positive effects', e.g. observed discriminative power of human annotations for the investigated task, cannot as easily be generalized to other, especially more difficult tasks.

At the same time common citizens, empowered by easy to use platforms, can greatly assist the effort of annotating images; at least, when the overall task is broken down in elementary sub-tasks generally doable even by non-experts without detailed explanations. Then common citizens can be used to provide annotations and drive phenotypic analysis. Such annotations help to develop and evaluate automated algorithms and allow to train machine learning-based solutions. Using such platforms a higher annotation throughput can be met than perhaps available locally in a lab, reducing significantly annotation effort.[2] It is time to consider how we can motivate the participation of citizens and design annotation tasks that can provide data of sufficient quality for other phenotyping tasks. This will have not only an effect on phenotyping but also on introducing this societally important problem to the broad public.

[2]We emphasize that Zooniverse is not an annotation platform per se and any workflow presented should have a strong ethical and reward mechanism to be accepted as a Zooniverse project. For tasks with a demanding rate and purely annotation objective gamification and crowdsourcing should be selected.

This approach, together to the annotation tool presented in Chapter 6, has an important drawback. The manual annotation of images needs real images of plants. Although efficient in terms of the reliability of the annotation, the collections of plant images requires a considerable amount of time. Taking as example the lifetime of an Arabidopsis plant (c.f. Figure 2.1), several weeks are required for the plant development. Once images are collected and plants are individually cropped, the annotation collection via Zooniverse needs the setup of a science project on the web portal. Only this process required approx. 2 months. Once the project is accepted by the Zooniverse community, several weeks have to pass in order to give time to the volunteers to annotate the images. Therefore, from the day of the sowing to the end of the data collection, it is likely that a semester has passed.

Due to this limitation, in the next chapter we present a deep neural network to artificially generate images of Arabidopsis plants, using a conditional Generative Adversarial Network (GAN). We learn the data distribution of the CVPPP 2017 dataset (only Arabidopsis plants, c.f. Table 6.1). This allows us to sample from this distribution to synthesise images of plants with a given leaf count. This approach can reduce the time to release new plant datasets, by producing artificial photo-realistic data.

8

Arabidopsis Rosette Image Generator (through) Adversarial Networks

In this chapter, we will show how to generate synthetic images of *Arabidopsis* plants using deep neural networks. Here, the first deep learning algorithm for plant phenotyping in this thesis is presented, anticipating the next part, which accomplishes leaf counting with deep architectures.

An attempt to generate rosettes was done in [183], where an empirical model was created analysing 5 *Arabidopsis* plants of about 11 leaves. Sigmoidal growth models were fitted based on the sets of plants under their study. Then, organs were dissected and leaves were used to fit B-splines to obtain vector images. A dataset of such generated *Arabidopsis* plants has recently been released in [80]. The issues of this model can be summarised as follows: (i) the approach is confined on the manual observation of morphological

This chapter is based on:

- M. V. Giuffrida, H. Scharr, and S. A. Tsaftaris. "ARIGAN: Synthetic Arabidopsis Plants using Generative Adversarial Network", *Proceedings of the Computer Vision Problems in Plant Phenotyping (CVPPP)*, ICCVW, 2017.

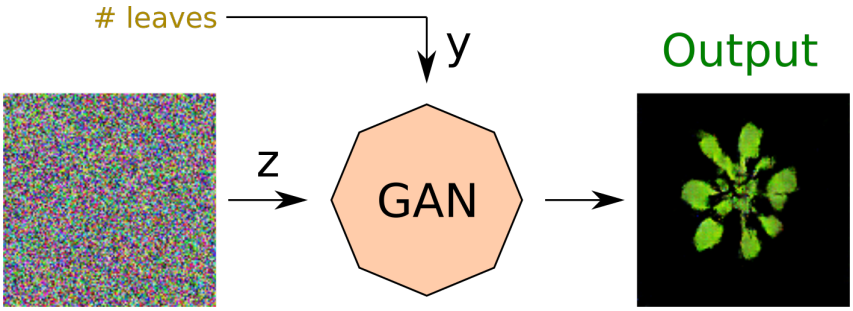


Figure 8.1: Schematic of the proposed method: a conditional generative adversarial network is trained to map random uniform noise z into *Arabidopsis* plants, given a condition y on the number of leaves to generate.

traits of a limited set of plants; and (ii) there is a lack of realism in the generated images, that is the absence of texture on the leaves.

Here, we show how *Generative Adversarial Network* (GAN) [100] can be trained to produce plant images. Although adversarial networks have brought many benefits, a main limitation is the lack of direct control over the images generated. For instance, in the case where we want to train a GAN to generate images of handwritten digits from the *mnist* dataset [184], it would be reasonable to have control over which digit to generate each time. For this reason *Conditional GAN* [185] was proposed to overcome such limitation. In this new formulation, generator and discriminator networks are endowed with an additional input, allowing them to be trained under certain conditions. In [186], the authors propose *StackGAN*, a two-stage GAN conditioned on image captions. Specifically, Stage-I generates coarse images, which are then provided to Stage-II to obtain more realistic images.

Here, we will show how to generate *Arabidopsis* plants, using a model inspired by [187], trained on the CVPPP 2017[1] dataset.

[1]Available at the following URL: `https://www.plant-phenotyping.org/CVPPP2017`

The network learns how to map random noise z into an *Arabidopsis* plant, under a condition y. For our purposes, y encodes the number of leaves that the artificially generated plant should have. The employed model, which we call *Arabidopsis Rosette Image Generator (through) Adversarial Network* (ARIGAN), is able to create 128×128 RGB images of *Arabidopsis* plants, as shown in Figure 8.1. We evaluate our model by generating an Ax dataset (using the CVPPP dataset name convention) and provide this data to the leaf counting algorithm presented in Chapter 4 to augment the training dataset.

8.1 Generative Adversarial Networks

In a GAN, there are two networks (agents) competing with each other: the *Generator* (G), which creates artificial images; and the *Discriminator* (D), which is trained to classify images coming from the training set (real) and the generator (fake). The spirit of the GAN is to improve G to create more realistic images, whilst D is trained to distinguish between real and generated images. Training works by improving in alternating fashion G or D, until an equilibrium is obtained. Generally speaking, the generator and the discriminator can be any network that satisfies the following criteria: (i) D needs to take as input an image and has to output '1' and '0' (real/not real); (ii) G needs to take as input random noise (e.g. drawn from an uniform or normal distribution) and has to give as output an image. LAPGAN [188] was proposed, which was able to produce better quality images using Laplacian pyramids. A new successful adversarial network providing outstanding results is *Deep Convolutional GAN* [187]. The benefits of this model mostly stem from the use of convolutional/deconvolutional layers for discriminator and generator respectively and the lack of pooling/upsampling layers.

8.2 Methodology

To generate images of *Arabidopsis* plants, we followed the DCGAN architecture [187], but have added an extra (de)convolution layer to generate 128×128 images. In Figures 8.2 and 8.3, we show the generator and discriminator model respectively. Both networks share the same layer structure in reverse order. We provide further details in the next sections.

8.2.1 Mathematical Background

A generative adversarial network has two models that train simultaneously: the generator G and the discriminator D. The generator network takes as input a random vector $z \sim p_z(z)$ and learns the set of parameters θ_g to generate images $G(z; \theta_g)$ that follow the distribution of real training images. At the same time, the discriminator D learns a set of parameters θ_d to classify $x \sim p(x)$ as real images and $G(z; \theta_g)$ as synthetic (or fake) images. The training process maximises the probability of D to assign the correct classes to x and $G(z)$, whilst G is trained to minimise $1 - D(G(z; \theta_g))$. Using the cross-entropy as loss function, the objective $V(D, G)$ to be optimised is defined as follows:

$$\begin{aligned} \min_G \max_D V(G, D) = &\mathbb{E}_{x \sim p_{data}(x)} \left[\log D(x)\right] \\ &+ \mathbb{E}_{z \sim p_z(z)} \left[\log(1 - D(G(z)))\right]. \end{aligned} \tag{8.1}$$

The optimisation of (8.1) can be done via stochastic gradient descent, alternating the update of θ_d and θ_g.

In order to control the image to generate, we add $y \sim p_y(y)$ as input that embeds the condition [185]. Therefore, we have a set of real data (e.g., training set) $\mathcal{D}_r = \{(x_i, y_i)\}_{i=1}^n$ for the discriminator and a set of sampled data $\mathcal{D}_s = \{(z_i, y_i)\}_{i=1}^n$ to train the generator. Hence, we update (8.1), such that $D(x; \theta_d)$ becomes $D(x|y; \theta_d)$ and $G(z; \theta_g)$ becomes $G(z|y; \theta_g)$.

The two networks, the generator and discriminator, could be networks of any architecture. In the next sections we provide details

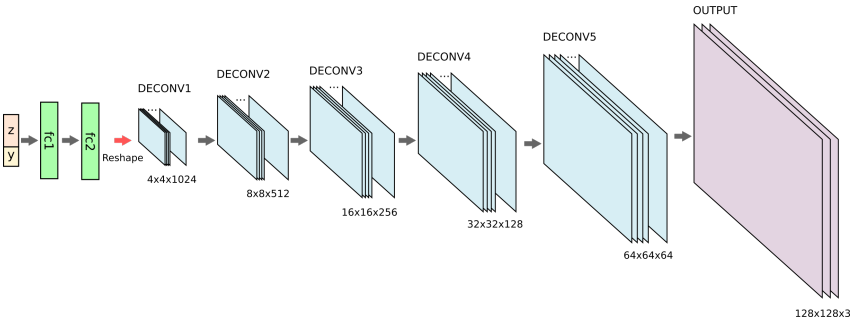


Figure 8.2: The generator takes as input a variable z (random noise) together with the condition vector y. These inputs are then provided to two fully connected layers, where *fc2* has the same amount of hidden units of the first deconvolutional layer. The information is then processed through 5 deconvolutional layers, where the last one provides an 128x128 RGB image. The condition y is applied to all the stages (fc and deconv layers). We showed it in the input only for sake of clarity.

about G and D. We used convolutional deep networks to generate images of *Arabidopsis* plants.

8.2.2 The Model G

Our model is inspired by [187], although we also added an additional (de)convolutional layer to obtain 128×128 images. The original model generates 64×64 images which is not suitable for *Arabidopsis* plants synthesis, where young plants might be only a few pixels in size and mostly indistinguishable.

The input layer takes a random variable $z \sim \mathbb{U}[-1, 1]$ concatenated to a variable y that sets the condition on the number of leaves. A typical approach for the condition is to use a one-hot encoding over the number of classes. We followed this approach, by considering the number of leaves as a *category* on which a condition should be set, where C denotes the number classes. Hence, a vector

$y \in \{0,1\}^C$ will have all zeros, except for a '1' located at the position corresponding to a certain class of plants. The condition y within the training set $\mathcal{D}_r$ corresponds with the ground-truth leaf count, whereas the y in $\mathcal{D}_s$ is randomly sampled, such that $y_t = 1$, where $t \sim \mathbb{U}[1, C]$ (namely, the '1' is located in a random location and the rest of the vector is filled with zeroes).

The so-formed input is then provided to two fully connected layers, denoted as *fc1* and *fc2*. The output of *fc2* matches the size of the filters for the *deconv1* layer, such that the output of the last fully connected layer can be easily reshaped. After 5 deconvolution layers, a $128 \times 128 \times 3$ output layer with *tanh* activation function will present the generated plant image. We do not employ any upsampling, but we use $(2, 2)$ stride instead, such that the network learns how to properly upscale between two consecutive deconvolutional layers. We adopted 5×5 filter size on all the deconvolutional layers [187]. Furthermore, the output of each layer is normalised [189] and passed through ReLU nonlinearity, before it is provided to the next layer. Similar setups also hold for the discriminator model. Although not graphically reported in Figure 8.2, the condition y is concatenated throughout all the steps of the network. In fact, each output of the fully connected layers has the vector y added. The deconvolutional layers also have the (leaf count) conditions as additional feature maps, spatially replicating y to properly match the layer size.

8.2.3 The Model *D*

Figure 8.3 visualises the discriminator model. It can be seen as a inverted version of the generator, where the order of the layers is flipped and deconvolutional layers are replaced with convolutional ones. Also for this model, as discussed in Section 8.2.1, the condition y is embedded at all stages of the network. Here, the last layer of the network is a single node that outputs a binary value (fake vs. real images), activated with a *sigmoid* function. Differently than *G*, the discriminator uses *Leaky ReLU* [190] as nonlinearity at each layer of the network [187], which has been shown to provide better

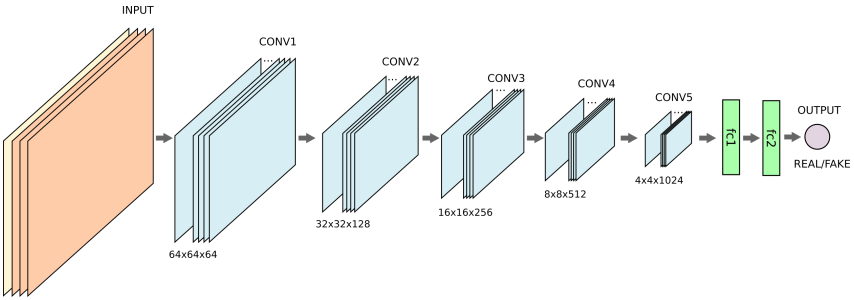


Figure 8.3: The discriminator takes as input an RGB image concatenated with the condition vector y properly reshaped to be stacked as an additional channel. The rest of the network is a reversed version of G (c.f. Figure 8.2). The last node of the network is a binary classifier that discriminates between real and generated (fake) images.

convergence for classification.

8.3 Experimental Results

In this section we show the experimental results of training the ARIGAN. We implemented the network on Theano [191] and the training was done on a NVidia Tesla K8 GPU. Training takes ~ 10 minutes per epoch on our setup.

