

University of Nottingham
UK | CHINA | MALAYSIA

Learning with Fewer Labels in Deep Learning for Plant Phenotyping

Thesis submitted to the University of Nottingham for the degree of **Doctor of Philosophy, July 2022.**

Feng Chen

Student ID: 14320514

Supervised by

Prof. Andrew P. French
Prof. Tony Pridmore

Signature Feng Chen

Date 29 / 07 / 2022

Abstract

Deep learning has become an important approach in the research of plant phenotyping. However, training a highly-performing deep learning model generally requires a large amount of labeled data. These training data are normally labeled by human annotators and the labeling process is expensive and time-consuming. In plant phenotyping, precisely labeling the images often needs professional knowledge, for example, knowing the species of a plant or the appearance of an organ, disease or pest, which further increases the costs of annotating the data. Hence, exploring effective training methods that use less labeled data to achieve comparable performance can significantly boost the development of deep learning in plant phenotyping.

In this thesis, three categories of approaches are explored, namely active learning, semi-supervised learning, and weakly-supervised learning, with a common purpose – reducing the dependence on labeled data. Novel approaches of learning with fewer labels are proposed and evaluated on various plant-phenotyping-related and generic tasks. In addition to boosting the performance of deep learning with fewer labels, the proposed methods are adaptive or require fewer parameters, so that the effort of optimizing the learning models is also reduced. These approaches are also evaluated on other computer vision tasks to show their generality.

In particular, we perform flower classification to demonstrate that ac-

tive learning can reduce 30% training data without sacrificing much performance. We perform semi-supervised aphid counting and crowd counting using Mean Teacher and FixMatch, which are two milestones in semi-supervised learning. Specifically, we improve the property and reduce the hyper-parameters of Mean Teacher by using an adaptive consistency weight. FixMatch is improved and adapted to heatmap regression using Monte-Carlo Dropout and a novel data augmentation method – Shadowout. Finally, a scenario combining semi and weakly-supervised learning – learning with incomplete pixel-level labels is explored on wheat spikelet localisation and crowd counting. We proposed to use a simple yet effective method – an asymmetric loss function to learn well-performed models in this scene.

Acknowledgements

First and foremost, I would like to express the most sincere gratitude to my major supervisor, Prof. Andrew French, who supported me all the time during my PhD. I was a bit lost when I started my PhD, but Andy always trusted me and encouraged me to explore new or even weird ideas. It was also him to provide me with the "fine-grained" supervision and guide me to carefully examine every idea. Besides academic advice, I also learned to balance my work and life as he often suggested me to take a break. Next, I would like to thank my second supervisor, Prof. Tony Pridmore, for providing precious advice and opportunities that were beneficial to my research and career.

In addition, I am grateful to the University of Nottingham for providing me with the PhD scholarship.

I would like to express the special gratitude to my master supervisor in the University of Edinburgh, Prof. Sotirios Tsaftaris, for leading me into the domains of machine learning and plant phenotyping, and connecting me with Andy and Tony.

I would also like to thank Dr. Michael Pound for selflessly sharing inspiring research ideas and technical solutions with me. My gratitude extends to the examiners of my annual reports, Prof. Michel Valstar and Dr. Mercedes Torres Torres, who patiently reviewed my progress and advised me on potential improvement.

Many thanks to all the members at the Computer Vision Lab for being supportive, inspiring and fun. Thanks to all my friends in Nottingham for bringing me joys and sorrows.

Last but not least, I would like to express the deepest gratitude to my parents for always encouraging me to find my true self and supporting me in pursuit of the life I want.

Contents

Abstract i

Acknowledgements iii

List of Tables xi

List of Figures xii

Chapter 1 Introduction 1

1.1 Aims and Objectives 2

1.2 Organization of the Thesis 2

1.3 Main Contributions 5

1.4 Publications 6

Chapter 2 Background 7

2.1 Convolutional Neural Networks 8

2.1.1 Overview 8

2.1.2 Basic Components of the CNNs 8

2.1.3 Training and Evaluating the CNNs 10

2.2 Plant Phenotyping in Computer Vision 12

2.3 Motivation of Exploring Learning with Fewer Labels 13

2.4 Active Learning 14

2.4.1 Categorization of Active Learning 15

2.4.2 Querying Strategy 15

2.5 Semi-supervised Learning 18

2.5.1 Introduction 18

2.5.2 Inductive Methods 18

2.5.3 Transductive Methods 25

2.6 Weakly-supervised Learning 26
2.6.1 Introduction 26
2.6.2 Coarse-grained Supervision 27
2.6.3 Inaccurate Supervision 29
2.7 Summary 31
Chapter 3 Active Learning on Flower Classification 33
3.1 Introduction 34
3.2 Active Learning 34
3.2.1 Pool-based Active Learning 34
3.2.2 Acquisition Functions 36
3.3 Active Learning on Flower Classification 38
3.3.1 VGG Flower Dataset 39
3.3.2 Implementation Details 40
3.3.3 Results 41
3.4 Discussion 44
3.5 Conclusion 45
Chapter 4 Semi-supervised Counting with Adaptive Mean Teacher 46
4.1 Introduction 47
4.2 Related Work 49
4.2.1 Crowd Counting 49
4.2.2 Mean Teacher 54
4.2.3 Limitations of the Ramp-up Function 58
4.3 Method 58
4.3.1 Adaptive Consistency Weight 58
4.4 Datasets 61
4.4.1 Aphid Dataset 61
4.4.2 ShanghaiTech Crowd Dataset 65
4.5 Experiments 65

4.5.1 CSRNet in Mean Teacher 66
4.5.2 Aphid Counting 68
4.5.3 Crowd Counting 73
4.6 Discussion 81
4.7 Conclusion 82
Chapter 5 Semi-supervised Counting with Monte-Carlo-Dropout FixMatch and Shadowout 85
5.1 Introduction 86
5.2 Related Work 87
5.2.1 FixMatch 87
5.2.2 Limitations of FixMatch 91
5.3 Method 92
5.3.1 Shadowout 92
5.3.2 FixMatch with Monte Carlo Dropout 94
5.4 Experiments 95
5.4.1 Implementation Details 96
5.4.2 Crowd Counting 100
5.4.3 Aphid Counting 105
5.5 Discussion 110
5.6 Conclusion 111
Chapter 6 Learning to Localise and Count With Incomplete Point-Annotations 113
6.1 Introduction 114
6.2 Related Work 117
6.2.1 Learning with Noisy Labels 117
6.2.2 Learning with Incomplete Labels 118
6.2.3 Limitations of Previous Work on Noisy or Incomplete Labels 119
6.2.4 Context-Aware Network 119

6.2.5 Stacked Hourglass Network . . . 122
6.3 Method . . . 124
6.3.1 Problem Definition . . . 124
6.3.2 AMSE vs. MSE . . . 126
6.4 Experiments . . . 128
6.4.1 Datasets . . . 128
6.4.2 Implementation Details . . . 131
6.4.3 Crowd Counting with AMSE . . . 132
6.4.4 Wheat Spikelet Localisation with AMSE . . . 143
6.5 Discussion . . . 149
6.5.1 Summaries of AMSE . . . 149
6.5.2 Limitations of Experiment Settings . . . 150
6.6 Conclusions . . . 151
Chapter 7 Conclusions and Future Work 153
7.1 Summaries of the Work . . . 154
7.2 Limitations of the Work . . . 156
7.3 Future Work . . . 158
7.3.1 Open-set Active Learning . . . 158
7.3.2 Improving AMSE with Uncertainty Estimation . . . 159
7.3.3 Learning the Concept of Repetition . . . 160
7.3.4 Coarse-grained Supervision . . . 162
7.4 Conclusions . . . 164
Bibliography 166

List of Tables

4.1 Average counts of different categories per image in the original (before cropped) aphid dataset 62

4.2 Statistics of the aphid dataset used in the following experiments. Num is the number of images, Max, Min and Avg count represent the maximum, minimum and average number of aphids in the images respectively. 63

4.3 Statistics of the ShanghaiTech crowd dataset. Num is the number of images, Max, Min and Avg count represent the maximum, minimum and average number of crowds in the images respectively. 65

4.4 Different training settings between label-only and semi-supervised aphid counting experiments. 68

4.5 P-values of the two-tailed paired student's t-test for aphid counting, verifying the results shown in Figure 4.10. The p-values that are less than the 0.05 significance level is bolded. 73

4.6 Training settings for the label-only and semi-supervised crowd counting experiments. It is the same setup as for aphid counting, except batch size is 16 in the label-only experiments because it generates the best performance compared to the other candidate values of batch size 74

4.7 P-values of the two-tailed two-sample unequal variance student's t-test for crowd counting, verifying the results shown in Figure 4.13 80

5.1 List of image transformations from RandAugment used in our implementation. Rotating, shearing and translating are removed because they cause displacement. Every time, $N = 2$ transformations are randomly selected from the list, and each transformation has a 50% chance to be applied to the image with the intensity (if applicable) randomly drew from the associated parameter range. 98

5.2 P-values of the two-tailed unpaired unequal variance student's t-test for crowd counting, verifying the results shown in Figure 5.4. Here we use FM to represent MC-FixMatch + Shadowout. 102

5.3 Training time on crowd counting of Mean Teacher (ramp=5), adaptive Mean Teacher, and MC-FixMatch + Shadowout. The training time is calculated by $n \times t$, where n is the index of the iteration that the model converges and t is the time (seconds) needed to train the model per iteration. The results are averaged from 5 runs and normalized by the training time of Mean Teacher. 102

5.4 Student's t-test results for Figure 5.6, verifying the significance of the performance change when applying different τ to MC-FixMatch on crowd counting. 104

5.5 P-values of the two-tailed paired Student's t-test results for Figure 5.7, comparing FM (i.e. MC-FixMatch + Shadowout) with the other training schemes on semi-supervised aphid counting. 107

5.6 Training time on aphid counting of Mean Teacher (ramp=5), adaptive Mean Teacher, and MC-FixMatch + Shadowout. The training time is calculated by $n \times t$, where n is the index of the iteration that the model converges and t is the time (seconds) needed to train the model per iteration. The results are averaged from 5 runs and normalized by the training time of Mean Teacher. 107

5.7 Student's t-test results for Figure 5.8, comparing the performance gap between different occlusion-based data augmentation on semi-supervised aphid counting. 109

6.1 Stats of the ShanghaiTech crowd dataset and ACID wheat dataset. The average count represents the average number of people or spikelets of the images in the corresponding dataset. To note, in the wheat spikelet localisation experiments, the images from ACID are randomly cropped to 256x256 resolution, so the average count reduces accordingly. 129