8.3.1 Dataset

We used the CVPPP LCC 2017 plant dataset to train our model. These plant images are taken from different publicly available datasets [6, 16], containing *Arabidopsis* (A1, A2, and A4) and Tobacco (A3) plants. Specifically, the training annotated datasets are:

1. **A1:** 128 *Arabidopsis* Thaliana Col-0;
2. **A2:** 31 *Arabidopsis* Thaliana of 5 differents cultivars;

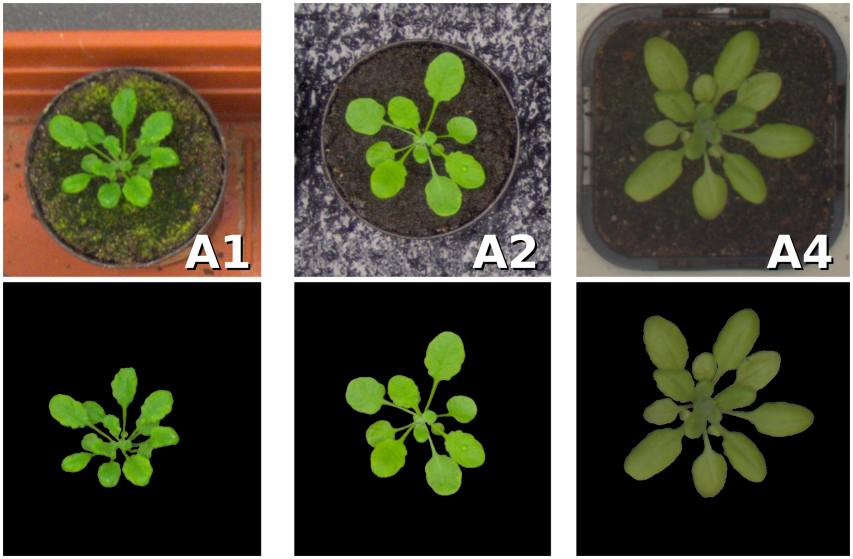


Figure 8.4: Images used to train ARIGAN. We used the segmentation mask to relax the learning process, due to the significant variability of these setups.

3. **A3:** 27 Tobacco plants;
4. **A4:** 624 *Arabidopsis* Thaliana Col-0.

For our purposes, we did not use A3 (Tobacco) dataset, due to the restricted number of images (27), compared to the *Arabidopsis* plants. Hence, we trained our network with the A1+A2+A4 dataset (c.f. Figure 8.4), containing a total of 783 images. Even though this number of training images is much bigger than in the previous CVPPP 2015 challenge,[2] it is still low for training a deep neural network with many parameters.[3] To overcome this issue, we performed dataset augmentation, by rotating the images by ten equidistant

[2]The A4 dataset was not included in 2015.

[3]Typically GANs are trained with data in order of magnitude of thousands or millions images [187].

angles of the range $[0, 2\pi)$, we also applied horizontal and vertical flipping to further increase variability, obtaining an overall 30-fold increase in the number of training images.

Input images were pre-processed to be all in the same size of 128×128 by cropping (to be made square) and rescaling. In order to match the output values of the *tanh*, we mapped the range of values from $[0, 255]$ to $[-1, 1]$.

8.3.2 Parameter Selection

We adopted similar parameters to those in [187] for our model. Specifically, we used an initial learning rate of 0.0002. For the Adam optimiser [192], the momentum β_1 was set at 0.5 and a L_2 weight decay is used and set at 10^{-5}. During training, a mini-batch of 128 was used and the discriminator was optimised after every update of the generator.

8.3.3 Qualitative Results

In Figure 8.5, we show generated plants at different training epochs. Specifically, we sampled a single random input along with a random condition and we gave it to the generator network (c.f. Figure 8.2). It can be seen that a clear *Arabidopsis* plant is obtained in about 30 epochs. In can be observed that some images have a light green appearance (typical of A1 and A2 dataset), others have dark green (A4), and some others have a mixture of those. Lastly, we see that at some epochs, the synthesised image contains a mixture of light and dark green texture, blended together seamlessly. However, generated images lack high frequencies, causing absence of some details, e.g., leaf veins or petiole.

Encouraged by these results, we extracted a subset of images to create a new dataset. We do this to quantitatively evaluate this artificial data using a state-of-the-art algorithm for leaf counting [7].

Figure 8.5: Fixed sample noise z is provided to the generator during training of ARIGAN. The number reported in the bottom right corner of each image refers to the epoch number.

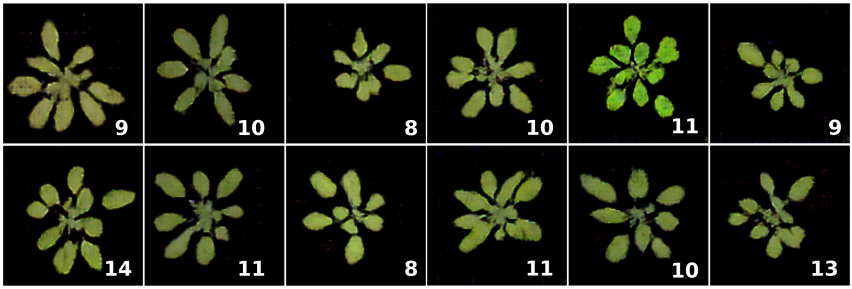


Figure 8.6: Samples from the Ax dataset generated with ARIGAN. Bottom-right numbers refer to the leaf count.

8.3.4 The *Ax* Dataset

Using our model, we artificially generated a new dataset of images. Following the nomenclature policy used for the CVPPP workshop, we decided to call it *Ax*. Specifically, we collected 57 images from our model some samples of which are shown in Figure 8.6. We collected the images at different stages of learning (we trained at least 20 epochs), providing random noise and conditions. Then we named the files using the *plantXXX_rgb.png* format, listing them in a CSV file with the count of leaves. We did not provide plant segmentation masks, as our algorithm does not generate background.

Quantitative Results

We evaluated the Ax dataset to train the leaf counting algorithm in [7]. For our experiments, we used the A4 dataset, as it contains the most number of images (624). The evaluation was done using 4-fold cross validation, where 468 images are randomly selected for training, and the remaining 156 for testing. We kept SVR parameters at their standard values, except $C = 3$.

In Table 8.1 we show the results of these experiments, using the evaluation metrics employed in [3]. Specifically, the table reports the results using A4 dataset only (top set of lines), paired with

Table 8.1: We trained the leaf counting algorithm in [7] using A4 dataset only (top set of lines) and A4+Ax (bottom set of lines). Results obtained with 4-fold cross validation. Results for DiC and |DiC| are reported as *mean* (*std*).

	Training Error	**Testing Error**
	Trained on A4 only	
DiC	0.013 (0.185)	0.147 (1.362)
|DiC|	0.026 (0.183)	0.942 (0.992)
MSE	0.031	1.865
R^2	0.999	0.947
	Trained on A4 and Ax	
DiC	0.229 (0.370)	0.186 (1.253)
|DiC|	0.042 (0.368)	0.891 (0.899)
MSE	0.137	1.596
R^2	0.996	0.955

the results using Ax from training as well.[4] Overall, considering that the leaf counting algorithm in [7] has not been demonstrated to work when different training sets are provided, we found that the additional dataset Ax improves the testing errors and reduces overfitting.

8.4 Discussion

This work aims to alleviate the lack of training data in plant phenotyping, we use a generative model to create synthetic *Arabidopsis* plants. Recently, Generative Adversarial Networks [100] were proposed, which have been proven to create realistic natural images.

[4] Ax images were added as part of the training set. Specifically, for each split of the cross validation, 57 images were added to make a training set of 525 plants images.

Encouraged from their results, we wanted to train a GAN to generate plants. Using the CVPPP dataset (only A1, A2, and A4), we trained an adversarial network inspired by DCGAN [187] to generate synthetic *Arabidopsis* images. Our *Arabidopsis Rosette Image Generator (through) Adversarial Network* (ARIGAN) is able to produce realistic 128×128 colour images of plant. Our network is a *Conditional GAN*, where an additional input of the network allows to set a condition over the number of leaves of a plant.

From our experiments, we found that ARIGAN learns how to generate realistic images of plant after a few of iterations (c.f. Figure 8.5). This qualitative results led us to create a dataset of artificial *Arabidopsis* plants images. Therefore, we gathered 57 images that our network generated to make the *Ax* dataset, as displayed in Figure 8.6. We evaluated our synthetic dataset using to train a state-of-the-art leaf counting algorithm [7]. Our quantitative experiments show that the extension of the training dataset with the images in Ax improved the testing error and reduced overfitting. We run a 4-fold cross validation experiment on A4 dataset. Evaluation metrics of our experiments are reported in Table 8.1. Our synthetic dataset *Ax* is available to download at `http://www.valeriogiuffrida.academy/ax`.

This chapter concludes the data collection part of this thesis. Approaches to annotating and generating new datasets were studied and investigated in this part to increase the number of labelled data for machine learning algorithms. Deep neural networks require large labelled datasets to solve a specific task. The data collected in this part can be used to train deep networks for the leaf count. In the next chapter, we present an architecture that predicts the number of leaves, using deep learning. The major findings of this work are two-fold: (i) deep network can efficiently infer the leaf counting from plant images; and (ii) multi-modal learning can improve the learned model, thus increase the accuracy of the learnt model.

Part IV

Deep Learning for Leaf Counting

9

Pheno-Deep Counter: The Versatile Leaf Counting Deep Network

In the last two parts of this thesis, we showed and demonstrated that machine learning is a valuable approach for plant phenotyping. In the context of leaf counting, we want to map the image x to the count y, using a regression model. In Chapter 4, we proposed a pipeline than extracted patches, encoded the images into a holistic descriptor, and trained a non-linear regressor. Formally, using the notation in Equation (2.1), the feature extraction function $\psi(\cdot)$ and the task (e.g. regression) function $\varphi(\cdot)$ were independent blocks.

A deep neural network embeds in an end-to-end architecture the feature extraction $\psi(\cdot)$ and the task $\varphi(\cdot)$ within the same model. In fact, the power of deep learning stems from end-to-end training. The backpropagation algorithm [63] trains the network parameters Θ simultaneously, finding the most suitable image features for the

This chapter is based on:

- M. V. Giuffrida, P. Doerner and S. A. Tsaftaris, "Pheno-Deep Counter: a unified and versatile deep learning architecture for leaf counting", The Plant Journal, 2018.

task at hand (c.f. Section 2.3). For this reason, deep neural networks have recently been employed to address the leaf counting problem as well. These approaches essentially combine the task of finding suitable image features with the task of learning a good regression model relating image features to leaf count [51, 79, 80]. These approaches show significant promise, but each of these is specialised: a new model and network for each plant species or cultivar, imaging condition, etc. is required. In addition, all three approaches use only optical images, whereas different imaging sensors such as near-infrared or fluorescence are now also commonly employed in plant phenotyping [30, 193–195].

In this chapter, we introduce the Pheno-Deep Counter (shorthanded as *PhenoDC*), a multi-input deep network that combines information coming from different imaging modalities to count the number of leaves of rosette-shaped plants. In contrast to other approaches, we aim to build a single unified model that can be used for a variety of plants and imaging scenarios, where plants are seen from the top in a laboratory setting. Critically, we demonstrate that, by agglomerating data from a variety of sources, the model learns better (deep learning algorithms require large amounts of data [196]). Our approach also significantly enhances utility, as the same model can be used in a variety of scenarios and can be easily adapted for this purpose.

The main contributions of PhenoDC are multi-fold:

i. *Multi-modal model:* an architecture that benefits from, and can use, multiple imaging modalities, e.g. classical colour (RGB) and near-infrared images. We show that by combining information coming from multiple modalities, PhenoDC improves leaf count prediction. As an example, training our network with only RGB images, PhenoDC predicts the correct leaf count in 55% of the cases. Adding other modalities (e.g., near-infrared and fluorescence), the prediction accuracy increases to 88%.

ii. *Ease of adaptation to new settings:* our model can be easily

adapted to work with another imaging setup (still assuming top-view), either by simply specialising the network to the new task or performing data agglomeration. We show that with a handful plant images (regardless of the species tested), our network can be trained to count leaves for the new scenario. We showcase several experiments using images of Arabidopsis thaliana plants, as well as other plant species, such as Tobacco and Komatsuna (a Japanese vegetable) [17].

iii. *State-of-the-art performance:* our approach can predict the number of leaves in unseen images with an error of ±1 leaf in $\sim 80\%$ of the cases as compared to 57% in [7] closing further the gap in achieving human-level performance [46]. This improves further when multi-modal learning is used.

iv. *Nocturnal leaf counting:* we show that our network is also capable of counting leaves during the night with near-infrared images, extending the applicability throughout the diel cycle, a feature not yet addressed by any other methods.

We perform a comprehensive analysis and comparison with other methods using a variety of data sources (c.f. Table 6.1). This work also includes several experiments and discussion points to help elucidate how one can adopt such approach (e.g. how many annotated samples are required and how to collect annotations).

9.1 The Network Architecture

Our deep neural network, shown in Figure 9.1, has been designed with the aim to accommodate inputs of variable size. To achieve this, our architecture breaks down the task of counting into several sub-tasks. First, each image goes through a network that aims to find a fixed length vector representation to better describe a plant image. This is achieved by a sub-network (modality branch), where each input source is processed independently. However, during training the network learns what is useful to retain from

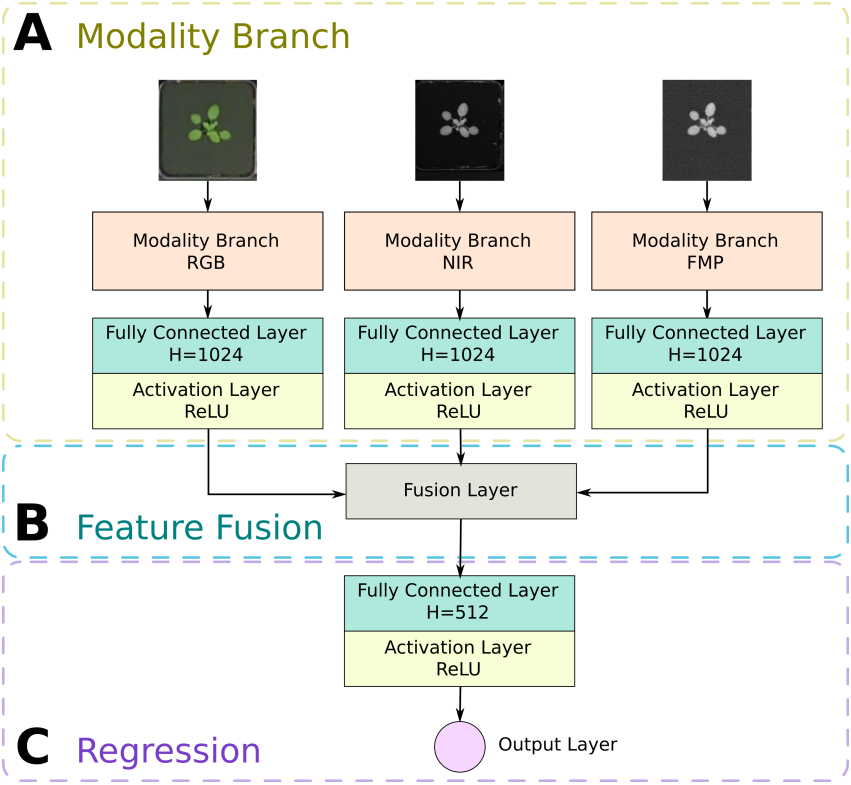


Figure 9.1: Schematic of the proposed deep architecture. (A) a modality branch, consisting of ResNet50 [14], extracts modality-dependent plant features as a feature vector of 1,024 neurons. (B) The fusion part combines those features to retain the most useful information from each modality. (C) The regression part, relates fused information with leaf count as a non-linear regression.

each modality, which results in an image descriptor (a vector per image) that jointly represents all the useful information. Multi-modal plant representation is accomplished by the feature fusion part of the architecture. Finally, the fused image descriptor is related to leaf count, by learning the parameters of a non-linear regression model between the descriptor and leaf count. After the network has been trained, evaluation of a plant's image(s) (the plural is used to denote the presence of different modalities) provides an estimate of the leaf count.

We optimise all computational blocks simultaneously to obtain a mapping between input images and leaf counts. For our purposes, we used up to three inputs: RGB, near-infrared, and fluorescence.

Modality branch. The sub-network that processes each input (c.f. Figure 9.1a). We used the ResNet50 architecture [14], as in [51]. Each input is processed independently from others and generates a vector representation specific to its input, ensuring meaningful and discriminative features. Each branch ends with a fully connected layer of $1,024$ neurons using a rectifier (ReLU) non-linearity, which allows the suppression of negative values during the process of feature extraction. Each input results in an output vector of the same size independent of input image size.

Feature Fusion. The process that combines information coming from all modalities to retain the most meaningful features. Following the concept in [197], we apply an element-wise max fusion layer. We display this segment of the network in Figure 9.1b.

Regression. The process of relating fused information to leaf count (c.f. Figure 9.1c). The output of the fusion layer is given to another fully connected layer of 512 neurons with ReLU activation function (c.f. Appendix A). At the end of the network, the output of the last layer is given to a single neuron that makes the actual prediction of the number of leaves. During training, we minimise the mean

squared error (MSE) between predicted leaf count and ground-truth. The model predicts real numbers and we round the leaf count to the nearest integer only at test time.

Training strategies. We employ three common training strategies to improve network training and performance. First, we initialise our network with pretrained parameters (rather than random ones), computed previously based on image recognition task [65]. Second, we use an L^2 regulariser in the last fully connected layer before the output (i.e. the regression component). This technique prevents the network from learning large weights which may produce unstable results. For all experiments in this paper, we set this regularisation constant to $\lambda = 0.02$. Finally, to artificially increase robustness to view changes (rotation, translation and position of camera), we perform dataset augmentation during training. Specifically, we apply random geometrical transformations to the training data (e.g., random rotations, zoom-ins, shifts). This helps the network to learn from more data without having to collect more data. Our network was trained using a learning rate of $\eta = 0.0001$. Validation set: One of the problems arising in network training is when to stop training. The typical approach in machine learning is to also use a small set of labelled data, called the validation set [105]. We therefore used an early stop criterion to interrupt the learning procedure, terminating the training after 10 epochs we observe that the validations error has started to get worse.

Image Preprocessing. While combining data across different sources (data agglomeration) has benefits, the images coming from different setups exhibit variations in intensity and size that need to be corrected. For instance, images in A1 [6] and images in A4 [16] were acquired with different cameras and different illumination conditions, although they may show the same plant species (Arabidopsis thaliana Col-0). To ameliorate variations in illumination, we perform histogram normalisation on all images and to standard-

ise image size we resize all images of a modality to the same size 320×320 pixels. For the multi-modal images [15], RGB images are too small to be provided to the RGB modality branch, as ResNet needs images at least of 200×200 pixels size. In this case, we up-sampled the images to 240×240 pixels, whereas the images from the other modalities were left unchanged.

Implementation details. We implemented our deep neural network using Keras [198], an open-source library for deep learning in python, with Tensorflow backend. We performed our training experiments in a machine with a TITAN X GPU. Note that such equipment is not necessary for fine-tuning and adapting our network to new experimental data.

9.2 Results

To showcase the performance of our approach, we employed four different datasets:

i. A special collection of the PRL dataset [6] and Aberystwyth dataset [16] that have been used in the latest CVPPP 2017 Leaf Counting Challenge (LCC); it contains five different sub-datasets (c.f. Table 6.1). These datasets contain RGB color images of four different plant experiments, using different plants (and different cultivars), growth conditions, and camera settings.

ii. The multi-modality imagery database for plant phenotyping [15], containing images of Arabidopsis thaliana Col-0 acquired in three different modalities (RGB, near-infrared, fluorescence);

iii. The RGB images in Komatsuna dataset [17];

iv. Nocturnal Arabidopsis plant images acquired using a near-infrared camera [18].

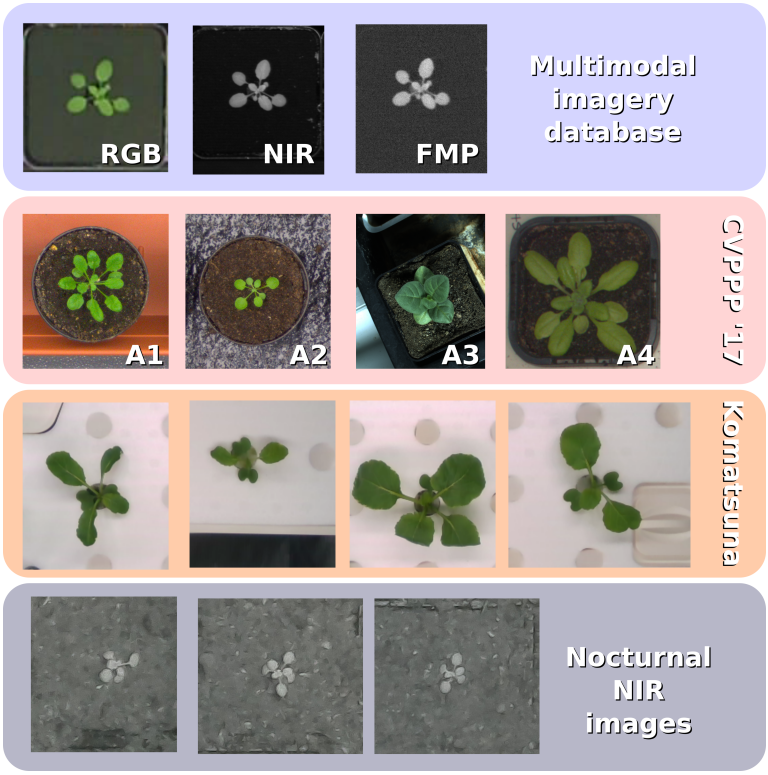


Figure 9.2: Sample images of the employed datasets. *First row:* RGB, near-infrared, and fluorescence images of the same plant from the multi-modal imagery database for plant phenotyping [15]. *Second row:* images from the A1, A2, A3, and A4 datasets from CVPPP 2017 [6, 12, 16]. *Third row:* samples of Komatsuna plants from [17]. *Last row:* samples of nocturnal images of Arabidopsis plants in [18].

Visual samples of these datasets are shown in Figure 9.2.

We used the evaluation metrics introduced in Section 2.4. We present a comprehensive set of experiments that demonstrate the reliability of PhenoDC for leaf counting. To train our model, data are split into (at least) two datasets, namely training and testing set. The training set is needed to optimise the set of parameters specifying our model. The testing set is required to evaluate the performance of the algorithm, using unseen data. In the following, in a series of experiments we show:

a. the benefit of data agglomeration across different sources;

b. the superior prediction performance in the recent benchmark CVPPP 2017 dataset;

c. that prediction error reduces when using multi-modal sources using the dataset in [15];

d. a set of experiments that demonstrate the flexibility of our network to adapt to other contexts, such as different plant species.

9.2.1 Proof-of-concept: data agglomeration helps

Herein, we aim to show that increasing data diversity in fact improves accuracy. We isolated the A1 set of images in the CVPPP 2017 dataset [6], which includes 128 images of Arabidopsis thaliana Col-0 for training. We followed the training procedure of [51], assessing the performance of our network using a 4-fold cross-validation, splitting randomly the training set with the following proportions: (i) 64 images for learning; (ii) 32 images for validation; and (iii) 32 images for testing. The validation set allows us to monitor model performance during training and prevents overfitting (the case where the model has essentially memorised the training set and therefore cannot adapt to new data). Using this learning protocol, the 4-fold cross-validation results are the following:

- DiC: -0.81 (0.85);

- |DiC|: 0.94 (0.70);
- MSE: 1.38;
- Percentage Agreement: 25%.

We proceeded to add more data drawn from the CVPPP 2017 dataset, namely the A2 (Arabidopsis thaliana of 5 genotypes), A3 (Tobacco), and A4 (Arabidopsis thaliana Col-0) set of images. As we progressed adding data, we observed that the mean squared error reduced by $\sim 50\%$. Specifically, he 4-fold cross-validation results with more training data are the following:

- DiC: 0.28 (0.80);
- |DiC|: 0.53 (0.66);
- MSE: 0.72;
- Percentage Agreement: 56%.

Finally, we wanted to evaluate which areas of an image contribute to the count. Ideally, the count produced by the network should only be influenced by regions of the image containing plant. Differently from other state-of-the-art methods [7,79], we do not provide per-plant segmentation masks during learning and inference to our network. Therefore, one can question whether the network is actually counting leaves, or if the prediction is influenced by unrelated regions of the images, such as background. To assess this, we employed the approach in [51]. Specifically, we mask part of the image with a 60×60 sliding window and see how this affects the leaf count. Ideally, when the window does not obscure parts of the plant (i.e. covers only the background), the leaf count should remain unchanged. In Figure 9.3, we display such an evaluation on a sample image of each of the four plant datasets in CVPPP 2017. We iteratively covered all possible locations of each image with this mask and observed that the largest contribution to the counting is very specific in the regions corresponding to plant leaves.

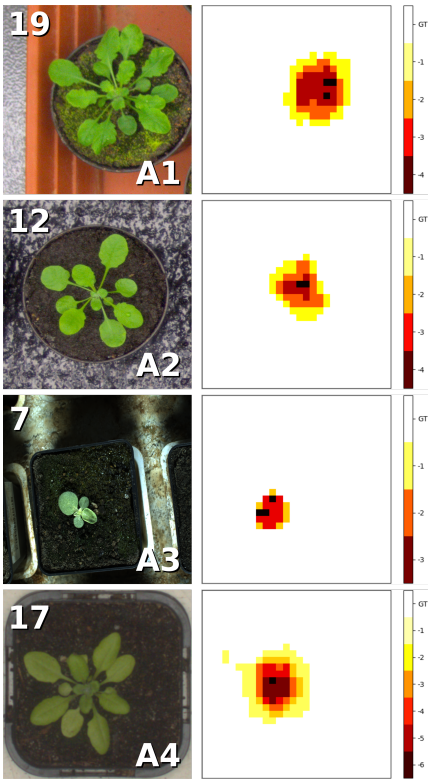


Figure 9.3: Visual diagram showing which part of a plant image contributes the most for the counting. We shift a 60×60 black patch entirely over a plant image and we show that areas corresponding to the plant gives the highest contribution to the count. In the top-left corner of each image we report the ground-truth (GT) leaf count of the plant.

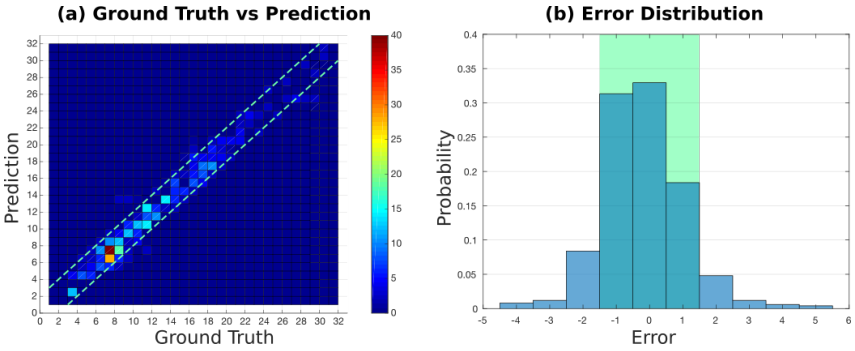


Figure 9.4: Leaf count prediction in the CVPPP dataset (all images altogether). (a) Ground-truth vs. prediction, shown as a scatter plot. Due to integer values color shows how many points are overlapping. Dashed parallel lines show the ± 1 leaf error range. Note that our approach has high agreement w.r.t. the real leaf count. (b) error distribution. Observe that there is 83% chance that the error will be ± 1 within 0 (green area), a number close to the agreement among human observers ($\sim 90\%$; c.f. Chapter 7).

This experiment highlights the benefit of data agglomeration, even when the sources are diverse. Since deep networks can form very complex functions (between input and output) more data is better and being "universal" is better than being specialised (for example one model per plant species) as it reduces the chance of memorisation.

9.2.2 Evaluation and comparison with state-of-the-art on the CVPPP 2017 dataset

In this experiment, we assess the performance of our network when trained on the heterogeneous CVPPP 2017 plant dataset and how it compares to state-of-the-art methods in the literature.

We report quantitative results in Table 9.1, comparing our performance with other deep learning methods for leaf counting [79] and

Table 9.1: Testing set results of PhenoDC trained on RGB images from the CVPPP 2017 dataset [6, 12, 16]. Evaluation metrics are detailed in Section 2.4.