6.2 Training parameters of wheat spikelet localisation (ACID-SHN) and crowd counting (ShA/ShB-CAN). In the optimizer part, lr denotes learning rate, wd denotes weight decay, and mo denotes momentum. 131

6.3 Validation MAE and RMSE (lower is better) for $dr = 0$ under different AMSE parameters β on ShanghaiTech dataset using Context-Aware Network: (a) validation results on ShanghaiTech Part A; (b) validation results on ShanghaiTech Part B. $\beta = 0$ is equivalent to using MSE loss function and $\beta \neq 0$ stands for using AMSE loss function in training. The best result (i.e. minimum) in each column is highlighted in bold. 133

6.4 Validation MAE/RMSE (lower is better) on ShA-CAN under different drop rates. Columns under $\beta = 0$: results obtained by training with MSE loss function. Columns under $\beta \neq 0$: best results obtained by training with AMSE loss function, with β value in bracket showing when the corresponding result is achieved. 135

6.5 Validation MAE/RMSE (lower is better) on ShB-CAN under different drop rates. Columns under $\beta = 0$: results obtained by training with MSE loss function. Columns under $\beta \neq 0$: best results obtained by training with AMSE loss function, with β value in bracket showing when the corresponding result is achieved. 136

6.6 Validation F1 score on spikelet-SHN under different drop rates. $\beta = 0$: results obtained by training with MSE loss function. $\beta \neq 0$: best results obtained by training with AMSE loss function, with β value in bracket showing when the corresponding result is achieved. 144

List of Figures

2.1 Pool-based (black arrows) and stream-based (blue arrows) active learning. 16
2.2 Π-model (top) and Temporal Ensembling (bottom). Image source: [74]. 23
2.3 Structure (top) and a visual example (bottom) of the class activation map. Image source: [170]. 28

3.1 Active learning pipeline: the black arrows and the red arrows respectively denote the paths of the training samples and pooling samples 36
3.2 Examples of VGG 17-category flower dataset. 39
3.3 Test accuracy of VGG-16 (top) and VGG-19 (bottom) with different acquisition functions. Each graph reports the mean and standard deviation of the rank-1 (left) and rank-5 (right) accuracy from 10 runs. The "complete training set" indicates training the models on 952 images without active learning, and the remainders are trained on 663 (68 initial + 595 pooled) images using active learning. 42
3.4 The average number (over 10 runs) of pooled images accumulated per class during training the VGG-16-based model, using different acquisition functions. 43
3.5 Visual examples of 6 most queried flower categories, from left to right, 1st to 6th. 44

4.1 The example shows a 3×3 convolution kernel (green) with different dilation rate r. With the same number of parameters, dilated convolution ($r > 1$) captures larger region from the input than normal convolution ($r = 1$). 53

4.2 The architecture of CSRNet, which consists of a pre-trained VGG-16 feature extractor frontend, a dilated convolution backend, and a 1×1 convolution output head. The normal convolution layers (yellow block) are shown as "conv-(kernel size)-(number of kernels)" and the dilated convolution layers (green blocks) are denoted as "dconv-(kernel size)-(number of kernels)-(dilation rate)". Each convolution and dilated convolution layer is followed by a ReLU activation except the final 1×1 convolution output layer. 53

4.3 Structure of the Mean Teacher model. The weights of the teacher model (θ') is an exponential moving average (EMA) of the student's weights (θ). The models are applied with noise (η and η'), respectively. The aim is to force the student model to predict consistently as the teacher model on the same input. Image source: [140]. 55

4.4 An example of using the sigmoid ramp-up function on the consistency weight. The consistency weight is increased from zero to its final value $\lambda = 1$ during the first 50 training epochs, and then remains fixed during the rest of training. . 57

4.5 Examples of different classes of the original aphid dataset. We crop the images to 1024×768 and train the model to count the aphids with (blue box) + without (red box) wings together in this chapter (details are described below). 63

4.6 Examples of the cropped aphid dataset. Original images are shown at the top and the corresponding true heatmaps are presented at the bottom. Each aphid count (shown at the corner of the heatmap) is the sum of all the pixels of the corresponding heatmap. 64
4.7 Examples of ShanghaiTech Part B crowd dataset. Original images are shown at the top and the corresponding true heatmaps are presented at the bottom, with the count of people (i.e. sum of all the pixels of the heatmap) shown at the bottom right. 66
4.8 Using CSRNet in Mean Teacher for heatmap regression. The sigmoid ramp-up function and the proposed adaptive consistency weight dynamically changes the weight of the consistency loss λ. 67
4.9 Validation MAE and RMSE (lower is better) of aphid counting with different number of labeled training images. The green bars indicate the percentage of the labeled data to the entire dataset. Each pair of the validation MAE (bottom blue line) and RMSE (top orange line) is shown with mean (i.e. the value next to the associated point) and standard deviation (i.e. the error bar on the associated point) over 5 runs. 71
4.10 Validation MAE (left) and RMSE (right) of aphid counting on 10% labeled training data using different training schemes. Each pair of MAE and RMSE is reported in mean (i.e. the value associated with the bar) and standard deviation (i.e. the error bar) from training on 5 different splits of 10% labeled data. 72

4.11 Validation MAE and RMSE (lower is better) of crowd counting with different number of labeled training images. The green bars indicate the percentage of the labeled data to the entire dataset. Each pair of the validation MAE (bottom blue line) and RMSE (top orange line) is shown with mean (i.e. the value below/above the associated point) and standard deviation (i.e. the error bar on the associated point) over 5 runs. 76
4.12 Validation MAE and RMSE of training the CSRNet with or without batch normalization on 100% and 10% labeled samples, denoted in mean (i.e. values in the bars) and standard deviation (i.e. error bars). The results are averaged from 5 runs of the model trained on the same split of the training data. Batch normalization (if in use) is applied before ReLU at each block of the backend. 76
4.13 Validation MAE (left) and RMSE (right) of crowd counting on 10% labeled training data using different training schemes. Each pair of MAE and RMSE is reported in mean (i.e. the value associated with the bar) and standard deviation (i.e. the error bar) from 5 runs of training on the same 10% labeled data. 78

5.1 FixMatch algorithm on image classification. For any unlabeled sample, a weakly-augmented version is predicted by the model (top). Only when the greatest value among the assigned probabilities exceeds the threshold (dotted line), the prediction is transferred to a one-hot pseudo-label. With a cross-entropy loss ($H(p, q)$), this pseudo-label is used to supervise the model trained on the strongly-augmented version of the same example (bottom). Image source: [132]. 89

5.2 Different occlusion-based image transformation methods applied on four aphid images. The original images are shown at the top row, and the effects of applying different transformations to these images are shown at the remaining rows, respectively. 93

5.3 Visual examples of every transformation in RandAugment used in our experiments (details described in 5.1). The original input image is shown in the middle. 99

5.4 Validation MAE (left) and RMSE (right) of crowd counting on 10% labeled training data using different training schemes. Each pair of MAE and RMSE is reported in mean (i.e. the value associated with the bar) and standard deviation (i.e. the error bar) from 5 runs of training on the same 10% labeled data (i.e. same data as in Chapter 4). 101

5.5 The number of selected pseudo-labels per training iteration on 10% labeled ShB, using different uncertainty thresholds τ. 103

5.6 Validation MAE and RMSE of MC-FixMatch (+ Shadowout) varying the uncertainty threshold τ, trained on 10% labeled data of ShB. The label-only bar is for the convenience of comparison. The results are reported as averaging 5 runs on the same split of data. 104

5.7 Validation MAE (left) and RMSE (right) of aphid counting on 10% labeled training data using different training schemes. Each pair of MAE and RMSE is reported in mean (i.e. the value associated with the bar) and standard deviation (i.e. the error bar) from training on 5 different splits of 10% labeled data (i.e. identical splits as in the aphid counting experiments of Chapter 4). 106

5.8 Validation MAE (left) and RMSE (right) of using different occlusion methods with MC-FixMatch. Random crop, random flip and RandAugment are still applied before the evaluated occlusion-based transformation. Each pair of MAE and RMSE is reported in mean (i.e. the value associated with the bar) and standard deviation (i.e. the error bar) from training on 5 different splits of 10% labeled aphid data (i.e. identical splits as Chapter 4). 108

6.1 Architecture of the Context-Aware Network. **Top**: general structure of Context-Aware Network. AMSE is used to supervise the network so it can learn from partially annotated data. As can be seen, the contribution of this chapter is the AMSE and the handling of partially-annotated data on the right hand side - the rest of the architecture remains the same as [92] **Bottom**: details of the contextual block. 121

6.2 Architecture of a four-block Stacked Hourglass Network. **Top**: general structure of the Stacked Hourglass Network, based on [111]; each white block in the figure represents a residual block (**bottom**). The method proposed in this thesis uses AMSE (indicated on the right of the figure) to supervise the model so it can learn from partially annotated data. 123

6.3 A graphical overview of reduced annotations: upper row: ShanghaiTech Part B [167], bottom row: ACID wheat dataset [111]. (a) input image, (b) original instance heatmap image, (c) and (d) show the same heatmap as (b) but with 50% and 90% of annotations dropped respectively. The aim is to train a well-performing network using dropped annotations as represented in (c) and (d). 125

6.4 A comparison between MSE and AMSE. In both graphs, the blue solid line represents MSE, and the dashed lines show the behaviour of AMSE under different parameters (noted as β). The graph on the left shows AMSE under $\beta > 0$, where the left part of each curve is suppressed, while the graph on the right shows the opposite. 127

6.5 Example images and the corresponding ground truth from the datasets used in this paper. Left: ACID dataset (image source: [111]). Right: ShanghaiTech dataset (image source: [167]). 130

6.6 Validation MAE (lower is better) on ShA-CAN under different drop rates. Blue line: results obtained by using MSE loss function (i.e. $\beta = 0$) in training. Orange line: best results obtained by using AMSE loss function in training, with β value under each point showing when the corresponding result is achieved. 134

6.7 Validation RMSE (lower is better) on ShA-CAN under different drop rates. Blue line: results obtained by using MSE loss function (i.e. $\beta = 0$) in training. Orange line: best results obtained by using AMSE loss function in training, with β value under each point showing when the corresponding result is achieved. 134

6.8 Validation MAE (lower is better) on ShB-CAN under different drop rates. Blue line: results obtained by using MSE loss function (i.e. $\beta = 0$) in training. Orange line: best results obtained by using AMSE loss function in training, with β value under each point showing when the corresponding result is achieved. 137

6.9 Validation RMSE (lower is better) on ShB-CAN under different drop rates. Blue line: results obtained by using MSE loss function (i.e. $\beta = 0$) in training. Orange line: best results obtained by using AMSE loss function in training, with β value under each point showing when the corresponding result is achieved. 137

6.10 β value where the best validation MAE is achieved under each drop rate on ShA/ShB-CAN. 138

6.11 Average validation MAE/bias of ShA/ShB-CAN on $dr = 0.5$ from repeated experiments. Under each parameter set, the curves show the mean and standard deviation (error bars) of 5 repeated experiments, with various random seeds to generate different dropped training data. All the curves are smoothed after every 10 epochs. (a)/(b) Average validation MAE/bias of ShA-CAN on $dr = 0.5$ with MSE and AMSE of $\beta = 0.5$ (i.e. optimal *beta*, below as the same); (c)/(d) Average validation MAE/bias of ShB-CAN on $dr = 0.5$ with MSE and AMSE of $\beta = 0.8$. 140

6.12 Validation MAE of ShA/ShB-CAN with various β on $dr = 0.3$ and $dr = 0.9$ through training: (a)/(b) validation MAE of ShA-CAN trained on $dr = 0.3/dr = 0.9$, (c)/(d) validation MAE of ShB-CAN trained on $dr = 0.3/dr = 0.9$. The results are averaged every 10th epoch. In each sub-graph, the blue dashed line (trained on fully-annotated data with MSE loss function) and the orange dashed line (trained on reduced data with MSE loss function) delineate the baseline boundaries of the experiments. The solid lines (trained on reduced data with AMSE loss function) are expected to fall between the dashed lines to show the effectiveness of AMSE; those closer to the blue dashed line show better performance. 142

6.13 Validation F1 score (higher is better) on spikelet localisation under different drop rates. Blue line: results obtained by using MSE loss function (i.e. $\beta = 0$) in training. Orange line: best results obtained by using AMSE loss function in training, with β value above each point showing when the corresponding result is achieved. Green bar: what percentage of annotations per image is used in training (100% stands for fully-annotated data). 145

6.14 (a) – (e): validation F1 scores of spikelet-SHN under $dr = 0.1, 0.3, 0.5, 0.7, 0.9$ respectively through training. Each curve is smoothed by averaging every 5 results. In each sub figure, the blue dashed line (trained on fully-annotated data with MSE loss function) and the orange dashed line (trained on dropped data with MSE loss function) delineate the baseline boundaries of the experiments. The solid lines are trained on dropped data with AMSE of different values of β. (f): The best value of β for each drop rate. 146

6.15 Inputs, ground truth and predicted heatmaps of spikelet localisation under different dr. The first row in each sub figure shows the validation images. The second row in each sub figure shows the ground truth (fully annotated). The third row in each sub figure shows the predictions by the model trained using MSE loss on the dropped data under the given dr. The fourth row in each sub figure shows the predictions by the model trained using AMSE loss with the optimal β under the given dr. 148

7.1 Example images of the data with mixed snacks (top) and synthetic shapes (bottom). The synthetic shapes are generated in random color and size to force the learning model focus on the contours of the shapes. 161

7.2 Example images of pollen cells (the bright blobs). 163

7.3 The pipeline of weakly-supervised pollen cell counting. . . . 164

Chapter 1

Introduction

1.1 Aims and Objectives

Improving the quality and yield of crops is one of the biggest challenges in the 21st century due to a rapidly growing population and climate change. Plant phenotyping, which explores the structure and function (i.e. the *phenome*) of plants, is key to optimize the genome and growing environment of plants and to fulfill the challenge. Computer vision and deep learning have developed significantly in last decade, and have become important tools in the research of plant phenotyping. Deep learning is particularly powerful if a tremendous number of correctly-labeled data (e.g. images) is available. However, labeling images costs a lot of time and money, especially in some professional areas such as plant phenotyping, where expert knowledge of certain species, organs, pests or diseases is required, which further exacerbates the difficulty of annotating the images.

In this thesis, we aim to reduce the efforts of labeling data by developing approaches of training deep learning models with fewer annotations. These methods are related to active learning (which incrementally require labels from oracles), semi-supervised learning (which utilize a small number of labeled data and a large amount of unlabeled data), and weakly-supervised learning (which only requires easy to produce or incomplete labels). In addition, we aim to develop fewer-hyper-parameter or adaptive solutions to reduce the difficulty of optimizing the learning models.

1.2 Organization of the Thesis

This section summarizes the work and the structure of the thesis.

Chapter 2 **reviews the background** of convolutional neural net-

works, vision-based plant phenotyping, and approaches of learning with fewer labels, particularly in the fields of active learning, semi-supervised learning, and weakly-supervised learning. Active learning reduces the number of labeled data by actively selecting the most informative samples and inquiring their labels from an oracle (e.g. expert human annotator). Compared to active learning, semi-supervised learning does not require extra human annotation power; instead, it directly explores the unlabeled data to improve the learning model. Rather than reducing the number of labels, weakly-supervised learning attempts to learn a task with easy-to-produce or noisy labels.

Chapter 3 **evaluates multiple acquisition functions**, which are key to active learning, in flower classification. We consider flower classification as a fundamental plant phenotyping task, where the visual traits of various species are important. The successful application of active learning demonstrates the possibility of training a model with fewer but more informative samples, which is noteworthy to the plant community. This section leads to a publication shown in Section 1.4 (3).

Chapter 4 evaluates a consistency-regularization-based semi-supervised learning approach, **Mean Teacher**, on heatmap-regression-based crowd counting and aphid counting. Mean Teacher forms the consistency between the predictions of the student model and the teacher model on a pair of slightly disturbed inputs. This consistency is implemented by optimizing a weighted loss function, where the weight is often set to monotonically increase assuming the teacher model becomes more and more reliable during training. However, we find this assumption is generally true but the reliability of teacher model fluctuates. Hence, an adaptive consistency weight update method is proposed, which improves the performance and reduces the hyper-parameters of Mean Teacher.

Chapter 5 continues to examine semi-supervised heatmap-regression-based crowd counting and aphid counting on a more recent and more generalizable method, **FixMatch**. FixMatch combines consistency regularization with pseudo-labeling, proposing to use stronger data transformation and to pseudo-label the unlabeled data with high-confidence predictions. However, as FixMatch is originally proposed for *classification* tasks, its strong data augmentation and way of selecting the confident predictions are not appropriate for heatmap regression tasks. Hence, we propose a new data transformation method, Shadowout, and to use Monte-Carlo Dropout to pseudo-label the unlabeled data, which adapts and improves FixMatch on heatmap regression tasks. This section leads to a manuscript being in development, shown in Section 1.4 (1).

Chapter 6 explores a novel scenario – learning with **per-image incomplete annotations**, on heatmap-regression-based crowd counting and wheat spikelet localisation. In this scenario, pixel-level annotations (e.g. point-annotations) are reduced (e.g. by 50%) in every training image. It leads to a novel task in between semi-supervised learning and weakly-supervised learning: on one hand, it requires fewer labels in total of a dataset; on the other hand, it reduces the labeling complexity of every image. This scenario attempts to boost a faster and less expensive annotating paradigm in plant phenotyping, when many similar objects per image need to be labeled by experts. An asymmetric loss function with only one hyper-parameter is proposed to train the learning model in this task. This section leads to a publication shown in Section 1.4 (2).

Chapter 7 first summarizes the thesis, and then introduces some future work to further reduce the dependence on labeled data, and even towards unsupervised learning, where some fundamental ideas have already been explored during my PhD.

1.3 Main Contributions

This section summarizes the major contributions of the thesis. The details of the contributions will be further explained and discussed in the corresponding chapters.

In Chapter 3, we **apply active learning with different acquisition functions to flower classification**. The experiments show that active learning is able to reduce the labeling efforts without sacrificing much performance in plant phenotyping.

In Chapter 4, we **propose a fully-annotated aphid dataset**. In addition, we **propose an adaptive consistency weight update method** to Mean Teacher, which improves the property of semi-supervised learning aphid counting and reduces the hyper-parameter-tuning efforts.

In Chapter 5, **we propose MC-FixMatch, which integrates Monte-Carlo Dropout with FixMatch**, to adapt the originally-classification-based FixMatch to heatmap regression. In addition, we **propose a counting-specific data augmentation, Shadowout**. The proposed approaches demonstrate better generalization abilities than Mean Teacher, as they achieve satisfying performance of semi-supervised learning not only in aphid counting, but also in the more generic crowd counting.

In Chapter 6, we **propose to use the asymmetric mean square error (AMSE)** to address the scenario of per-image incomplete point-annotations in heatmap regression. The experiments on wheat spikelet localisation and crowd counting show that the proposed method is able to counteract the reduction of point-annotations, both intentionally or unintentionally.

1.4 Publications

Papers published or drafted during my PhD are listed in reverse chronological order as below:

1. **Feng Chen**, Samuel Asamoah, Rumiana Ray, Andrew P. French, *Adapting FixMatch to Semi-supervised Aphid Counting*, The British Machine Vision Conference (BMVC), 2022 (manuscript in preparation)

2. **Feng Chen**, Michael Pound, Andrew P. French, *Learning to Localise and Count with Incomplete Dot-annotations*, Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW), 2021

3. **Feng Chen**, Andrew P. French, Tony Pridmore, *Using Active Learning to Reduce Training Data in Deep-learning-based Flower Classification*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshop (CVPRW), 2019

4. John Atanbori, **Feng Chen**, Andrew P. French, Tony Pridmore, *Towards Low-cost Image-based Plant Phenotyping Using Reduced-parameter CNN*, The British Machine Vision Conference Workshop (BMVCW), 2018

5. Mario V. Giuffrida, **Feng Chen**, Hanno Scharr, and Sotirios A. Tsaftaris, *Citizen Crowds and Experts: Observer Variability in Image-based Plant Phenotyping*, Plant Methods, 2018.

Chapter 2

Background

2.1 Convolutional Neural Networks

2.1.1 Overview

Modern CNNs have been developed rapidly since the AlexNet [71] defeated every other competitor in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [30, 118, 117, 31] in 2012. In the last decade, CNNs achieved great performance in a lot of computer vision tasks such as image classification [151], object detection [91] and semantic classification [76]. CNNs also demonstrated their effectiveness in many areas including medical image analysis [90], autonomous driving [48] and agriculture [60]. With this rapid development, the architectures of CNNs also "evolved" – a number of generic models were created. For example, VGGNet [129] and GoogLeNet (also known as Inception V1) [138] were the first two "deep" CNNs; ResNet [52] introduced the residual block to mitigate the problem of vanishing gradients and make even deeper networks possible. These generic models are still playing an important role in today's research and applications.

Below we introduce the basic components of CNNs, and how to train and evaluate the performance of the CNNs.