		A1	*A2*	*A3*	*A4*	*A5*	*All*†
DiC	***PhenoDC***	**-0.39 (1.17)**	**-0.78 (1.64)**	**0.13 (1.55)**	**0.29 (1.10)**	**0.25 (1.21)**	**0.19 (1.24)**
	GLC [7]	-0.79 (1.54)	-2.44 (2.88)	-0.04 (1.93)	-	-	-
	[81]‡#	0.20 (1.40)	-	-	-	-	-
	[79]	-0.33 (1.38)	-0.22 (1.86)	2.71 (4.58)	0.23 (1.44)	0.80 (2.77)	0.73 (2.72)
|DiC|	***PhenoDC***	**0.88 (0.86)**	1.44 (1.01)	**1.09 (1.10)**	**0.84 (0.76)**	**0.90 (0.85)**	**0.91 (0.86)**
	GLC [7]	1.27 (1.15)	2.44 (2.88)	1.36 (1.37)	-	-	-
	[81]	1.10 (0.90)	-	-	-	-	-
	[82]‡#	0.80 (1.10)	-	-	-	-	-
	[79]	1.00 (1.00)	**1.56 (0.88)**	3.46 (4.04)	1.08 (0.97)	1.66 (2.36)	1.62 (2.30)
MSE	***PhenoDC***	**1.48**	**3.00**	**2.38**	**1.28**	**1.53**	**1.56**
	GLC [7]	2.91	13.33	3.68	-	-	-
	[79]	1.97	3.11	28.00	2.11	8.28	7.90
%	***PhenoDC***	**33.1**	11.1	**30.4**	**34.5**	**33.2**	**32.9**
	GLC [7]	27.3	**44.4**	19.6	-	-	-
	[79]	30.3	11.1	7.1	29.2	23.8	24.0

†A paired t-test between our method and [79] (the only two approaches from the CVPPP Workshop 2017) shows statistically significant differences (p-value < 0.0001).
‡Trained on A1 only.
#Training and inference are performed using per-leaf segmentations and not total leaf count as with the other methods.
Bold results show best performance.

leaf counting via segmentation [81,82], as well as with the machine learning algorithm in [7]. The CVPPP 2017 dataset contains as a subset data of previous competitions allowing comparisons across the years and methods (but not on all data). Overall, PhenoDC outperforms all others, scoring the lowest MSE error in all datasets (1.56). Note that the single input model of our deep architecture achieved the best results on the CVPPP 2017 dataset in the Leaf Counting Challenge (LCC). A paired t-test shows statistically significant gains when compared to [79] (p-value < 0.0001; last column of Table 9.1). Figure 9.4 collates results across all images as: (a) the correlation between ground-truth and prediction, showing high agreement of our method ($R^2 = 0.96$); (b) the distribution of error in leaf count, where it can be seen that in $\sim 80\%$ of the cases the error is confined within the ± 1 leaf range (for comparison in [7], report 57% agreement on the same range).[1]

In conclusion, PhenoDC is more reliable in terms of leaf counting, compared to the current state of the art approaches.

9.2.3 Multiple Modalities and Leaf Counting

In this section, we assess whether our network benefits from multimodal learning, leading to improved leaf count predictions. For this experiment, we used the dataset in [15], which contains images of Arabidopsis thaliana wild-type (Col-0) acquired using multiple sensors. In [15], they used 16 plants for 9 days, acquiring top-view images from 9am to 11pm (15 frames a day). This setup produced a dataset containing 2,160 individual images altogether, albeit only 576 images are annotated (images taken at 9am, 12pm, 4pm, and 8pm). Images were taken simultaneously in the following modalities: visible light (RGB), fluorescence (FMP), near-infrared (NIR), and depth. The multiple sensors acquired the same plants simulta-

[1] It is relevant to point out that, differently from our method, in [7], they used only the A1, A2, and A3 images. *PhenoDC* still has an accuracy of ± 1 leaf range $\sim 80\%$, when trained and tested on the same portion of the data to make fair comparisons.

Table 9.2: Testing performance of *PhenoDC* on the multi-modal dataset [15]. We report results when the network is trained using only a single input and when also using all inputs.

Training on	DiC	|DiC|	MSE	%
RGB Only	0.02 (0.75)	0.48 (0.57)	0.56	55.7
FMP Only	-0.06 (0.72)	0.45 (0.56)	0.52	58.7
NIR Only	0.13 (0.61)	0.33 (0.53)	0.39	69.6
All	0.11 (0.40)	0.13 (0.39)	0.17	88.5

neously. Due to the heterogeneity of such sensors and their placement, image resolution (and effective image size) and alignment vary. We excluded depth images due to their extremely low resolution (30×30 pixels), compared with the others. (Image samples are shown in Figure 9.2). We randomly split the labelled dataset into 3 parts (50% training, 25% validation, and 25% testing) and trained our network using 4-fold cross-validation.

To establish a baseline for our multi-modal results and to find the most useful single modality (for the counting task), we first trained our network using only one of the available modalities as input at a time prior to using all modalities. As reported in Table 9.2, we obtained the best single-input result using the near-infrared (NIR) images (MSE = 0.39). This is due to the fact that NIR images, in this dataset, are sharper and more detailed. To demonstrate this, we visualise the activations produced by our network for each of the modality branches. In Figure 9.5, we show the output of the first residual block [14] for three sample plants of the dataset (mean activation across the feature maps). Overall, most of the activations are focused on the region where the plant is located. Note that, while some pixels are active on the background on RGB or FMP, the IR activations are mostly dominant on the plant, which demonstrate the benefit of using multi-modal information. We obtained the best performance when all three inputs were used simultaneously: MSE

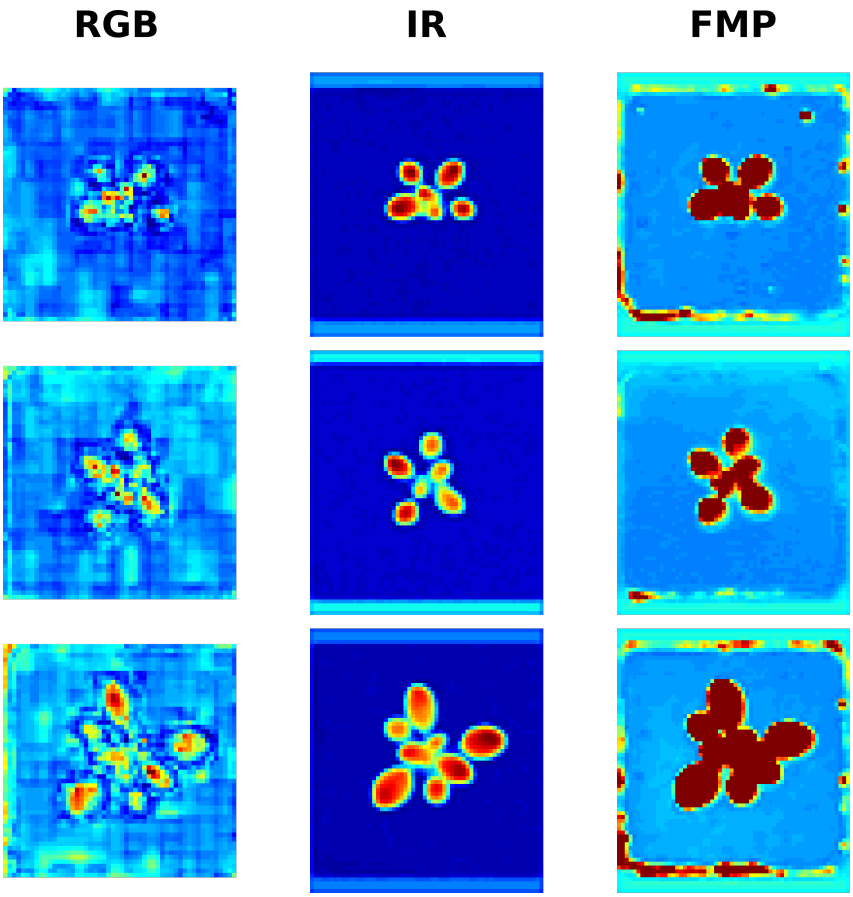


Figure 9.5: Activations after the first residual block in the RGB, IR, and FMP modality branches. The output of this block layer consists of 256 feature maps. We display the mean for each pixel.

was reduced by more than 50%, and Percent Agreement increased by $\sim 19\%$.

We conclude that combining information coming from multiple modalities improves counting accuracy. The fusion layer learns (c.f. Figure 9.1b) to retain the most useful image features coming from any of the modality branches (c.f. Figure 9.1c). These experiments highlight that multi-modal learning can be useful for plant phenotyping purposes, and that our architecture can handle any number of inputs.

9.2.4 Evaluation of Network Adaptivity Capabilities

In this section, we address the problem of how one can use PhenoDC by adapting to other experimental setups, different to the one used during training.

We rely on the principle of fine-tuning a pre-trained network to significantly reduce the number of new training examples required to adapt the network [199] and increase performance [196]. Fine-tuning entails the labelling of just few images and their use to update the parameters of a network that has been pre-trained to solve the same task but in a different context (e.g. different plant species).

We demonstrate this capability in three different cases using the following datasets: Tobacco plants (A3) from [6], the Komatsuna plants from [17], and other Arabidopsis cultivars using night-time images [18]. (Further details of all these image datasets are in Table 6.1 and Figure 9.2). For these experiments, we first pre-trained our neural network using only the Arabidopsis plant images A1, A2, and A4, in the CVPPP 2017 dataset [6, 16]. This training dataset containing Arabidopsis plants, as reported in Table 6.1, does not contain a large number of images, making the learning process challenging. The following experiments also were aimed to assess the number of training images required to adapt the network into another scenario.

Table 9.3: Fine-tuning of the parameters of PhenoDC on Tobacco images [6] previously pre-trained with Arabidopsis plants A1, A2, and A4 [6, 16]. We progressively increase the number of training images to find a suitable number of images required to create a meaningful model that can count Tobacco leaves. Below we report the results on the held-out testing set.

# of training images	DiC	| DiC |	MSE	%
7	-0.39 (1.65)	1.32 (1.07)	2.83	23.2
14	0.00 (1.32)	0.96 (0.90)	1.75	32.1
21	0.27 (1.36)	0.87 (0.90)	1.91	41.1
27	0.25 (1.20)	0.86 (0.87)	1.50	37.5

Tobacco plants [different species, imaging camera, and settings]

We fine tuned the pre-trained network using a variable number of tobacco training images. Specifically, we selected 7, 14, 21, and then 27 images to fine-tune the pre-trained network. The results of these experiments are reported in Table 9.3. Overall, we observe that more training data leads to better predictions in the testing set. As expected, the lowest error is obtained when we use all 27 images for training (MSE = 1.50). In Figure 9.6, we show the distribution of the error that we registered during progressive learning. As more images are used, the error distribution narrows around the 0. In fact, in $\sim 80\%$ of the data in the testing set our method is within 1 leaf error from the ground-truth (green areas in Figure 9.6), thus achieving more accurate predictions. Hence, we can conclude that fine-tuning with a handful of images ($>= 21$ in this setup), *PhenoDC* can produce reliable leaf count.

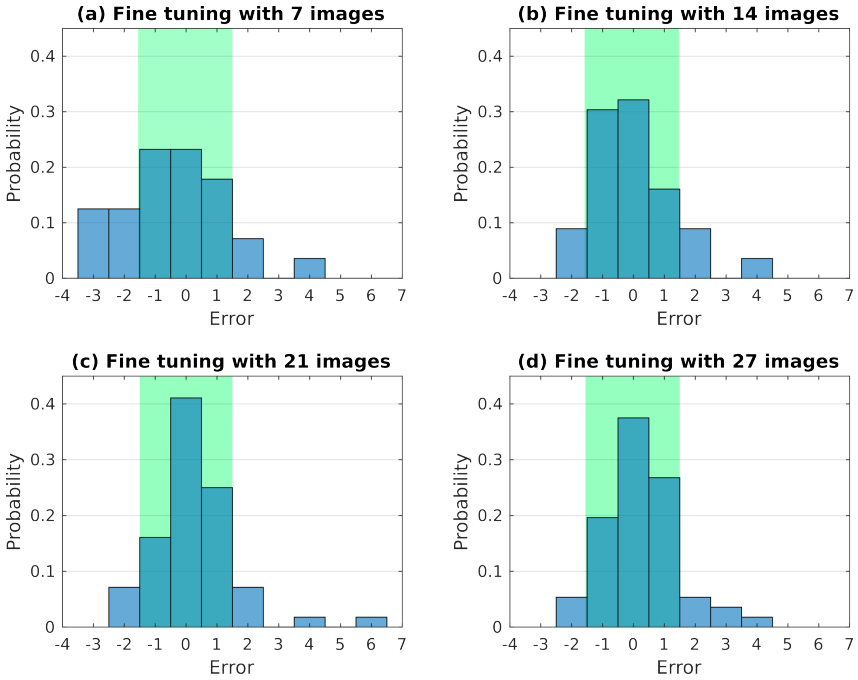


Figure 9.6: Error distribution of our network fine-tuned using Tobacco plants in A3 dataset [6]. We reported the distribution of the error committed in the testing set, after refining the network parameters with 7 Tobacco plants (a), 14 (b), 21 (c), and 27 (d) plants. When we train with more images (>=21), the green area (error up to ± 1 leaf, c.f. Figure 9.4) contains more than 80% of the cases.

The Komatsuna case [different species, imaging camera, and settings]

This dataset contains 300 RGB images of 5 different Komatsuna plants, 6 images/day for 10 days. (Images were taken from 3pm until 3pm every 4 hours). We split the dataset as follows (c.f. Table 6.1):

- training set: 2 plants (IDs 00 and 01), entire timeline (120 images);

Table 9.4: A similar process to that described in Table 9.3 but repeated for komatsuna plant leaf counting based on data available in [17]. The model has been trained on Arabidopsis as described in Table 9.3. Results shown refer to the testing set.

# of training images	Hours of the day	DiC	|DiC|	MSE	%
10	3 p.m.	-0.74 (1.08)	0.96 (0.89)	1.71	35.0
20	3 p.m., 11 a.m.†	-0.54 (0.95)	0.86 (0.65)	1.19	25.0
30	3 p.m., 3 a.m.†, 11 a.m.†	0.18 (0.92)	0.67 (0.66)	0.88	44.2
40	3 p.m., 3 a.m.†, 7 a.m.†, 11 a.m.†	0.24 (0.84)	0.59 (0.64)	0.76	49.1

- validation set: 1 plants (ID 04), entire timeline (60 images);
- testing set: 2 plants (IDs 02 and 03), entire timeline (120 images).

We fine tuned our pre-trained network by progressively increasing the training set size to 10, 20, 30, and then 40 images per plant, choosing time frames that followed the plant growth evolution. Overall, the results in Table 9.4 show that more data contribute to more accurate results. Predictions become very accurate when 40 images per plant are used during training, showing a reduction of the MSE by 50%, compared with training using 10 images per plant.

Nocturnal images of Arabidopsis plants [different cultivars, settings and modality]

Night images are usually acquired using infrared cameras and specific LED lights that illuminate the scene with near-infrared radiation (wavelength of 940nm which does not alter natural plant development; [15,18]). We selected and annotated a subset of night images from [18]. Specifically, we selected 18 plants and sampled one image per night every other day for eight days (totally 72 images). Examples of nocturnal images are shown in Figure 9.2. We pretrained the network using the NIR images from [15] and fine

tuned using 10 plants for training (40 images in total), 4 plants for validation (16 images), and the last 4 for testing (16 images). Since these images come from different ascensions of A. thaliana, we randomly changed the training/validation/testing set 4 times. Quantitative results on the testing error are: DiC: -0.14 (0.77); |DiC|: 0.52 (0.59); MSE: 0.61; and Percent Agreement: 53.1%. Overall, the error is very low (MSE < 1), demonstrating the utility of our machine learning approach to leaf counting during the night.

To summarise, these experiments demonstrated that *PhenoDC* can adapt to different scenarios of considerable complexity. Acceptable performance can be attained using few images (e.g. 14 in the case of Tobacco). In addition, by fine-tuning our network with Arabidopsis images acquired during the night, we permit plant growth analysis during the entire circadian cycle [193].

9.3 Discussion

In this chapter, we report *Pheno-Deep Counter*, a deep artificial neural network that can predict the total number of leaves from top-view plant images. We showed the effectiveness and reliability of our network architecture using several plant datasets. Specifically, we show that data agglomeration helps to improve accuracy: as more datasets were added, the mean squared error fell by 50%. A similar error reduction was also observed when the network was trained with multi-modal data, showing that combining information coming from multiple imaging sources helps to train a better regression model and to learn better features. We showed that our method can adapt to new settings and demonstrated that a refinement step, fine-tuning, can be used to achieve excellent performance even with only a few images for training. We also demonstrate that NIR modalities can be used to count leaves during darkness, permitting leaf counts for detailed plant growth analysis throughout the circadian cycle.

Our approach to leaf counting learns meaningful image features across all modalities and then relates features to leaf count via non-

linear regression. We train both aspects together, thus adapting image features whilst learning the regressor. Furthermore, our approach offers a single model to solve the same task for any input. Our robust and accurate neural network can be extended for new input/modalities without changing the overall architecture. This simplifies adoption and permits the sharing of model updates when new experiments have been made available on the basis of our architecture. Therefore, by placing our pre-trained PhenoDC and source code (and instructions) into the pubicly available at `https://bitbucket.org/tuttoweb/pheno-deep-counter`, we hope to accelerate the adoption of such methods in plant phenotyping analysis.

Our network was evaluated on top-down views of dicot rosette-shaped plants. Clearly, this is one setup, among many others. It is possible though that an ideal leaf counting algorithm would be able to work also on monocots, and even tree canopies with thousands of leaves, given enough training data. Unfortunately, we presently lack such curated datasets with these scenarios and we are unable to experimentally assess how PhenoDC would perform, albeit it still brings us a step closer towards generalisation.

In this work, we focused on "how many", rather than "which" annotated images, are needed to train a good regression model. It goes without saying that adequate image resolution and quality are necessary. Generally, images that show appearance diversity are good images to annotate. In a time-lapse setting, images spanning a set interval of the time series would be a good start. However, better approaches exist to find the best set of images to jointly inform the model, known as active learning. Active learning with neural networks is an ongoing research problem in machine learning. We previously showed, using plant descriptors and data mining [200], a promising potential in identifying images for annotation.

Furthermore, this work assumed that ground-truth annotations (provided by expert observers) are considered as gold standard and error-free. However, it is widely known (e.g., in medical image

analysis applications) that even expert observers show variation. Recently, several related works showed that variations exist among annotators in labelling plant images [46, 164]. Interestingly, intra- and inter-observer variation can be used to also assess algorithm performance. Based on the findings in Chapter 7, inter-observer variation has a mean square error of 0.81 (non-experienced annotators on a subset of Arabidopsis images used in [3]). Experienced and non-experienced annotators are within the ± 1 leaf error range in the $\sim 90\%$ of the cases, whereas *PhenoDC* is within ± 1 leaf error in $\sim 80\%$ of the cases, thus bringing us closer to human-level performance.

Evidently, "true" ground-truth data can only be attained by aggregating observations from many annotators to reach a consensus. Since doing so with experts is time-consuming, recent studies using online dedicated platforms, such as Zooniverse, can alleviate this problem, by tapping into the power of citizen scientists. An alternative is to use simulated or synthetic data, where ground-truth is absolute by design. Simulated data have recently been used in the plant community to count the number of fruits [80, 183]. Simulated images are provided by a software that takes object parameters as input (e.g., plant age, number of leaves). However, images may lack visual realism, but recent innovations in image synthesis [49] point to the potential of creating synthetic images of realistic appearance.

In conclusion, we present a deep learning approach to leaf counting with a neural network. Trained with examples of images and corresponding plant leaf counts, our approach can achieve outstanding results in a variety of settings. Our model handles many input modalities and has been tested with images of different species, cultivars and also with images at night. By making it openly available to the community we hope that it will stimulate large-scale analysis in plant phenotyping of a crucial plant trait: leaf count and help relieve the analysis bottleneck [28].

Part V

Conclusions and Future Work

10 Conclusions

10.1 Summary

This thesis discussed three important, and connected, research topics: (i) counting leaves with machine learning; (ii) data labelling of plant images; and (iii) applying deep learning to plant phenotyping.

In Part II, we showed a shallow machine learning approach for leaf counting, which we referred to as GLC. This method was published at the CVPPP 2015 workshop and was the best performing algorithm for the leaf counting challenge. Images are encoded using a holistic plant descriptor. The core of this algorithm is the learning of a visual dictionary, using image patches extracted from the log-polar representation. Due to the radial arrangement of leaves of a rosette plant, the log-polar representation allowed cancelling out the rotation and *pre-aligning* all the leaves in a common reference space. However, the log-polar representation resulted in a stretched central area of the plant, which is the most critical for leaf counting due to small emergent leaves. Therefore, we proposed a novel shallow network, based on the Restricted Boltzmann Machine [23], to learn rotation-invariant features. This allowed the extraction of patches from the original plant image, without using the log-polar

transformation. However, this method did not obtain better results than the GLC. Therefore, we argued that deep learning could bring a big benefit to the plant community, due to the ability of learning highly discriminative images features. However, due to the public availability of plant datasets and the sheer volume of annotated images, we need more training data for deep neural networks.

Part III showed three different strategies to collect more plant datasets. Firstly, we developed a tool to assist plant experts during the annotation process of images. This tool was later embedded in the *Phenotiki Analysis Software* [3]. Secondly, we made use of Zooniverse, an online platform for crowdsourcing, where volunteers could annotate plant datasets. We studied the observer variability between experts and non-experts, and related the findings with the variability exhibited by the citizens from Zooniverse. Lastly, we presented a deep generative adversarial network to synthesise the images of *Arabidopsis* plants with a specific number of leaves.

Part IV shows a deep neural network that can learn leaf counting from multiple sources and image modalities. Based on our previous approach [51] presented at the CVPPP 2017 workshop, we show that a multi-modal deep architecture can learn better features to perform a more accurate leaf counting.

10.2 Findings

In this section, we provide an extensive answer to the research questions presented in Chapter 1.

Q1: Can machine learning help in plant phenotyping?

In this thesis, we experimentally demonstrated that machine learning can be used in plant phenotyping. In Chapters 4 and 9, we showed two approaches, shallow and deep architectures respectively, to perform leaf count.

In a general supervised paradigm, machine learning algorithms

learn to relate images to a target variable. Specifically for this thesis, we presented in Chapter 4 an approach that extracted and learnt visual features from the log-polar representation of images in order to train a non-linear regressor [7]. In Chapter 9, we presented a deep learning architecture for leaf counting that train end-to-end the feature extractor and regressor simultaneously. We also showed that our deep learning architecture extracts high discriminative features and predicts a more accurate leaf count [48].

Q2: Can machine learning phenotype?

Once we assessed that machine learning could be used in plant phenotyping, the question of whether it is able to produce reliable data remains open. We show that our algorithm for leaf counting can phenotype in Chapter 6. The tools presented in the *Phenotiki Analysis Software* were validated, comparing algorithmic predictions against manual observation, showing any statistical difference.

Q3: Can we learn better and more compact features using rotation redundancies?

Top-view images of rosette-shaped plants exhibit radially arranged leaves. Therefore, the rotation is a nuisance factor that we want to cancel out. In Chapter 5, we showed two variants of a rotation-invariant RBM. Both methods are able to learn a robust and compact rotation-invariant features space. They differ in the way the dominant orientation is estimated.

In [24], our *Explicit Rotation-Invariant RBM* uses histograms of orientations to determine the dominant orientation of input images. This exogenous process has some limitations and cannot accommodate other datasets. For this reason, in [47], we presented a revised model that can use the reconstruction error to infer the suitable dominant orientation. This endogenous process showed better performance compared to the previous approach presented in [24].

However, replacing the K-means in Chapter 4 with the rotation-invariant RBM did not improve the leaf count results. This inspired us to focus on deep neural networks.

Q4: Can deep learning be employed in plant phenotyping?

The training of deep models with high generalisation capabilities needs large labelled datasets [196]. Table 6.1 lists the currently labelled datasets available for plant phenotyping and leaf counting. Overall, we can estimate ~ 1.5k images available for training. (Datasets without a testing set have to be split into training/validation/test sets.). Learning deep networks without overfitting with such small datasets is challenging. We showed in Chapters 8 and 9 that deep learning can also be used in this such constrained setup, using data agglomeration and regularised training.

Q5: How can more labelled data be collected?

We discussed three approaches to obtain more data for machine learning algorithms. In Chapter 6, we showed an annotation tool to facilitate experts during the annotation process. In Chapter 7, we showed how the Zooniverse online platform could be used to collect annotated data with crowdsourcing. Lastly, Chapter 8 showed how generative neural networks can synthesise annotated plant images.

Q6: Can multi-modal learning perform a better leaf count?

In [51], we showed that a deep neural network based on ResNet [14] can be trained to perform leaf count. In that paper we demonstrated that learning from multiple datasets simultaneously can improve the generalisation capabilities of the model. This encouraged us to explore the possibility of adding even more data sources. In Chapter 9, we used a multi-modal network to learn from multiple imageries, such as near-infrared and fluorescence. We showed that multi-modal learning improves the leaf count predictions.

10.3 Limitations

In this thesis, we presented two aspects related to the problem of learning to count leaves in rosette plants. One aspect focused on the development of reliable algorithms to count leaves, whereas the other aspect tackled the problem of how to obtain more training data for the machine learning algorithms. Although both aspects were analysed thoroughly and the deep learning approaches provided promising results, there are some limitations needing further investigation. These limitations are listed below:

Leaf counting aspect

- **Continuous predictions.** The leaf counting algorithms in Chapters 4 and 9 output a real number as number of leaves. In order to obtain an integer number, we round the inferred number to the nearest integer. However, a higher leaf count error within the ± 1 leaf range is due to this solution. An *ideal* direct regression model should be able to predict an integer number.
- **Unreliable ground-truth.** In Chapter 7, we showed that expert annotators show disagreement even over an easy task as counting the number of leaves in an image. Therefore, the ground-truth of plant datasets should not be considered completely reliable. However, our machine learning approaches could not learn from *noisy* labels.
- **Single inference.** Our algorithms in Chapters 4 and 9 were trained to perform one task. Since multiple information can be extracted from plants, an algorithm could be trained to infer multiple traits simultaneously within the same (deep) architecture.

Data annotation aspect

- **Segmentation task.** Related to the *single inference* limitation discussed above, Chapter 7 did not investigate the possibility of training citizen scientists to segment the leaves of the plant images. A further investigation assessing whether volunteers are able to provide finely grained and reliable per-leaf segmentations is necessary.

11 Future Work

In the previous chapter, we highlighted several important limitations of the work presented in this thesis. In this chapter, we will discuss ideas and suggestions aimed at these gaps.

11.1 Future Work in Leaf Counting

11.1.1 Learning from Noisy Labels

Another interesting aspect that needs further investigation is the learning of leaf count using noisy labels. As we showed in Chapter 7, experts and non-experts disagree when annotating a plant image for the leaf count. Therefore, it would be interesting learning a neural network with a certain ground-truth. We will assume that each image in the dataset has multiple (uncertain) annotations.

A naive approach is to obtain the ground-truth with a consensus over all the labels for the image, such as mean and median. This approach has already been explored in [201], showing that the deep network performed better using rounded up average. This results is due to the inclination of annotators to underestimate the leaf number. However, since an average is not a real ground truth, further

approaches on learning from noisy labels have to be investigated. Learning from noisy labels for the regression task has received little attention from the community. In [202], they showed a revised ridge regression approach able to learn from noisy labels. However, this approach has two main drawbacks. Firstly, it does not involve deep neural networks and hand-crafted or dictionary-based features need to be extracted (c.f., Chapter 4). Secondly, their method assumes that the same annotators provide labels for all the data in the training set, which was not the case with our Zooniverse project discussed in Chapter 7 (some annotators label only some images).

In the literature, there are several approaches to deep neural networks trained with noisy labels for classification. In [132], the authors alternate the update of the network parameters and the update of the labels. During training, the noisy annotations are replaced with the predictions of the network. The authors show that, after a sufficient amount of epochs, the network is able to recover the correct labels for most of the data in the training set. In [172], their architecture is able to model *flip noise* (images assigned to the wrong class, namely noisy labels), and *outlier noise* (images not related with any of the classes in the training set, but assigned to a class, namely an outlier). The authors tackle both of the noises, by learning a transition matrix and determining the probability that a given image can be assigned to the wrong class. Recently, a new approach to learning from noisy labels using distillation has been proposed [134]. This technique assumes two datasets: a *teacher* training set, and a *student* set. The former one is assumed to be clean (e.g. labels are not noisy) and it helps to learn from the *student* (noisy) dataset.