2.1.2 Basic Components of the CNNs

A typical CNN is composed of convolutional layers, non-linear activation functions, pooling layers, and sometimes fully-connected layers, which are introduced below.

Convolutional layer extracts features by sliding multiple trainable kernels (i.e. convolutional kernels) across the input (e.g. RGB images or fea-

ture maps from last layer). This process is also known as convolution. The convolutional kernel is usually fixed-size (e.g. 3×3-size) and much smaller than the input. Therefore, the convolutional kernel can not only capture local spatial relationships, which is very important to computer vision, but also reduces the computation complexity – the number of pixels in images or videos are often very large.

Non-linear activation functions are usually applied after the convolutional layer to add non-linearity to the learning models. This non-linearity enables the models to approach arbitrary functions, especially complex non-linear functions, which are a common representation of things in real life. For example, Rectified Linear Unit (ReLU) is a simple yet popular activation function, which is denoted as:

$$f(x) = \max(0, x), \tag{2.1}$$

where x represents the pixel values of the feature maps extracted from the last layer.

Pooling layers are usually applied after the combination of the convolutional layer and the non-linear activation function, to "summarize" the extracted features. Similar to the convolutional layer, the pooling layer slides a fixed-size but non-trainable pooling kernel across the input. For example, the max/average pooling kernel outputs the maximum/average value within the area of its kernel size of the input. The pooling layer also reduces the overall number of parameters of the CNNs by shrinking the previous feature maps.

Fully-connected layers are usually used in the last few layers of CNNs for final predictions (e.g. classification scores). Different from the convolu-

tional layer which slides the convolutional kernels of small size across the input, a fully-connected layer independently multiplies every pixel of the input with different trainable weights. On the one hand, a fully-connected layer can map the previously extracted features to the desired output; on the other hand, it consumes massive computational power. To note, in some scenarios such as semantic segmentation and heatmap regression, where the prediction is image-like, a fully-connected layer is not necessary.

2.1.3 Training and Evaluating the CNNs

CNNs are built to mathematically solve different tasks, usually by predicting the expected outputs from the given inputs. The establishment of the CNNs require three unique sub-datasets – the CNNs initially "learn" (i.e. optimize the trainable parameters) from the *training dataset* and then evaluated on the *validation dataset* and the *test dataset*. Below the difference between these three datasets and some other important concepts of training and evaluating the CNNs are introduced.

2.1.3.1 Training Stage

In supervised learning, each data pair from the training dataset consists of the input (e.g. an image) and the corresponding labels (e.g. classification labels) which are often provided by the human annotators. The CNNs are trained to fit on these training pairs by minimizing one or multiple *losses* between the model predictions and the true labels. These loss functions are often task-specific, for example, cross entropy for classification and mean squared error for regression. Besides, there are various options of *optimization* approaches to update the trainable parameters of the CNNs towards

the minimal values such as SGD, RMSprop and Adam [65]. To note, this thesis mainly discusses different approaches to reduce the dependence on the training labels.

The concepts of *batch size*, *iteration* and *epoch* are also introduced here, which may appear in the remainder of the thesis. As the number of the training samples are often quite large in practice, the training datasets are split to batches and the models are trained to fit each batch one after another. Batch size refers to the number of samples in each batch; iteration refers to the number of batches that a model has been fit on; and epoch refers to how many times a model has been fit on the entire training set.

In addition, some techniques are applied to the CNNs during training for better performance. For example, dropout [134] randomly drops part of the features (e.g. 50%) from some layers at each training iteration to reduce overfitting; batch normalization [56] normalizes the features of some layers at each batch to accelerate training. In most cases, dropout and batch normalization should be turned off during evaluation.

2.1.3.2 Evaluation Stage

The trained CNNs are evaluated on the validation set and the test set. The inputs from both datasets are labeled, and the trained models are expected to predict the same as the labels from the associated inputs. In most cases, the data from the training, validation and test sets are different.

The validation set is often used after every epoch or every N epoch of training to evaluate the current model on the unseen data. The validation set can provide information of when the best model is achieved, or when to stop training, for example, at the point of the validation loss plateaus

or increases while the training loss remains decreasing (i.e. overfitting). Besides, some hyper-parameters of training a model like batch size and learning rate of the optimizer are allowed to change based on the validation results.

The test set should only be used once when the final model is ready (i.e. any component or parameter of the model is optimized and fixed), to imitate the scenarios of deploying the learning model in real life.

In summary, the data in both validation and test sets should never be used to optimize the trainable parameters of the learning model, while the validation set is allowed to refine some hyper-parameters (e.g. batch size and learning rate) during training. However, in some cases, only the validation set is used to evaluate the trained model without the test set. As a result, the optimal model selected according to its validation performance represents the theoretical best model to be deployed in real life.

2.2 Plant Phenotyping in Computer Vision

Plant phenotyping explores the observable traits of plants (e.g. shape and color) which are highly related to the genotypes and growing conditions (e.g. light, water and soil) of the plants; it plays an important role in modern agriculture such as breeding and weeding. The arise of CNNs and deep learning has drawn attention from the plant community in recent years because of its efficiency [58, 8].

Computer-vision-based plant phenotyping includes a wide range of tasks, for example, species classification and recognition (e.g. flower classification [103, 105]), plant organ counting (e.g. leaf counting [44, 34, 43]

and fruit counting [2, 10]), plant segmentation [128, 156], etc. The data used in these tasks are generally divided by two types according to their collected conditions – controlled environment and field data [98]. The data gathered from *controlled environment* (e.g. lab and greenhouse) is usually less noisy with consistent background; the *field* data, however, is often noisier, varying in illumination, camera angles, background, etc. Different gathering devices are used in vision-based plant phenotyping, for instance, hand-holding cameras [142, 64, 50, 55], cameras on unmanned aerial vehicles (UAVs) [157] and ground vehicles [42, 9].

Although the community of vision-based plant phenotyping is growing fast, a noticeable issue is lacking data, especially perfectly annotated data.

2.3 Motivation of Exploring Learning with Fewer Labels

The breakthroughs of CNNs and deep learning in the last decade heavily relied on supervised learning, where the pairs of input data and its corresponding annotations are used to train the models. In the age of information explosion, it seems easy to collect data and the corresponding labels, so do we really need to trap ourselves with limited labels? There are three reasons that motivate us to explore effective learning methods with fewer labels:

1. Annotating data is time-consuming and expensive, especially for large-scale dataset.

2. In a lot of areas such as agriculture and medical image analysis, lacking data and labels, or only having weak labels, are still very common.

This can be caused by factors like shortage of area experts and low budgets; sometimes the data itself is ambiguous for labeling, too.

3. Humans can naturally learn a concept or how to complete a task with limited exemplars and ambiguous description. The exploration of learning with few or weak labels dramatically boosts our understanding of how the algorithms, especially neural networks, interact with the data, and thus promotes the human-like learning paradigm to machine.

To summarize, learning with less annotations is important because it not only attempts to solve the real scenarios of lacking labels and weak annotations, but also enhances our understanding of neural networks and may lead to a more human-like and efficient learning paradigm.

The following sections will introduce three domains that attempt to train the well-performing models with less annotations, namely active learning, semi-supervised learning, and weakly-supervised learning. Different approaches from these domains will be further explored in the rest of the thesis.

2.4 Active Learning

Active learning is a method that attempts to utilize as little labeled data as possible to train a model that can perform as well as possible [149]. The assumption behind active learning is that redundant information often exists in the source of inputs. Specifically, active learning attempts to identify the most valuable unlabeled data based on some rules, and *queries* the labels of this data from an *oracle* (e.g. human annotator) so that the

learner (e.g. a CNN model) can achieve the desired performance.

2.4.1 Categorization of Active Learning

In terms of the working scenarios, active learning can be categorized to membership-query-synthesis, stream-based and pool-based active learning.

Membership-query-synthesis active learning queries labels from the entire input space, including the synthetic data (e.g. transformed data) [4, 5, 24]. However, the synthetic samples may only be recognizable for the machine but human (e.g. a twisted hand-written number [75]), which makes the membership-query-synthesis approaches unsuitable when using human oracles.

Stream-based active learning decides whether to query labels for each input independently (i.e. the unlabeled data is shown one by one) [28, 69], which is generally good for devices of small memory (e.g. cellphones) and real-time scenarios.

The main difference between the stream-based and the *pool-based* active learning is that the former can only see one unlabeled sample at a time while the latter can access the entire unlabeled dataset and queries labels for arbitrary number of unlabeled samples [78], which is more suitable for devices of large memory. The working cycle of stream-based and pool-based active learning is shown in Figure 2.1.

2.4.2 Querying Strategy

The *querying rules* shown in 2.1 are the keys of active learning because they decide which unlabeled data is the most valuable to the learner.

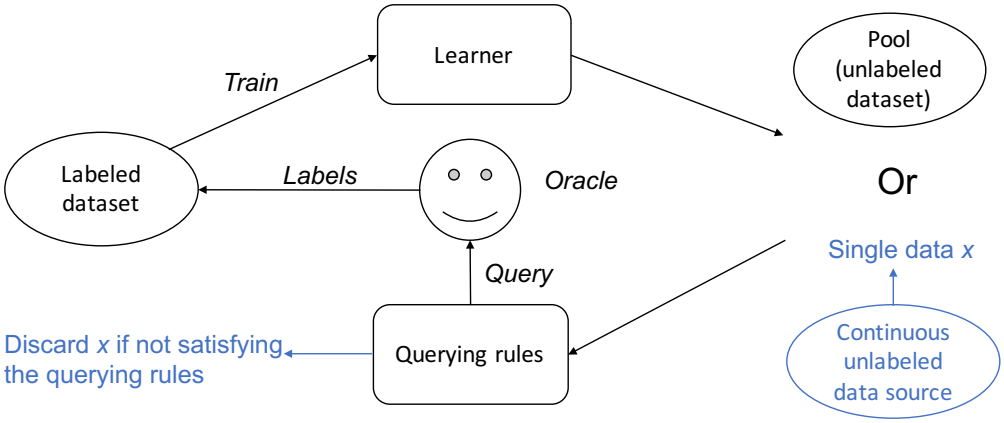


Figure 2.1: Pool-based (black arrows) and stream-based (blue arrows) active learning.

Informativeness-based and *representativeness-based* methods are two typical querying strategies, where the former aim to select the samples that reduce as much uncertainty as possible to the learner, and the latter select the samples that better represent the input distribution [171].

2.4.2.1 Informativeness-based Querying Strategy

Query by committee [122], *support vector machine* (SVM) [25] and *uncertainty sampling* [78] are the well explored approaches among informativeness-based strategies [73].

Query by committee takes advantage of the inconsistency of the predictions on the same data from multiple models of different parameters, and queries the data that have the most inconsistent predictions.