Overall, most of these approaches are applied for classification and further analysis and investigations for the regression case should be done.

11.1.2 Multi-Task Learning

A typical supervised deep neural network is divided into two parts: (i) feature extraction; and (ii) inference (c.f. Section 2.2). Taking *PhenoDC* as an example (c.f. Section 9.1), the purpose of the modality branch is to extract meaningful features for the task at hand. However, those features can also be used to accomplish other related tasks on plant image analysis, such as leaf segmentation.

Multi-task learning has recently been used in plant phenotyping. In [203], they proposed a deep neural network that simultaneously predicts heat map for counting spikes of the wheat and classifying if the plant is awned. Our approaches in [48, 51] can be trained to extract other important plant traits at the same time, such as age, species and/or genotype, PLA, etc.

11.1.3 Learning to Predict Integer Numbers

Our leaf counting algorithms (c.f. Chapters 4 and 9) predict a real number and the actual number of leaves is obtained by rounding to the nearest integer. However, this operation increases the error within the range of ± 1 leaf.

A future venture would be to investigate how to predict integer numbers directly. This could be achieved with a neural network that learns a discrete probability density function (e.g., Poisson).

11.2 Future Work in Data Collection

11.2.1 Cross-Modal Image Synthesis

The multi-modal imagery dataset in [15] contains a limited number of images in different modalities. However, other datasets (e.g. CVPPP 2017) only contain RGB images. As an example, cross-modal synthesis will allow the learning of a neural network that can transform RGB images into near-infrared. Modality synthesis is an interesting topic currently explored in other research fields, such as

medical imaging [197].

As a first approach, a Generative Adversarial Network [100] can be trained to perform the cross-modal synthesis. As a next step, *PhenoDC* can be extended, such that convolutional layers between two modality branches would perform the synthesis process in an end-to-end architecture.

11.2.2 Synthesis of Plant Images with a Per-Leaf Segmentation

The ARIGAN architecture (c.f., Chapter 8) provides image and total leaf count. Further investigations to extend it to also output per-leaf segmentation should be performed. Once images of plants with per-leaf segmentation are generated synthetically, they can be used to train deep segmentation networks with more data [81, 82].

Bibliography

[1] J. L. Micol, "Leaf development: time to turn over a new leaf?," *Current Opinion in Plant Biology*, vol. 12, no. 1, pp. 9–16, 2009.

[2] D. C. Boyes, A. M. Zayed, R. Ascenzi, A. J. McCaskill, N. E. Hoffman, K. R. Davis, and J. Görlach, "Growth stage-based phenotypic analysis of Arabidopsis: a model for high throughput functional genomics in plants.," *The Plant cell*, vol. 13, no. 7, pp. 1499–510, 2001.

[3] M. Minervini, M. V. Giuffrida, P. Perata, and S. A. Tsaftaris, "Phenotiki: An open software and hardware platform for affordable and easy image-based phenotyping of rosette-shaped plants.," *The Plant journal : for cell and molecular biology*, jan 2017.

[4] A. B. Chan, Z. S. J. Liang, and N. Vasconcelos, "Privacy preserving crowd monitoring: Counting people without people models or tracking," *CVPR*, pp. 1–7, 2008.

[5] K. Chen, C. C. Loy, S. Gong, and T. Xiang, "Feature Mining for Localised Crowd Counting," *Procedings of the British Machine Vision Conference 2012*, pp. 21.1–21.11, 2012.

[6] M. Minervini, A. Fischbach, H. Scharr, and S. A. Tsaftaris, "Finely-grained annotated datasets for image-based plant phenotyping," *Pattern Recognition Letters*, vol. 81, pp. 80–89, oct 2016.

[7] M. V. Giuffrida, M. Minervini, and S. Tsaftaris, "Learning to count leaves in rosette plants," in *Proceedings of the Computer Vision Problems in Plant Phenotyping (CVPPP)* (S. A. Tsaftaris, H. Scharr, and T. Pridmore, eds.), pp. 1.1–1.13, BMVA Press, September 2015.

[8] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation," *Proceedings of the 24th International Conference on Machine Learning (2007)*, no. 2006, pp. 473–480, 2007.

[9] T. Tieleman, "Training restricted boltzmann machines using approximations to the likelihood gradient," in *ICML*, (New York, NY, USA), pp. 1064–1071, ACM, 2008.

[10] N. Alajlan, M. Kamel, and G. Freeman, "Geometry-Based Image Retrieval in Binary Image Databases," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, pp. 1003–1013, jun 2008.

[11] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *CoRR*, vol. abs/1708.07747, 2017.

[12] H. Scharr, M. Minervini, A. Fischbach, and S. A. Tsaftaris, "Annotated Image Datasets of Rosette Plants," tech. rep., Forschungszentrum Jülich, 2014.

[13] M. Brown and D. G. Lowe, "Automatic Panoramic Image Stitching using Invariant Features," *International Journal of Computer Vision*, vol. 74, pp. 59–73, apr 2007.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Identity Mappings in Deep Residual Networks," in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), (Cham), pp. 630–645, Springer International Publishing, 2016.

[15] J. A. Cruz, X. Yin, X. Liu, S. M. Imran, D. D. Morris, D. M. Kramer, and J. Chen, "Multi-modality imagery database for plant phenotyping," *Machine Vision and Applications*, 2015.

[16] J. Bell and H. Dee, "Aberystwyth Leaf Evaluation Dataset," 2016.

[17] H. Uchiyama, S. Sakurai, M. Mishima, D. Arita, T. Okayasu, A. Shimada, and R.-i. Taniguchi, "An easy-to-setup 3d phenotyping platform for KOMATSUNA dataset," in *ICCVW*, Oct 2017.

[18] A. Dobrescu, L. C. T. Scorza, S. A. Tsaftaris, and A. J. McCormick, "A "Do-It-Yourself" phenotyping system: measuring growth and morphology throughout the diel cycle in rosette shaped plants," *Plant Methods*, vol. 13, p. 95, nov 2017.

[19] A. B. Chan and N. Vasconcelos, "Counting people with low-level features and bayesian regression," *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 2160–2177, 2012.

[20] J.-M. Pape and C. Klukas, "Utilizing machine learning approaches to improve the prediction of leaf counts and individual leaf segmentation of rosette plant images," in *Proceedings of the Computer Vision Problems in Plant Phenotyping (CVPPP)* (S. A. Tsaftaris, H. Scharr, and T. Pridmore, eds.), pp. 3.1–3.12, BMVA Press, September 2015.

[21] K. Sohn and H. Lee, "Learning Invariant Representations with Local Transformations," *Proceedings of the 29th ICML*, pp. 1311–1318, 2012.

[22] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[23] G. E. Hinton, "Training products of experts by minimizing contrastive divergence.," *Neural computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[24] M. V. Giuffrida and S. A. Tsaftaris, *Rotation-Invariant Restricted Boltzmann Machine Using Shared Gradient Filters*, pp. 480–488. Cham: Springer International Publishing, 2016.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[26] L. Grady, "Random walks for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1768–1783, 2006.

[27] M. Kern, "Food, feed, fibre, fuel and industrial products of the future: Challenges and opportunities. understanding the strategic potential of plant genetic engineering," *Journal of Agronomy and Crop Science*, vol. 188, no. 5, pp. 291–305, 2002.

[28] M. Minervini, H. Scharr, and S. A. Tsaftaris, "Image Analysis: the new bottleneck in plant phenotyping," *Signal Processing Magazine*, vol. 32, no. 4, pp. 126–131, 2015.

[29] R. T. Furbank and M. Tester, "Phenomics – technologies to relieve the phenotyping bottleneck," *Trends in Plant Science*, vol. 16, pp. 635–644, dec 2011.

[30] F. Fiorani and U. Schurr, "Future scenarios for plant phenotyping," *Annual review of plant biology*, 2013.

[31] J. Ubbens, M. Cieslak, P. Prusinkiewicz, and I. Stavness, "The use of plant models in deep learning: an application to leaf counting in rosette plants," *Plant Methods*, vol. 14, p. 6, jan 2018.

[32] S. A. Tsaftaris, M. Minervini, and H. Scharr, "Machine Learning for Plant Phenotyping Needs Image Processing," 2016.

[33] A. Singh, B. Ganapathysubramanian, A. K. Singh, and S. Sarkar, "Machine Learning for High-Throughput Stress Phenotyping in Plants," *Trends in Plant Science*, vol. 21, pp. 110–124, feb 2016.

[34] C. Granier, L. Aguirrezabal, K. Chenu, S. J. Cookson, M. Dauzat, and P. Hamard, "PHENOPSIS, an automated platform for reproducible phenotyping of plant responses to soil water deficit in Arabidopsis thaliana permitted the identification of an accession with low sensitivity to soil water deficit," *N Phytol*, vol. 169, 2006.

[35] M. Jansen, F. Gilmer, B. Biskup, K. A. Nagel, U. Rascher, and A. Fischbach, "Simultaneous phenotyping of leaf growth and chlorophyll fluorescence via Growscreen Fluoro allows detection of stress tolerance in Arabidopsis thaliana and other rosette plants," *Funct Plant Biol*, vol. 36, 2009.

[36] J. De Vylder, F. Vandenbussche, Y. Hu, W. Philips, and D. Van Der Straeten, "Rosette Tracker: An Open Source Image Analysis Tool for Automatic Quan-

tification of Genotype Effects," *Plant Physiology*, vol. 160, no. 3, pp. 1149–1159, 2012.

[37] S. Dhondt, N. Gonzalez, J. Blomme, L. De Milde, T. Van Daele, D. Van Akoleyen, V. Storme, F. Coppens, G. T.S. Beemster, and D. Inzé, "High-resolution time-resolved imaging of in vitro arabidopsis rosette growth," *The Plant Journal*, vol. 80, no. 1, pp. 172–184, 2014.

[38] S. Arvidsson, P. Pérez-Rodríguez, and B. Mueller-Roeber, "A growth phenotyping pipeline for Arabidopsis thaliana integrating image analysis and rosette area modeling for robust quantification of genotype effects," *New Phytologist*, vol. 191, pp. 895–907, 2011.

[39] M. Awlia, A. Nigro, J. Fajkus, S. M. Schmoeckel, S. Negrão, and D. Santelia, "High-throughput non-destructive phenotyping of traits that contribute to salinity tolerance in Arabidopsis thaliana," *Front Plant Sci*, vol. 7, 2016.

[40] A. Hartmann, T. Czauderna, R. Hoffmann, N. Stein, and F. Schreiber, "HTPheno: an image analysis pipeline for high-throughput plant phenotyping," *BMC Bioinformatics*, vol. 12, no. 1, p. 148, 2011.

[41] A. Telfer, K. M. Bollman, and R. S. Poethig, "Phase change and the regulation of trichome distribution in Arabidopsis thaliana," *Development*, vol. 124, pp. 645–654, 1997.

[42] D. Bradley, O. Ratcliffe, C. Vincent, R. Carpenter, and E. Coen, "Inflorescence commitment and architecture in Arabidopsis," *Science*, vol. 275, no. 5296, pp. 80–83, 1997.

[43] A. Walter and U. Schurr, "The modular character of growth in Nicotiana tabacum plants under steady-state nutrition," *Journal of Experimental Botany*, vol. 50, no. 336, pp. 1169–1177, 1999.

[44] M. Koornneef, C. Hanhart, P. van Loenen-Martinet, and H. Blankestijn de Vries, "The effect of daylength on the transition to flowering in phytochrome-deficient, late-flowering and double mutants of Arabidopsis thaliana," *Physiologia Plantarum*, vol. 95, no. 2, pp. 260–266, 1995.

[45] M. Kouressy, M. Dingkuhn, M. Vaksmann, A. Clément-Vidal, and J. Chantereau, "Potential contribution of dwarf and leaf longevity traits to yield improvement in photoperiod sensitive sorghum," *European Journal of Agronomy*, vol. 28, no. 3, pp. 195–209, 2008.

[46] M. V. Giuffrida, F. Chen, H. Scharr, and S. A. Tsaftaris, "Citizen crowds and experts: Observer variability in image-based plant phenotyping," *The Plant Methods*, 2018.

[47] M. V. Giuffrida and S. A. Tsaftaris, "Explicit Factorization of Rotations in Restricted Boltzmann Machines," *IEEE Transactions on Image Processing (submitted)*, 2018.

[48] M. V. Giuffrida, P. Doerner, and S. A. Tsaftaris, "Pheno-Deep Counter: a unified and versatile deep learning architecture for leaf counting," *The Plant Journal (under review)*, 2018.

[49] M. Valerio Giuffrida, H. Scharr, and S. A. Tsaftaris, "Arigan: Synthetic arabidopsis plants using generative adversarial network," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[50] M. Minervini, M. V. Giuffrida, and S. Tsaftaris, "An interactive tool for semi-automated leaf annotation," in *Proceedings of the Computer Vision Problems in Plant Phenotyping (CVPPP)* (S. A. Tsaftaris, H. Scharr, and T. Pridmore, eds.), pp. 6.1–6.13, BMVA Press, September 2015.

[51] A. Dobrescu, M. Valerio Giuffrida, and S. A. Tsaftaris, "Leveraging multiple datasets for deep leaf counting," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[52] M. Rahnemoonfar and C. Sheppard, "Deep Count: Fruit Counting Based on Deep Simulated Learning," *Sensors*, vol. 17, p. 905, apr 2017.

[53] T. Kozuka, G. Horiguchi, G.-T. Kim, M. Ohgishi, T. Sakai, and H. Tsukaya, "The Different Growth Responses of the Arabidopsis thaliana Leaf Blade and the Petiole during Shade Avoidance are Regulated by Photoreceptors and Sugar," *Plant and Cell Physiology*, vol. 46, pp. 213–223, jan 2005.

[54] M. C. Cox, F. F. Millenaar, Y. E. d. J. van Berkel, A. J. Peeters, and L. A. Voesenek, "Plant movement. submergence-induced petiole elongation inrumex palustris depends on hyponastic growth," *Plant Physiology*, vol. 132, no. 1, pp. 282–291, 2003.

[55] T. Tuytelaars and K. Mikolajczyk, "Local invariant feature detectors: A survey," *Found. Trends. Comput. Graph. Vis.*, vol. 3, pp. 177–280, July 2008.

[56] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," *Workshop on statistical learning in computer vision, ECCV*, vol. 1, no. 1-22, pp. 1–2, 2004.

[57] H. Lee, A. Battle, R. Raina, and A. Y. Ng, "Efficient Sparse coding algorithms," *Advances in neural information processing systems*, pp. 801–808, 2006.

[58] A. Agarwal and B. Triggs, *Computer Vision – ECCV: 9th European Conference on Computer Vision*, ch. Hyperfeatures – Multilevel Local Coding for Visual Recognition, pp. 30–43. Springer Berlin Heidelberg, 2006.

[59] I. Arel, D. C. Rose, and T. P. Karnowski, "Deep Machine Learning - A New Frontier in Artificial Intelligence Research," *IEEE Computational Intelligence Magazine*, vol. 5, no. 4, pp. 13–18, 2010.

[60] X. Wei, S. L. Phung, and A. Bouzerdoum, "Visual descriptors for scene categorization: experimental evaluation," *Artificial Intelligence Review*, vol. 45, no. 3, pp. 1–36, 2015.

[61] F. Rosenblatt, "The Perceptron - A Perceiving and Recognizing Automaton," tech. rep., Cornell Aeronautical Laboratory, 1957.

[62] D. R. Cox, "The Regression Analysis of Binary Sequences," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 20, no. 2, pp. 215–242, 1958.

[63] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, p. 533, oct 1986.

[64] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[65] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, pp. 211–252, dec 2015.

[66] J. P. Barrett, "The Coefficient of Determination—Some Limitations," *The American Statistician*, vol. 28, no. 1, pp. 19–20, 1974.

[67] H. Scharr, M. Minervini, A. P. French, C. Klukas, D. M. Kramer, X. Liu, I. Luengo, J.-M. Pape, G. Polder, D. Vukadinovic, X. Yin, and S. A. Tsaftaris, "Leaf segmentation in plant phenotyping: a collation study," *Machine Vision and Applications*, vol. 27, pp. 585–606, may 2016.

[68] D. Kong, D. Gray, and H. Tao, "A viewpoint invariant approach for crowd counting," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 3, pp. 1187–1190, 2006.

[69] B. Tan, J. Zhang, and L. Wang, "Semi-supervised elastic net for pedestrian counting," *Pattern Recognition*, vol. 44, no. 10–11, pp. 2297 – 2304, 2011. Semi-Supervised Learning for Visual Content Analysis and Understanding.

[70] C. Wang, H. Zhang, L. Yang, S. Liu, and X. Cao, "Deep People Counting in Extremely Dense Crowds," in *Proceedings of the 23rd ACM International Conference on Multimedia*, MM '15, (New York, NY, USA), pp. 1299–1302, ACM, 2015.

[71] J. Chung and K. Sohn, "Image-Based Learning to Measure Traffic Density Using a Deep Convolutional Neural Network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, pp. 1670–1675, may 2018.

[72] S. Aich and I. Stavness, "Improving Object Counting with Heatmap Regulation," 2018.

[73] C. Arteta, V. Lempitsky, J. A. Noble, and A. Zisserman, "Interactive object counting," in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), (Cham), pp. 504–518, Springer International Publishing, 2014.

[74] V. Lempitsky and A. Zisserman, "Learning To Count Objects in Images," in *Advances in Neural Information Processing Systems 23* (J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, eds.), pp. 1324–1332, Curran Associates, Inc., 2010.

[75] L. Fiaschi, R. Nair, U. Koethe, and F. A. Hamprecht, "Learning to Count with Regression Forest and Structured Labels," in *21st International Conference on Pattern Recognition (ICPR 2012)*, pp. 2685–2688, 2012.

[76] M. S. Norouzzadeh, A. Nguyen, M. Kosmala, A. Swanson, M. S. Palmer, C. Packer, and J. Clune, "Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning," *Proceedings of the National Academy of Sciences*, p. 201719367, 2018.

[77] C. Arteta, V. Lempitsky, and A. Zisserman, "Counting in the wild," in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), (Cham), pp. 483–498, Springer International Publishing, 2016.

[78] Y. Ovadia, Y. Halpern, D. Krishnan, J. Livni, D. Newburger, R. Poplin, T. Zha, and D. Sculley, "Learning to count mosquitoes for the sterile insect technique," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1943–1949, ACM, 2017.

[79] S. Aich and I. Stavness, "Leaf counting with deep convolutional and deconvolutional networks," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[80] J. R. Ubbens and I. Stavness, "Deep Plant Phenomics: A Deep Learning Platform for Complex Plant Phenotyping Tasks," *Frontiers in Plant Science*, vol. 8, jul 2017.

[81] B. Romera-Paredes and P. H. S. Torr, "Recurrent Instance Segmentation," in *Computer Vision – ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), pp. 312–329, Cham: Springer International Publishing, 2016.

[82] M. Ren and R. S. Zemel, "End-to-end instance segmentation with recurrent attention," in *CVPR*, 2017.

[83] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 2, p. 252 Vol. 2, 1999.

[84] J. Canny, "A computational approach to edge detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-8, pp. 679–698, Nov 1986.

[85] A. Marana, L. da Fontoura Costa, R. Lotufo, and S. Velastin, "Estimating crowd density with minkowski fractal dimension," in *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, vol. 6, pp. 3521–3524 vol.6, Mar 1999.

[86] C. Xu and Y. Chen, "Classifying image texture with statistical landscape features," *Pattern Analysis and Applications*, vol. 8, no. 4, pp. 321–331, 2006.

[87] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, vol. 1 of *Springer Series in Statistics*. New York, NY: Springer New York, 2009.

[88] R. Cossu, "Segmentation by means of textural analysis," *Pixel*, vol. 1, pp. 21–24, 1988.

[89] J. Selinummi, J. Seppälä, O. Yli-Harja, and J. Puhakka, "Software for quantification of labeled bacteria from digital microscope images by automated image analysis," *Biotechniques*, pp. 39(6):859–63, 2005.

[90] D. Lowe, "Object recognition from local scale-invariant features," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 2, pp. 1150–1157 vol.2, 1999.

[91] L. Breiman, "Random forests," *Machine learning*, pp. 5–32, 2001.

[92] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[93] J. P. Cohen, H. Z. Lo, and Y. Bengio, "Count-ception: Counting by fully convolutional redundant counting," *CoRR*, vol. abs/1703.08710, 2017.

[94] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[95] C. Zhang, H. Li, X. Wang, and X. Yang, "Cross-scene crowd counting via deep convolutional neural networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[96] C. Shang, H. Ai, and B. Bai, "End-to-end crowd counting via joint learning local and global count," in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 1215–1219, Sept 2016.

[97] V. A. Sindagi and V. M. Patel, "Generating high-quality crowd density maps using contextual pyramid cnns," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[98] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *ICLR*, vol. abs/1409.1556, 2015.

[99] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma, "Single-image crowd counting via multi-column convolutional neural network," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[100] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 2672–2680, Curran Associates, Inc., 2014.

[101] J. V. B. Soares and D. W. Jacobs, "Efficient segmentation of leaves in semi-controlled conditions," *Machine Vision and Applications*, vol. 24, pp. 1623–1643, nov 2013.

[102] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *SIGKDD Explorations*, vol. 11, no. 1, pp. 10–18, 2009.

[103] A. Coates, A. Arbor, and A. Y. Ng, "An analysis of single-layer networks in unsupervised feature learning," *International Conference on Artificial Intelligence and Statistics*, pp. 215–223, 2011.

[104] E. E. Aksoy, A. Abramov, F. Wörgötter, H. Scharr, A. Fischbach, and B. Dellen, "Modeling leaf growth of rosette plants using infrared stereo image sequences," *Computers and Electronics in Agriculture*, vol. 110, no. JANUARY, pp. 78–90, 2015.

[105] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, vol. 11. Elsevier B.V., forth ed., 2008.

[106] Y. Jia, C. Huang, and T. Darrell, "Beyond spatial pyramids: Receptive field learning for pooled image features," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3370–3377, 2012.

[107] Y. Boureau, J. Ponce, and Y. Lecun, "A theoretical analysis of feature pooling in visual recognition," in *International Conference on Machine Learning*, pp. 111–118, 2010.

[108] H. Drucker, C. J. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, "Support vector regression machines," in *Advances in neural information processing systems*, pp. 155–161, 1997.

[109] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, 2004.

[110] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. New York, NY, USA: Cambridge University Press, 2004.

[111] M. Minervini, M. M. Abdelsamea, and S. Tsaftaris, "Image-based plant phenotyping with incremental learning and active contours," *Ecological Informatics*, vol. 23, pp. 35–48, 2014.

[112] M. Minervini, C. Rusu, and S. A. Tsaftaris, "Computationally efficient data and application driven color transforms for the compression and enhancement of images and video," in *Color Image and Video Enhancement*, ch. 13, Springer, 2015.

[113] R. Ostrovsky, Y. Rabani, L. J. Schulman, and C. Swamy, "The effectiveness of Lloyd-type methods for the k-means problem," *Journal of the ACM*, vol. 59, no. 6, pp. 28:1–28:22, 2013.

[114] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[115] M. A. Carreira-Perpinan and G. E. Hinton, "On contrastive divergence learning," *Proceedings of the tenth international workshop on artificial intelligence and statistics*, pp. 33–40, 2005.

[116] G. Casella and E. I. George, "Explaining the gibbs sampler," *The American Statistician*, vol. 46, no. 3, pp. pp. 167–174, 1992.

[117] J. J. Kivinen and C. K. I. Williams, "Transformation Equivariant Boltzmann Machines," in *ICANN*, vol. 6791, pp. 1–9, Springer, 2011.

[118] U. Schmidt and S. Roth, "Learning rotation-aware features: From invariant priors to equivariant descriptors," *Proceedings of the IEEE CVPR*, pp. 2050–2057, 2012.

[119] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, (New York, NY, USA), pp. 609–616, ACM, 2009.

[120] S. Roth and M. J. Black, "Fields of experts," *International Journal of Computer Vision*, vol. 82, no. 2, pp. 205–229, 2009.

[121] Z. Shou, Y. Zhang, and H. J. Cai, "A study of transformation-invariances of deep belief networks," in *IJCNN*, pp. 1–8, IEEE, 2013.

[122] D. Cheng, T. Sun, X. Jiang, and S. Wang, "Unsupervised feature learning using Markov deep belief network," in *2013 IEEE International Conference on Image Processing*, pp. 260–264, IEEE, 2013.

[123] H. Lee, C. Ekanadham, and A. Y. Ng, "Sparse deep belief net model for visual area V2," *Advances in Neural Information Processing Systems 20 (NIPS 2007)*, pp. 873–880, 2008.

[124] G. Hinton, *A Practical Guide to Training Restricted Boltzmann Machines*. Springer Berlin Heidelberg, 2nd ed., 2012.

[125] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 886–893, 2005.

[126] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, sep 1995.

[127] B. V. Dasarathy, *Nearest Neighbor ({NN}) Norms: {NN} Pattern Classification Techniques*. Los Alamitos, CA: IEEE Computer Society Press, 1991.

[128] M. V. Giuffrida and S. A. Tsaftaris, "Theta-RBM: Unfactored Gated Restricted Boltzmann Machine for Rotation-Invariant Representations," tech. rep., arXiv, jun 2016.

[129] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *AISTATS* (Y. W. Teh and M. Titterington, eds.), vol. 9 of *Proceedings of Machine Learning Research*, pp. 249–256, PMLR, 2010.

[130] A. Rasmus, T. Raiko, and H. Valpola, "Denoising autoencoder with modulated lateral connections learns invariant representations of natural images," *arXiv:1412.7210*, 2014.

[131] I. Goodfellow, H. Lee, Q. V. Le, A. Saxe, and A. Y. Ng, "Measuring invariances in deep networks," in *NIPS 22*, pp. 646–654, 2009.

[132] D. Tanaka, D. Ikami, T. Yamasaki, and K. Aizawa, "Joint optimization framework for learning with noisy labels," *CoRR*, vol. abs/1803.11364, 2018.

[133] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *ICML*, pp. 833–840, Omnipress, 2011.

[134] Y. Li, J. Yang, Y. Song, L. Cao, J. Luo, and L. J. Li, "Learning from Noisy Labels with Distillation," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-Octob, pp. 1928–1936, 2017.

[135] K. H. Cho, T. Raiko, and A. Ilin, "Gaussian-Bernoulli deep Boltzmann machine," *Proceedings of the International Joint Conference on Neural Networks*, pp. 1–9, 2013.

[136] K. McGuinness and N. E. O'Connor, "A comparative evaluation of interactive segmentation algorithms," *Pattern Recognition*, vol. 43, no. 2, 2010.

[137] W. A. Barrett and E. N. Mortensen, "Interactive live-wire boundary extraction," *Medical Image Analysis*, vol. 1, no. 4, pp. 331–341, 1997.

[138] D. Cremers, M. Rousson, and R. Deriche, "A review of statistical approaches to level set segmentation: Integrating color, texture, motion and shape," *International Journal of Computer Vision*, vol. 72, no. 2, pp. 195–215, 2007.

[139] C. Couprie, L. Grady, L. Najman, and H. Talbot, "Power watershed: A unifying graph-based optimization framework," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 7, pp. 1384–1399, 2011.

[140] J. De Vylder, F. J. Vandenbussche, Y. Hu, W. Philips, and D. Van Der Straeten, "Rosette Tracker: an open source image analysis tool for automatic quantification of genotype effects," *Plant Physiology*, vol. 160, no. 3, pp. 1149–1159, 2012.

[141] H. M. Easlon and A. J. Bloom, "Easy leaf area: Automated digital image analysis for rapid and accurate measurement of leaf area," *Applications in Plant Sciences*, vol. 2, no. 7, 2014.

[142] C. Weight, D. Parnham, and R. Waites, "LeafAnalyser: a computational method for rapid and large-scale analyses of leaf shape variation," *The Plant Journal*, vol. 53, no. 3, pp. 578–586, 2008.

[143] L. Remmler and A.-G. Rolland-Lagan, "Computational method for quantifying growth patterns at the adaxial leaf surface in three dimensions," *Plant Physiology*, vol. 159, no. 1, pp. 27–39, 2012.

[144] K. Pearson, "The problem of the random walk," *Nature*, vol. 72, no. 1865, p. 294, 1905.