Support-vector-machine-based methods query the data closest to the linear decision boundary [68, 15] because these samples are the most ambiguous.

Uncertainty sampling queries the unlabeled data that is most uncertain to the learner by some *acquisition functions*, for example, *max entropy*

[124], *max variation ratios* [38] and *min standard deviation* [3, 61]. Multiple acquisition functions on pool-based active learning will be evaluated in Chapter 3.

As the informativeness-based approaches rely on the learning model itself to select the most informative unlabeled samples, the labeled dataset, which initializes the model, is important. A labeled dataset of small size and bad quality may cause the model to choose the less useful unlabeled data.

2.4.2.2 Representativeness-based Querying Strategy

Representativeness-based strategies, including *density-based* [39, 159] and *diversity-based* [15, 159, 54] methods, aim to find the unlabeled samples that well represent the input structure.

Density-based methods query data that occur in the high-density input space because these instances well represent the input distribution; this way of selecting instances naturally eliminates the impacts from the outliers.

A diversity-based strategy aims to select the most diverse sample which can better describe the entire input space. However, the influences of outliers are not negligible and a misrepresented distribution can be produced when using this approach. Therefore, diversity-based methods are often combined with other querying techniques. For example, one can integrate support vector machine with diversity to query the unlabeled data that is both close to the decision boundary and maximizing the diversity (e.g. Euclidean distance) between itself and the last queried sample. The support vector machine in this example reduces the risks of selecting the outliers and misleading the learning model.

The representativeness-based strategies often require a large unlabeled dataset and a proper clustering approach to find the most representative samples, which may lead to slow convergence.

2.5 Semi-supervised Learning

2.5.1 Introduction

Semi-supervised learning [18, 175, 173] aims to utilize a dataset with a smaller *labeled* subset and a larger *unlabeled* subset to train a well-performing model. Different from active learning, semi-supervised learning directly explores the unlabeled data without querying the labels from the human annotators to approach the true distribution of the inputs.

Semi-supervised learning can be divided into *inductive* and *transductive* approaches [18, 175, 176]. The difference between them is that the former attempts to learn a classifier or a regressor to fit the distribution of the inputs, while the latter directly explores the relationship from the observed data points.

2.5.2 Inductive Methods

Inductive semi-supervised learning prospers in the era of deep learning with the appearance of more and more powerful CNNs and generative adversarial nets (GANs) [46]. Inductive semi-supervised learning can further be categorized into *generative models*, *pseudo-label* methods and *consistency-regularization* approaches [171, 125, 143].

We will evaluate consistency regularization in Chapter 4 and an ap-

proach combining consistency regularization with pseudo-labeling in Chapter 5.

2.5.2.1 Generative Models

In semi-supervised learning, generative models explicitly model the generating process of data points from both labeled and unlabeled samples, which mainly consist of mixture models, generative adversarial networks, and variational autoencoders [144].

Mixture models are powerful when the prior knowledge of the data distribution $p(x|y)$ is known. For example, assuming $p(x|y)$ a mixture of Gaussian distributions so its parameters can be estimated using expectation maximization (EM) [29] with the observed data points. When $\hat{p}(x|y)$ (estimation of $p(x|y)$) is obtained, one can assign a label c to an unlabeled sample x that maximizes $\hat{p}(x|y)p(y = c)$. However, it is rare to possess the correct assumption of the data distribution in advance, so applying mixture models in real life is difficult.

A *variational autoencoder (VAE)* [67] usually consists of an encoder that encodes original inputs into latent variables and a decoder that attempts to recover the original inputs from the latent variables. In semi-supervised learning, a classifier can be added to the encoding stage to predict labels on the unlabeled data. The latent variables generated from the encoder are then augmented with the predicted labels (on unlabeled data) and the true labels (from labeled data) before forwarding to the decoder [66]. The classifier is expected to correctly classify the unlabeled data when the VAE is properly trained.

A *generative adversarial networks (GAN)* [47] simultaneously trains a

generator to generate samples that are hard to differ from real data and a discriminator to distinguish between real and fake (i.e. generated) data. In most cases, the discriminator of a GAN is a binary classifier to classify inputs into real or fake, but they can also be altered to incorporate true classes of those inputs and hence to predict labels on unlabeled data. For example, Springenberg [133] constructs the discriminator to have N outputs, where N represents the number of possible categories of the inputs. While classifying the inputs, the discriminator reserves the ability to distinguish from real or generated data by calculating the entropy on the predictions – the entropy should be low, which means that the predicted class is certain, for real data, and it should be the contrary for generated data. In addition, one can also construct a GAN with two discriminators to classify the inputs and distinguish between real and generated data separately, which is known as a triple adversarial network [79].

2.5.2.2 Pseudo-label Method

Pseudo-label methods essentially utilize the learning model to label the unlabeled data. The pseudo-labels can either be the model predictions themselves or *transferred* from the model predictions, for example, in classification, one can transfer the predictions of categorical probabilities to one-hot pseudo-labels. Pseudo-label approaches include self-training and co-training, where the former pseudo-labels the unlabeled data by the learning model itself, while the latter uses multiple models.

Self-training approaches first train the learning model with the *labeled* data. The trained model then is used to predict on the *unlabeled* data; those predictions satisfying some criteria (e.g. low uncertainty [116, 154] and high confidence [177]) are transferred to pseudo-labels and added to the

labeled set to train the learning model again. This procedure will repeat until the model satisfies the expected performance or all the unlabeled data is pseudo-labeled.

Self-training can be easily applied to the training of most learning models. However, the single model may produce incorrect pseudo-labels, for example, when the predicted confidence is high according to the model but the prediction is actually wrong, which may mislead the model. Moreover, this incorrect pseudo-labeling and misleading prediction can accumulate errors, because the single model cannot correct itself.

Co-training refers to the methods that utilize two or more models to pseudo-label the unlabeled data [14, 153, 102]. For instance, in a *dual-model* scenario, models A and B are first trained on *different* labeled subsets. Same as self-training, the two models only pseudo-label the unlabeled samples whose predictions are high-confidence, but they add the pseudo-labels to the other model's labeled set. In the scenario of three models, aka *tri-training* [172, 49, 112], models A, B and C are trained on three different labeled subsets. The pseudo-labels that are consistently predicted by the other two models will be added to the remaining model's labeled set.

The core of co-training is to explore the disagreement among models' predictions, so the diversity within the data and models is crucial. Training on *multi-view* data offers a natural way to boost the diversity [174, 139, 84] from data. Multi-view data uses different aspects to describe the same objects, such as such as texture and color features, or front views, left views and top views. Hence, the models in co-training or tri-training can be trained on different views of the same samples to boost the distinction. Although multi-view data do not naturally exist in many scenarios, some research shows that the multi-view instances can be generated from the

single-view data [168, 137].

To promote the diversity with respect to the models, one can apply different hyper-parameters [152] or using a combination of various algorithms [158].

2.5.2.3 Consistency Regularization

Consistency regularization [135, 1, 36] explores two important assumptions of data distribution in semi-supervised learning – the *low-density* assumption and the *smoothness* assumption. The former assumes that the decision boundary should lie on the low-density region in the input space. The latter assumes if two instances are close to each other in the input space, their labels should be the same or similar.

Semi-supervised support vector machines (S3VMs) [145, 59, 20, 86] are one of the earliest, most famous methods that explores low-density separations. S3VMs aim to to find the decision boundaries crossing the low-density area that can still correctly classify the data. The research of S3VM largely focuses on developing optimization approaches [12, 59, 19] to identify the proper boundaries. However, the generalization ability of S3VMs may decrease when multiple distinct classification boundaries satisfying both the low-density assumption, and correctly classifying the data, are found.

Teacher-student structures explore the smoothness assumption – the model predictions of an instance and its perturbed counterpart should be consistent. In practice, the object of teacher-student models is to minimize the difference between the predictions of the original input and its perturbed version by using some cost functions such as mean square error

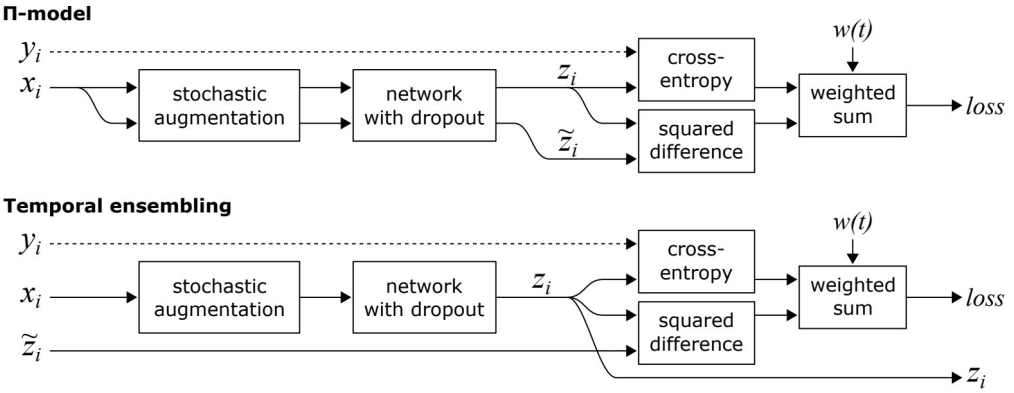


Figure 2.2: ⊓-model (top) and Temporal Ensembling (bottom). Image source: [74].

(MSE), cross-entropy, or Kullback–Leibler (KL) divergence. The ⊓-model [74], the Temporal Ensembling [74], and the Mean Teacher [140] are the representative teacher-student models.

The ⊓-*model* first applies the stochastic noise to the data to create two disturbed samples for every input. Besides, dropout [134] is applied to the learning model to make every prediction slightly different. For both labeled and unlabeled data, the ⊓-model forces the learning model to predict consistently on the disturbed versions of the same input, by applying a *consistency loss* (e.g. MSE or KL divergence) to the predictions. For the labeled data only, a *supervision loss* (e.g. cross-entropy) between the predictions and the labels is also calculated. The final loss is a combination of the supervision loss and the *weighted* consistency loss. In practice, the weight of the consistency loss ramps up (e.g. by a sigmoid function) during training, because the learning model is assumed to become increasingly reliable. Figure 2.2 (top) shows the structure of the ⊓-model.

The ⊓-model illustrates a standard paradigm of the teacher-student methods; that is, using the teacher model to supervise the student model, although only one model plays both roles in the ⊓-model. However, when the labeled set is small, the learning model may be unreliable, especially at

the early training stages, and hence keeps predicting contradictory labels on the unlabeled data in different training epochs, which further worsens the property of the model.

A natural improvement on the $\sqcap$-model is to use ensembling on the model predictions: (1) at every training epoch, ensemble the predictions on the same inputs using the dropout model; (2) use historical ensembles across the predictions from different training epochs. The former requires to forward the same inputs multiple times to obtain the prediction ensembles (e.g. by averaging all the predictions), which costs much computational power and time. The latter requires to store the historical predictions and hence increases memory requirements. The *Temporal Ensembling* (Figure 2.2 bottom) upgrades the $\sqcap$-model by using the historical ensemble of predictions. Instead of saving all the previous predictions, the Temporal Ensembling updates the teacher's predictions using *exponential moving average*, which costs much less memory. The Temporal Ensembling stablizes the training and boosts the performance of semi-supervised learning by using the historically-ensembled predictions to supervise the learning model. However, the prediction to each input can only be updated once in every training epoch, which is slow considering the learning model is optimized more frequently (i.e. at every training iteration).

Rather than ensembling the model outputs, the *Mean Teacher* explicitly creates a teacher model, which is the historical ensemble (i.e. using exponential moving average) of the student model, to supervise the student model. Thus, the teacher model can be updated at every training iteration after optimizing the student model. To note, in the $\sqcap$-model and the Temporal Ensembling, the student model and the teacher model are actually sampled (i.e. using dropout) from the same model trained with the supervision loss and the consistency loss, but in the Mean Teacher,

the student model is optimized by the losses before updating the distinct teacher model. In addition, the Mean Teacher applies random translations and horizontal flips to the input data, instead of Gaussian noise. More details of Mean Teacher will be discussed and evaluated in Chapter 4.

2.5.3 Transductive Methods

Different approaches of *inductive* semi-supervised learning are introduced above, which all train the learning models on both the labeled and the unlabeled data to fit the data distribution. On the other hand, the *transductive* methods directly look for the relationship between all the labeled and unlabeled data points.

Graph-based methods [136] are the most popular in transductive semi-supervised learning. Graph-based methods explore another important assumption of data distribution, the *manifold* assumption, which assumes that the input space consists of multiple low-dimensional manifolds and the data points from the same manifold should have the same label (the smoothness and the low-density assumption are introduced in Section 2.5.2.3).

Graph-based methods construct a graph where the nodes represent the observed data points and the edges represent the similarity [13] between different nodes, so the labels (from the labeled data) can be propagated (to the unlabeled data) with respect to the edges.

There are two major issues in the graph-based methods. The first problem is that the graph is fixed after construction, and so is the label propagation. Hence, the construction of the graph significantly affects the performance [164, 147, 53]; for example, if there are only few data points or the observed instances are unable to represent the input

space, the produced results will be poor. The second downside is that it is computationally-expensive to construct an adequate graph; the necessity of building a new graph when observing new data points even worsens the problem of computational power and storage.

2.6 Weakly-supervised Learning

2.6.1 Introduction

As its name implies, *weakly-supervised learning* refers to the methods that enable the models to learn with weak supervision. Weak supervision can be categorized into *coarse-grained* and *inaccurate* supervision, where the former indicates learning a finer task (e.g. image segmentation) with coarser labels (e.g. image-level classification labels), and the latter refers to learning with inaccurate annotations. These two classes can also appear simultaneously in real applications.

In a broad sense of the word, active learning and semi-supervised learning can also be categorized as weakly-supervised learning (according to [171]) because they use *incomplete supervision* per dataset, that is, the datasets are not fully labeled. However, here weakly-supervised learning is introduced independently because recently it often represents learning with coarse-grained or inaccurate labels.

We will examine a scenario that contains incomplete pixel-level labels *per image* in Chapter 6, which is a combination of semi-supervised learning (i.e. incomplete labels) and weakly-supervised learning (i.e. inaccurate labels).

2.6.2 Coarse-grained Supervision

Coarse-grained supervision means the model is trained on the coarser labels to perform a finer task, for example, learning to perform semantic segmentation, which usually requires pixel-level annotations, with image-level annotations like classification labels. Class activation mapping is a widely used technique in coarse-grained supervision.

Class activation mapping (CAM) [170] is used to highlight the areas on the input images that a trained classification model focuses on with respect to a given category. These areas produce visual clues, such as contours and locations of the objects, using only the image-level labels, so they are naturally suitable for weakly-supervised learning. Class activation mapping is widely used as a visualization tool in deep learning as well.

Figure 2.3 shows how to generate the class activation maps from a classification model in [170]. The authors replace the original fully-connected layers before the output layer in the classification model with the global average pooling (GAP) [88] layers. Therefore, every feature map generated from the last convolutional layer is averaged to a single value and passed to the output layer (i.e. a fully-connected layer) to generate the class scores using softmax. Once the modified classification model is trained, one can directly calculate a weighted sum of the feature maps from the last convolutional layer using the weights of the output layer (i.e. $w_1, w_2, ..., w_n$ in Figure 2.3) with respect to a certain category (e.g. Australian terrier in Figure 2.3). This weighted sum of the feature maps is hence called the class activation map, and it can be upsampled to the same size of the input to identify the regions where the model pays more attention with respect to the given class.

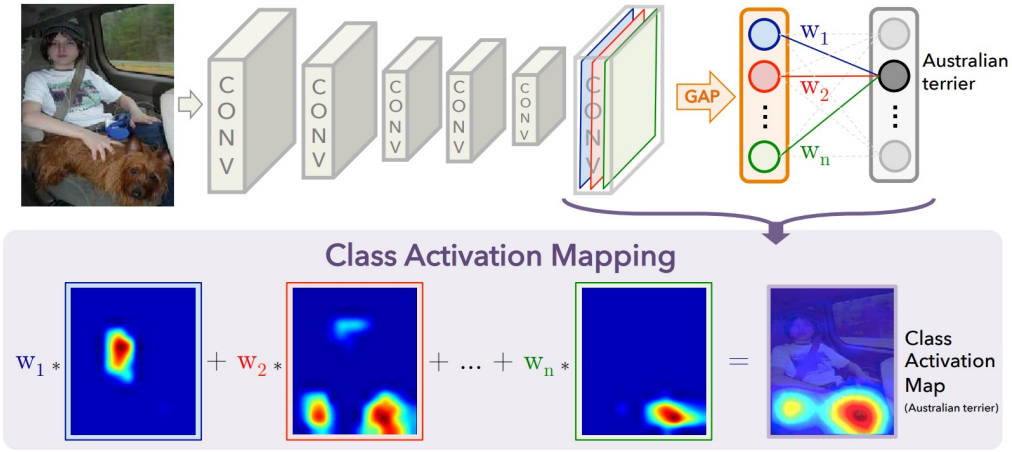


Figure 2.3: Structure (top) and a visual example (bottom) of the class activation map. Image source: [170].

One can train such a model with class labels and use the generated class activation maps to help perform object localization and segmentation of a certain class. However, as the use of global average pooling is compulsory in this kind of class activation mapping, many models need to alter their original structure before generating the class activation maps, which is not ideal, especially for pre-trained models.

Gradient-weighted Class Activation Mapping (Grad-CAM) [121] proposed to compute the weights of the feature maps using global average of gradients so the global average pooling is no longer necessary. Grad-CAM can also generate the class activation maps from shallower layers rather than only the last convolutional layer; this helps to promote the precision of localization because the shallower feature maps often reserve better spatial information, while the deeper ones involve more semantic information and may not be able to form a precise spatial representation due to downsampling.

Further CAM-related work was proposed constantly, for example, *Grad-CAM++* [21] improves the performance of localizing *multiple* objects from the same class. There are also gradient-free approaches which consider the

gradients to be noisy and easy to saturate, such as *Score-CAM* [148] and *Ablation-CAM* [113].

Besides class activation mapping, there are other methods attempting to address coarse-grained supervision. For example, in crowd counting, every human head in each image needed to be point-annotated before training a model, which can result in hundreds of clicks per image. However, in coarse-grained supervised crowd counting, only the numbers of counts (i.e. image-level labels) are provided. To achieve coarse-grained supervised crowd counting, Liu et al. [93] first pre-train a model on a large unlabeled dataset to rank the counts of an image and its sub-images, which is feasible because the counts of the sub-images are guaranteed to be the same or fewer than the counts of the original image; the pre-trained weights are then optimized on a small, weakly-labeled set to further improve the performance. Yang et al. [162] train a crowd counting model using weak labels by adding a sorting branch that sorts the numbers of counts between different images. Bellocchio et al. [11] performs weakly supervised fruit counting by constructing a multi-branch CNN that simultaneously counts the fruits in an image and its sub-images, and then force the consistency between the count from the whole image and the sum of the counts from the sub-images.

2.6.3 Inaccurate Supervision

Inaccurate supervision stands for the scenarios where the model is trained on the labels that are not always correct. These scenarios can be caused by fatigue and negligence of the human annotators which is a common problem in crowdsourcing [163, 146].

Since the release of the ImageNet dataset [30], crowdsourcing has become a popular method of labeling data for developing deep learning algorithms. In crowdsourcing, the annotating jobs are outsourced to people who have been trained for a relatively short time for the given labeling tasks. These people are normally non-experts in the corresponding areas and paid with reasonable prices. The users of crowdsourcing can easily access the platforms like Amazon Mechanical Turk (MTurk) [26] and hire some annotators to label the data, which is a fast and cost-controllable method to obtain labels, especially for the datasets of large scale. The quality of the acquired annotations, however, is not always as desired [131], because the annotators are non-experts, and sometimes it is also difficult to supervise them.

To reduce noisy labels, one way is to take advantage of the *ensemble* methods. This can either be asking multiple annotators to label the same instances [126], or training multiple models [160] on the instance-label pairs, and then use some statistical strategies like majority voting or averaging across the obtained labels or predictions to get their less noisy counterparts.

Another way of decreasing label noise is based on the observation that the noisy level is correlated to the difficulty of the task and the characteristics of the annotators. Thus, the confidence of each obtained labels can be estimated from the mentioned factors using some prior information (e.g. specialties of the annotators) and algorithms (e.g. expectation maximization [114]). Recent research also finds that deep neural networks can tolerate label noise to some extent [166]. In addition, a proper instruction or training for the annotators may be the most effective way to improve the quality of the acquired labels [169, 33, 123].

2.7 Summary

In this chapter, convolutional neural networks, including their key components and how to properly train and evaluate them, and vision-based plant phenotyping are reviewed. Following that, three different methods that enable learning models to be trained with fewer or easier labels are introduced:

- **active learning** begins with a small number of labeled samples and gradually queries labels of the most informative or representative unlabeled samples from oracles (e.g. human annotators) using some querying functions until reaching a given number of training epochs;
- **semi-supervised learning** learns the data distribution directly from a small labeled set and a large unlabeled set by using generative models, consistency regularization, and pseudo labeling;
- **weakly-supervised learning** utilizes coarser-grained labels to supervise finer-grained tasks, or inaccurate (e.g. noisy) labels to train models.