[145] C. Gardiner, *Stochastic Methods: A Handbook for the Natural and Social Sciences*. Springer Series in Synergetics, Springer, 2009.

[146] R. N. Bhattacharya and E. C. Waymire, *Stochastic Processes with Applications*. Classics in Applied Mathematics, Society for Industrial and Applied Mathematics, 2009.

[147] L. Lovász, "Random walks on graphs: A survey," 1993.

[148] R. Courant and D. Hilbert, *Methods of Mathematical Physics*, vol. 1 of *Methods of Mathematical Physics*. Wiley, 2008.

[149] B. Mohar, "The Laplacian spectrum of graphs," in *Graph Theory, Combinatorics, and Applications*, pp. 871–898, Wiley, 1991.

[150] S. Goff, M. Vaughn, S. McKay, E. Lyons, A. Stapleton, D. Gessler, N. Matasci, L. Wang, M. Hanlon, A. Lenards, A. Muir, N. Merchant, S. Lowry, S. Mock, M. Helmke, A. Kubach, M. Narro, N. Hopkins, D. Micklos, U. Hilgert, M. Gonzales, C. Jordan, E. Skidmore, R. Dooley, J. Cazes, R. McLay, Z. Lu, S. Pasternak, L. Koesterke, W. Piel, R. Grene, C. Noutsos, K. Gendler, X. Feng, C. Tang, M. Lent, S.-j. Kim, K. Kvilekval, B. Manjunath, V. Tannen, A. Stamatakis, M. Sanderson, S. Welch, K. Cranston, P. Soltis, D. Soltis, B. O'Meara, C. Ane, T. Brutnell, D. Kleibenstein, J. White, J. Leebens-Mack, M. Donoghue, E. Spalding, T. Vision, C. Myers, D. Lowenthal, B. Enquist, B. Boyle, A. Akoglu, G. Andrews, S. Ram, D. Ware, L. Stein, and D. Stanzione,

"The iplant collaborative: Cyberinfrastructure for plant biology," *Frontiers in Plant Science*, vol. 2, p. 34, 2011.

[151] J. J. Kieber, M. Rothenberg, G. Roman, K. A. Feldmann, and J. R. Ecker, "CTR1, a negative regulator of the ethylene response pathway in arabidopsis, encodes a member of the Raf family of protein kinases," *Cell*, vol. 72, no. 3, pp. 427–441, 1993.

[152] P. Guzmán and J. R. Ecker, "Exploiting the Triple Response of Arabidopsis to Identify Ethylene-Related Mutants," *The Plant Cell*, vol. 2, no. 6, pp. 513–523, 1990.

[153] T. Caspar, S. C. Huber, and C. Somerville, "Alterations in Growth, Photosynthesis, and Respiration in a Starchless Mutant of Arabidopsis thaliana (L.) Deficient in Chloroplast Phosphoglucomutase Activity," *Plant Physiology*, vol. 79, no. 1, pp. 11–17, 1985.

[154] P. Perata and A. Alpi, "Plant responses to anaerobiosis," *Plant Science*, vol. 93, no. 1, pp. 1–17, 1993.

[155] A. Wiese, M. M. Christ, O. Virnich, U. Schurr, and A. Walter, "Spatio-temporal leaf growth patterns of arabidopsis thaliana and evidence for sugar control of the diel leaf growth cycle," *New Phytologist*, vol. 174, no. 4, pp. 752–761, 2007.

[156] H. Poorter, N. P. R. Anten, and L. F. M. Marcelis, "Physiological mechanisms in plant growth models: Do we need a supra-cellular systems biology approach?," *Plant, Cell and Environment*, vol. 36, no. 9, pp. 1673–1690, 2013.

[157] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, "Speeded up detection of squared fiducial markers," *Image and Vision Computing*, vol. 76, pp. 38 – 47, 2018.

[158] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[159] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans Syst Man Cybern*, vol. 9, 1979.

[160] J. H. Kellgren and J. S. Lawrence, "Radiological assessment of osteo-arthrosis," *Annals of the Rheumatic Diseases*, vol. 16, no. 4, pp. 494–502, 1957.

[161] K. Hartung and H.-P. Piepho, "Are ordinal rating scales better than percent ratings? a statistical and "psychological" view," *Euphytica*, vol. 155, pp. 15–26, May 2007.

[162] J. A. Poland and R. J. Nelson, "In the eye of the beholder: the effect of rater variability and different rating scales on qtl mapping," *Phytopathology*, vol. 101, no. 2, pp. 290–298, 2011.

[163] N. MacLeod, M. Benfield, and P. Culverhouse, "Time to automate identification," *Nature*, vol. 467, no. 7312, p. 154, 2010.

[164] S. Ghosal, D. Blystone, A. K. Singh, B. Ganapathysubramanian, A. Singh, and S. Sarkar, "An explainable deep machine vision framework for plant

stress phenotyping.," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 115, pp. 4613–4618, may 2018.

[165] F. W. Nutter, Jr., "Assessing the Accuracy, Intra-rater Repeatability, and Inter-rater Reliability of Disease Assessment Systems," *Phytopathology*, vol. 83, no. 8, p. 806, 1993.

[166] C. H. Bock, P. E. Parker, A. Z. Cook, and T. R. Gottwald, "Visual Rating and the Use of Image Analysis for Assessing Different Symptoms of Citrus Canker on Grapefruit Leaves," *Plant Disease*, vol. 92, pp. 530–541, apr 2008.

[167] C. H. Bock, G. H. Poole, P. E. Parker, and T. R. Gottwald, "Plant Disease Severity Estimated Visually, by Digital Photography and Image Analysis, and by Hyperspectral Imaging," *Critical Reviews in Plant Sciences*, vol. 29, pp. 59–107, mar 2010.

[168] A. M. Mutka and R. S. Bart, "Image-based phenotyping of plant disease symptoms," *Frontiers in Plant Science*, vol. 5, p. 734, 2014.

[169] D. N. Laskin and G. J. McDermid, "Evaluating the level of agreement between human and time-lapse camera observations of understory plant phenology at multiple scales," *Ecological Informatics*, vol. 33, pp. 1 – 9, 2016.

[170] D. N. Laskin and G. J. McDermid, "Evaluating the level of agreement between human and time-lapse camera observations of understory plant phenology at multiple scales," *Ecological Informatics*, vol. 33, pp. 1 – 9, 2016.

[171] H. S. Naik, J. Zhang, A. Lofquist, T. Assefa, S. Sarkar, D. Ackerman, A. Singh, A. K. Singh, and B. Ganapathysubramanian, "A real-time phenotyping framework using machine learning for plant stress severity rating in soybean," *Plant Methods*, vol. 13, p. 23, apr 2017.

[172] S. Sukhbaatar and R. Fergus, "Learning from Noisy Labels with Deep Neural Networks," *arXiv preprint*, pp. 1–11, 2014.

[173] S. E. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, "Training deep neural networks on noisy labels with bootstrapping," *arXiv*, p. 1412.6596, 2014.

[174] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: A database and web-based tool for image annotation," *Int. J. Comput. Vision*, vol. 77, pp. 157–173, May 2008.

[175] J. Howe, "The rise of crowdsourcing." `https://www.wired.com/2006/06/crowds/`. Accessed: 2017-04-01.

[176] T. Caspar, S. C. Huber, and C. Somerville, "Alterations in growth, photosynthesis, and respiration in a starchless mutant of arabidopsis thaliana (l.) deficient in chloroplast phosphoglucomutase activity," *Plant Physiology*, vol. 79, pp. 11–17, 09 1985.

[177] R. V. Lenth, "Java applets for power and sample size [computer software]." `http://www.stat.uiowa.edu/~rlenth/Power`. Accessed: 2017-04-01.

[178] K. L. Gwet, *Handbook of Inter-Rater Reliability: The Definitive Guide to Measuring the Extent of Agreement Among Raters*. Gaithersburg, MD: Advanced Analytics, LLC, 2014.

[179] J. M. Girard, "MATLAB functions for computing inter-observer reliability." `https://github.com/jmgirard/mReliability`. Accessed: 2017-05-20.

[180] J. Martin Bland and D. Altman, "Statistical methods for assessing agreement between two methods of clinical measurement," *The Lancet*, vol. 327, pp. 307–310, 2017/06/06 1986.

[181] A. Sheshadri and M. Lease, "SQUARE: A benchmark for research on computing crowd consensus.," in *First AAAI Conference on Human Computation and Crowdsourcing – HCOMP* (B. Hartman and E. Horvitz, eds.), AAAI, 2013.

[182] T. R. Jones, A. E. Carpenter, M. R. Lamprecht, J. Moffat, S. J. Silver, J. K. Grenier, A. B. Castoreno, U. S. Eggert, D. E. Root, P. Golland, and D. M. Sabatini, "Scoring diverse cellular morphologies in image-based screens with iterative feedback and machine learning," *Proceedings of the National Academy of Sciences*, vol. 106, no. 6, pp. 1826–1831, 2009.

[183] L. Mündermann, Y. Erasmus, B. Lane, E. Coen, and P. Prusinkiewicz, "Quantitative modeling of arabidopsis development," *Plant Physiology*, vol. 139, no. 2, pp. 960–968, 2005.

[184] Y. LeCun, "MNIST Dataset."

[185] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," in *arXiv*, 2014.

[186] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," in *ICCV*, 2017.

[187] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *CoRR*, vol. abs/1511.06434, 2015.

[188] E. Denton, S. Chintala, A. Szlam, and R. Fergus, "Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks," in *Advances in Neural Information Processing Systems 28* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 1486–1494, Curran Associates, Inc., 2015.

[189] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.), vol. 37 of *Proceedings of Machine Learning Research*, (Lille, France), pp. 448–456, PMLR, 07–09 Jul 2015.

[190] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[191] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, and Y. Bengio, "Theano: new features and speed improvements," *CoRR*, vol. abs/1211.5590, 2012.

[192] D. Kinga and J. B. Adam, "A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, vol. 5, 2015.

[193] F. Apelt, D. Breuer, Z. Nikoloski, M. Stitt, and F. Kragler, "Phytotyping 4D : A light-field imaging system for non-invasive and accurate monitoring of spatio-temporal plant growth," *The Plant Journal*, vol. 82, no. 4, pp. 693–706, 2015.

[194] M. A. Gehan, N. Fahlgren, A. Abbasi, J. C. Berry, S. T. Callen, L. Chavez, A. N. Doust, M. J. Feldman, K. B. Gilbert, J. G. Hodge, J. S. Hoyer, A. Lin, S. Liu, C. Lizárraga, A. Lorence, M. Miller, E. Platon, M. Tessman, and T. Sax, "PlantCV v2: Image analysis software for high-throughput plant phenotyping," *PeerJ*, vol. 5, p. e4088, 2017.

[195] C. Klukas, D. Chen, and J.-M. Pape, "IAP: an open-source information system for high-throughput plant phenotyping.," *Plant physiology*, vol. 165, no. June, pp. 506–518, 2014.

[196] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[197] A. Chartsias, T. Joyce, M. Giuffrida, and S. Tsaftaris, "Multimodal MR Synthesis via Modality-Invariant Latent Representation," *IEEE Transactions on Medical Imaging*, vol. 37, no. 3, 2018.

[198] F. Chollet *et al.*, "Keras." `https://github.com/fchollet/keras`, 2015.

[199] Y. Bengio, "Deep Learning of Representations for Unsupervised and Transfer Learning," in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning* (I. Guyon, G. Dror, V. Lemaire, G. Taylor, and D. Silver, eds.), vol. 27 of *Proceedings of Machine Learning Research*, (Bellevue, Washington, USA), pp. 17–36, PMLR, 2012.

[200] S. He, *Active learning for image processing*. PhD thesis, The University of Edinburgh, 2016.

[201] F. Chen, *Citizen Scientists and Machine Learning to Help Feed the World*. PhD thesis, The University of Edinburgh, 2017.

[202] K. Ristovski, D. Das, V. Ouzienko, Y. Guo, and Z. Obradovic, "Regression learning with multiple noisy oracles," *Frontiers in Artificial Intelligence and Applications*, vol. 215, no. June 2015, pp. 445–450, 2010.

[203] M. P. Pound, J. A. Atkinson, D. M. Wells, T. P. Pridmore, and A. P. French, "Deep learning for multi-task plant phenotyping," in *Computer Vision Workshop (ICCVW), 2017 IEEE International Conference on*, pp. 2055–2063, IEEE, 2017.

A

Neural Network Activation Functions

In order to understand the role of activation functions, we should abstract some definitions of neural networks. In Section 2.3, we showed that the input data is processed through layers of mathematical operations (typically multiplications of weights). If Figure 2.4 is an artificial neural network, each node it contains is a neuron. To quantify whether some part of the data is 'stimulating' a neuron, we need to have an activation functions indicating whether the information is triggering the neuron or not. A list of typical activation functions is provided in Table A.1.

Specifically, *linear activation* functions are useful for e.g. regression tasks. *Logistic* and *tanh* functions are sigmoid functions used for several purposes, such as segmentation, image synthesis, etc. *Rectified Linear Unit* is the most used non-linear activation functions used in deep learning, usually following a convolutional layer. In case $\alpha = 0$, one will get classical ReLU, whereas $\alpha > 0$ (usually, 0.01), one will have a *Leaky ReLU* activation function, useful to facilitate the backpropagation of very deep networks.

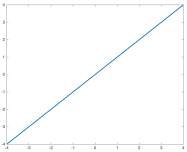
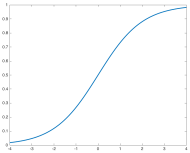
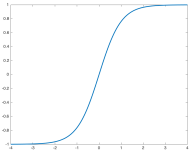
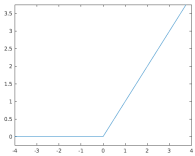
Expression	Range	Plot
Linear Activation		
$\phi(x) = x$	$(-\infty, +\infty)$	
Logistic Function		
$\sigma(x) = \frac{1}{1+e^{-x}}$	$(0, 1)$	
Hyperbolic tangent		
$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	$(-1, 1)$	
Rectified Linear Unit (ReLU)		
$\mathrm{ReLU}(x) = \max(\alpha x, x)$	$(-\infty, +\infty)$	

Table A.1: Most frequent activation functions used on artificial neural networks.

"So Long, and Thanks for All the Fish."

DOUGLAS ADAMS