The goal of this thesis is to reduce the labeling expenses and efforts on plant phenotyping and potentially on other computer vision tasks, and hence the three methods mentioned above will be explored progressively in the following chapters.

In particular, we start from exploring different acquisition functions of active learning (introduced in Section 2.4.2.1) on flower classification to train the models with less labeled data in Chapter 3. Afterwards, we explore consistency regularization (introduced in Section 2.5.2.3) and pseudo-labeling (introduced in Section 2.5.2.2) of semi-supervised aphid counting

and crowd counting in Chapter 4 and 5, which also requires fewer training images but does not need the oracles. Finally, we explore learning with fewer pixel-level labels per image on wheat spikelet localisation and crowd counting in Chapter 6, which reduces the number of labels in every training image; it can also be deemed as inaccurate supervision (introduced in Section 2.6.3).

Chapter 3

Active Learning on Flower Classification

3.1 Introduction

This chapter has led to a publication at the Workshop of Computer Vision Problems in Plant Phenotyping (CVPPP) in the Conference on Computer Vision and Pattern Recognition (CVPR) 2019 (Section 1.4 3).

The quality and quantity of labeled data are both important for supervised learning, but labeling data is time-consuming and expensive, especially in the areas that require professional knowledge, for example, plant phenotyping. On the other hand, some data may be more informative than the others, which gives them priority to be labeled. *Active learning* utilizes this property to actively identify and label the most informative data in each acquiring loop until the desired performance of a learner is achieved. In other words, active learning reduces the total number of labeled data by exploiting the training datasets more efficiently.

In this chapter, we apply active learning to *flower classification* to reduce the demand of labeled data. Specifically, we evaluate different *acquisition functions*, which are the key components of active learning, on the VGG 17-category flower dataset [103, 104, 105, 106].

3.2 Active Learning

3.2.1 Pool-based Active Learning

As mentioned in Chapter 2, active learning can be categorized into membership-query-synthesis, stream-based and pool-based methods. In this chapter, the *pool-based* methods are explored because they are popular [115].

Pool-based methods select the most representative samples from the pool of unlabeled data and to query the information sources (e.g. human annotators) for their annotations. A typical pool-based dataset is formed by a small *initial training set* D_{train} and a relatively large *pooling set* D_{pool} at the beginning, where the data in D_{train} are labeled and the data in D_{pool} are unlabeled. As the training proceeds, more samples (i.e. pooled samples) are selected from the pooling set based on some rules imposed by the acquisition functions, and added to the training set. A pooled sample x^p is defined as:

$$x^p = argmax_{x \in D_{pool}} a(x, M), \tag{3.1}$$

where $a(x, M)$ is a function of x and the learning model M, which is also called the acquisition function.

Algorithm 1 Pool-based active learning

Require:
The number of training iterations: N
The number of queried samples in each iteration: m
The initial training set: D^0_{train}
The initial pooling set: D^0_{pool}
The learning model at the initialization phase: M^0
for k in range$(1, N)$ **do**
Select m most informative samples from D^{k-1}_{pool} by the acquisition function $a(x, M^{k-1})$
Add the selected samples to the training set to form D^k_{train}
Delete the selected samples from the pooling set to form D^k_{pool}
Update M^{k-1} to M^k using the new training set D^k_{train}
end for

A typical pool-based active learning pipeline is illustrated in Figure 3.1, which is also described in Algorithm 1. In practise, instead of fixing the total number of iterations, one can also train the model until it converges or satisfies some predefined criteria, for example, when the validation errors are low enough.

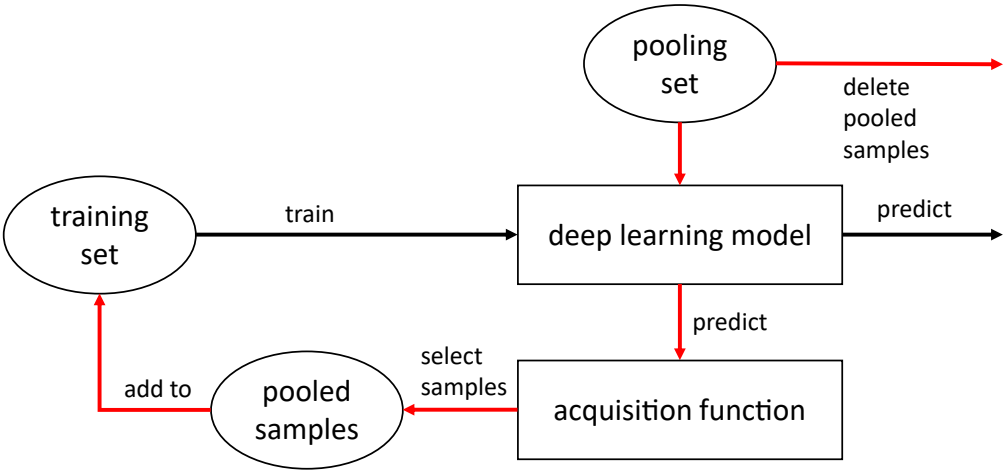


Figure 3.1: Active learning pipeline: the black arrows and the red arrows respectively denote the paths of the training samples and pooling samples

3.2.2 Acquisition Functions

An acquisition function is the key component in active learning. As shown in Equation 3.1, it is this that decides which sample should be queried and hence affects the training speed and the final performance of the active learning algorithm. In this chapter, four different acquisition strategies, which attempt to acquire the most informative samples, are explored, namely random selection, max entropy, max variation ratio and min standard deviation. These strategies are introduced below.

3.2.2.1 Random Selection

Random selection randomly selects new samples from the pool in each acquisition iteration, which is used as a baseline in the following experiments.

3.2.2.2 Max Entropy

In information theory, entropy is a concept denoting the uncertainty; the samples which are more uncertain to the learning models are considered

more valuable for improving the property of the models. Therefore, *max entropy* [124] selects the samples which maximize their predictive entropy. The predictive entropy is defined as:

$$H(y|x, M) = -\sum_{c} p(y = c|x, M) \log p(y = c|x, M), \tag{3.2}$$

where $x \in D_{pool}$, c is a given class, and $p(y = c|x, M)$ is the probability that x belongs to class y predicted by model M.

Max entropy tends to query the samples with similar probability in each class (i.e. high entropy) rather than those with a high probability to one class (i.e. low entropy). For example, in a four-category classification task, x with the predictive probabilities of [0.25 0.25 0.25 0.25] is likely to be queried, while x with the predictive probabilities of [0.85 0.05 0.05 0.05] is unlikely to be queried.

3.2.2.3 Max Variation Ratio

Max variation ratio attempts to maximize the variation ratio (VR, [38]) which is defined as:

$$VR(x) = 1 - \max_{y} p(y|x, M), \tag{3.3}$$

where $x \in D_{pool}$, and and $p(y|x, M)$ is the probability that x belongs to class y predicted by model M.

Instead of measuring the predictive entropy (like max entropy does), the variation ratio uses the *highest predictive probability* to quantify the information of the samples. For instance, in a four-category classification task, x with the predictive probabilities of [0.4 0.2 0.2 0.2] is more likely

to be queried than the one predicted with [0.7 0.1 0.1 0.1], due to the difference between their highest probability.

3.2.2.4 Min Standard Deviation

The last acquisition function that is explored – *min standard deviation* [3, 61] – selects the samples minimizing the standard deviation across their predictive probabilities. This standard deviation (SD) is defined as:

$$SD(x) = \sqrt{\frac{1}{C}\sum_{c}[p(y=c|x,M) - \overline{p}(y|x,M)]^2}, \tag{3.4}$$

where $x \in D_{pool}$, $p(y=c|x,M)$ is the probability that x belongs to class y predicted by model M and $\overline{p}(y|x,M)$ is the average predictive probability of sample x.

It calculates the standard deviation over all classes for each sample, and hence considers the level of disagreement across all classes as a measurement, which is similar to max entropy to some extent – the samples evenly assigned to each class tend to be queried.

3.3 Active Learning on Flower Classification

In this section, we aim to demonstrate that active learning is capable of reducing a large number of labeled images with a small cost of the performance. We evaluate various acquisition functions on flower classification (i.e. the VGG flower dataset) using different learning models (i.e. VGG-16/-19-based). We consider flower classification as a form of plant phenotyping task where the visual traits of the flowers are key to distinguish various species.

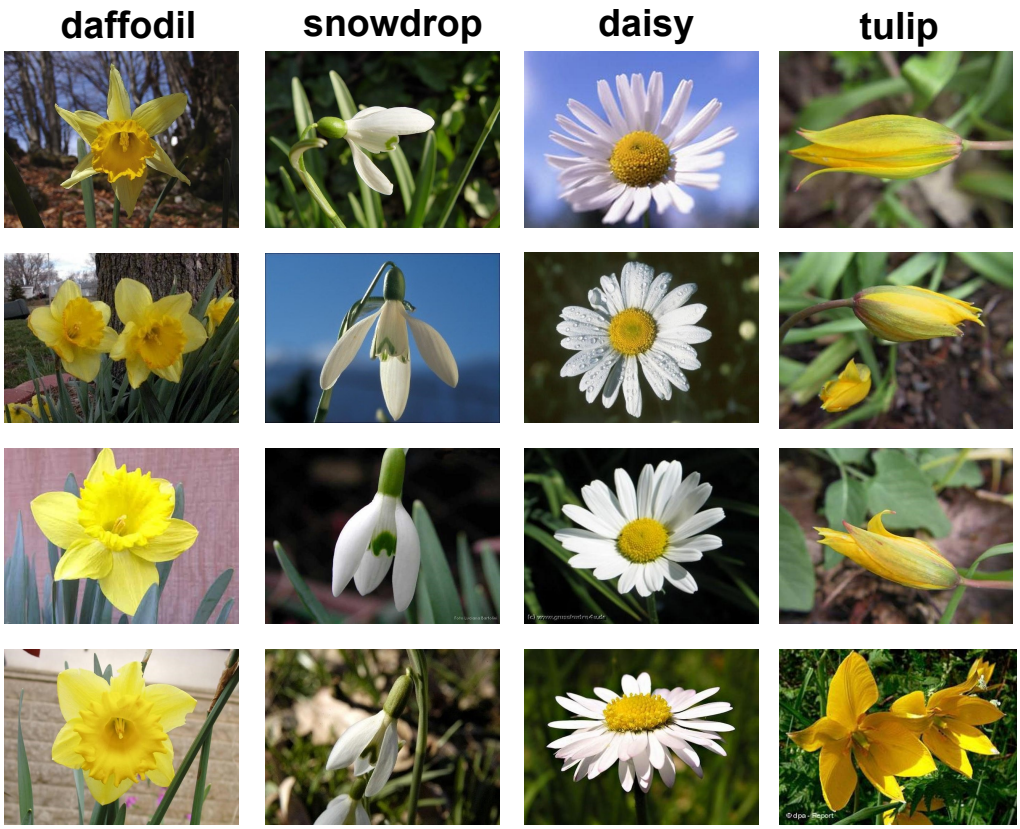


Figure 3.2: Examples of VGG 17-category flower dataset.

3.3.1 VGG Flower Dataset

The VGG flower dataset [103, 104, 105, 106] is proposed by the Visual Geometry Group (VGG) at University of Oxford. The dataset contains 1,360 RGB images of flowers from 17 different categories (i.e. 80 images per class). Figure 3.2 shows some examples from the dataset.

In the experiments below, the dataset is split to the training set consisting of 952 images (70%), the validation set with 204 images (15%), and the test set with 204 images (15%); all the subsets contain even number of images from each flower category. For active learning, the training set is further divided to an *initial training set* containing 68 images and the *pooling set* with the remaining 884 training images.

3.3.2 Implementation Details

We train a VGG-16-based and a VGG-19-based [129] model to perform flower classification. Specifically, in each model, the fully-connected layers and the output layer at the end of the original architecture are replaced with a softmax layer; the remaining layers of the model are loaded with the ImageNet pre-trained weights. We only fit the softmax layer to perform classification while the other weights of the model are freezed.

We compare the performance of different acquisition functions, namely random selection, max entropy, max variation ratio and min standard deviation using fewer images, as introduced in Section 3.2.2, with a model trained using all the training data (i.e. 952 images).

The active learning pipeline is described in Figure 3.1 and Algorithm 1. In short, m images are selected from the pooled set D_{pool} and added to the training set D_{train} every epoch, based on the acquisition function, and the model M is retrained by the new training set. Here we set m to 17 (i.e. the number of categories) and the sizes of the initial D_{train} and the initial D_{pool} are set to 68 and 952 as introduced above. All the active learning models are trained for 35 epochs because the active learning models achieve comparable performance in validation as the model trained by the complete training set at this point. Hence, $17 \times 35 = 595$ labeled images are gradually added to the training set. Rank-1 and rank-5 accuracy are reported on the test set, respectively.

3.3.3 Results

3.3.3.1 Active Learning vs Training on the Complete Dataset

Figure 3.3 shows the test accuracy of training the VGG-16/19-based models on the complete training set (952 images) without active learning, and on the incomplete training set (68 + 595 images) using active learning.

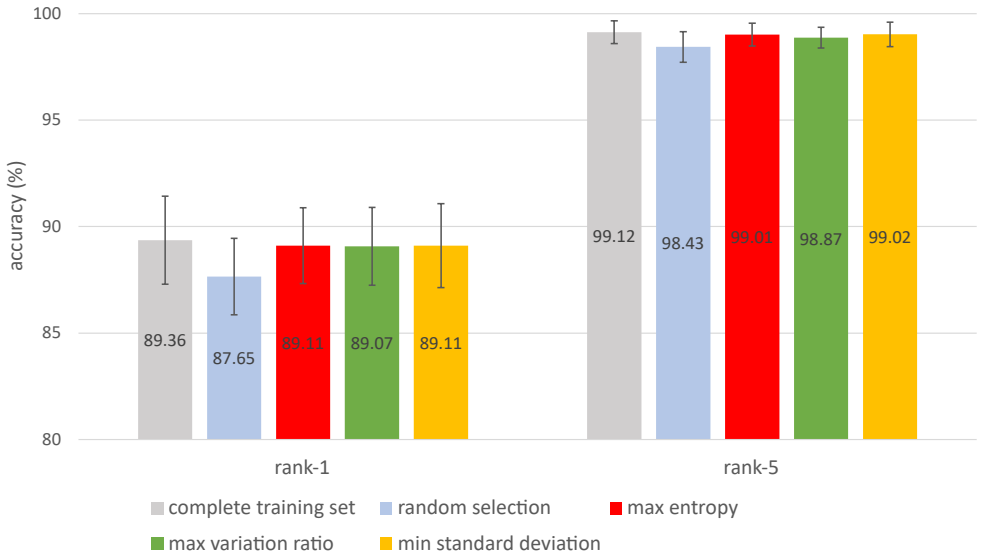
As shown in Figure 3.3, random selection achieves the worst accuracy in both models, although the performance difference between random selection and the others in rank-5 accuracy is small (because rank-5 accuracy is a much looser metric). Max entropy, max variation ratio and min standard deviation all achieve almost the same accuracy as training on the complete dataset, with minor performance gap between each other, by using only 70% training data (i.e. 663 vs 952).

To summarize Figure 3.3, active learning with a proper acquisition function (e.g. max entropy, max variation ratio or min standard deviation) enables the model to learn flower classification with fewer but more informative samples.

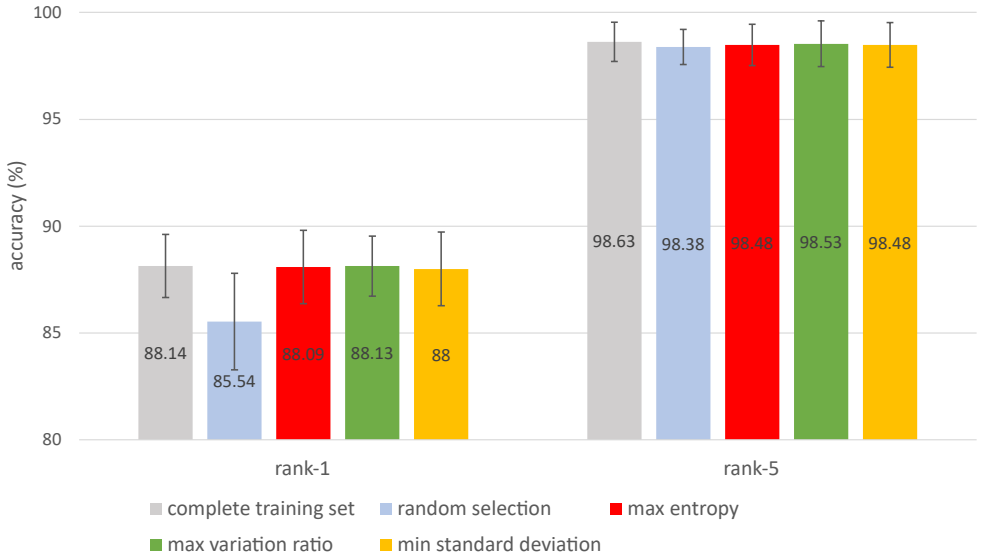
3.3.3.2 The Number of Queried Samples per Class

One may also be interested in which class of flower is most queried by the acquisition function because this may indicate the difficulty of learning a certain species and hence can aid to the data augmentation or class balance strategy, for example, collecting more training images of the difficult-to-classify category.

Figure 3.4 shows the average number (from 10 runs) of pooled images per category at the end of training (i.e. when 595 images in total are

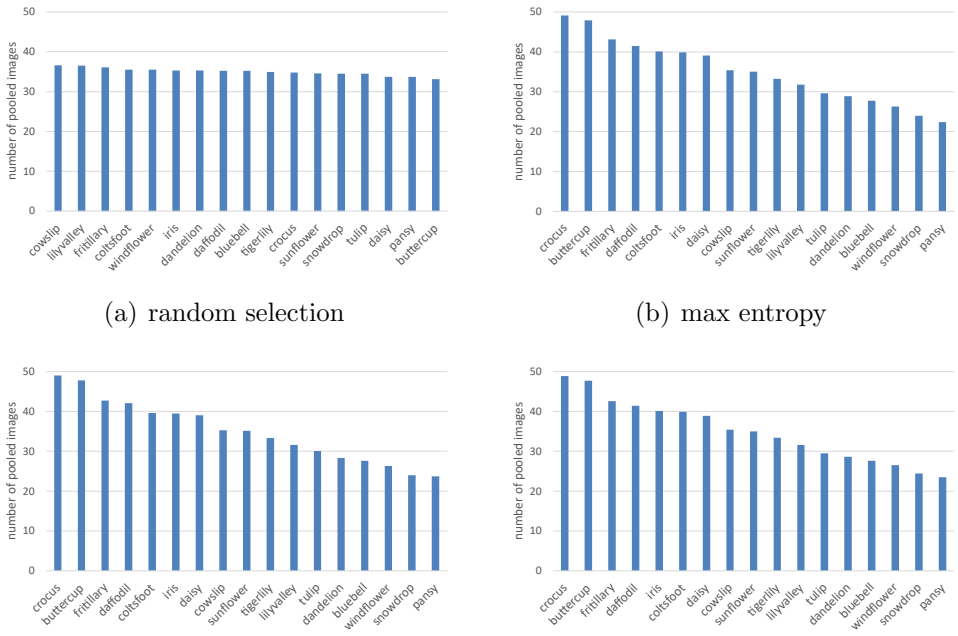


(a) Results of VGG-16-based experiments



(b) Results of VGG-19-based experiments

Figure 3.3: Test accuracy of VGG-16 (top) and VGG-19 (bottom) with different acquisition functions. Each graph reports the mean and standard deviation of the rank-1 (left) and rank-5 (right) accuracy from 10 runs. The "complete training set" indicates training the models on 952 images without active learning, and the remainders are trained on 663 (68 initial + 595 pooled) images using active learning.



(a) random selection

(b) max entropy

(c) max variation ratio

(d) min standard deviation

Figure 3.4: The average number (over 10 runs) of pooled images accumulated per class during training the VGG-16-based model, using different acquisition functions.

pooled) the VGG-16-based model, varying the acquisition functions. As shown in Figure 3.4, random selection evenly queries different classes of images as expected. The other acquisition functions, namely max entropy, max variation ratio and min standard deviation, almost rank the categories in the same order – except for min standard deviation (sub-figure (d)) ranks Iris as 5th and Coltsfoot as 6th, while max entropy and (sub-figure (b)) and max variation ratio (sub-figure (c)) rank the opposite. This may also explain why max entropy, max variation ratio and min standard deviation achieve very similar accuracy. Some visual examples of the 6 most queried categories, namely Crocus, Buttercup, Fritillary, Daffodilm, Coltsfoot and Iris are shown in Figure 3.5 for reference.

In summary, the number of queried images per category indicates that active learning is also able to give clues about what kind of images are more valuable to the model, which may help develop better datasets to improve

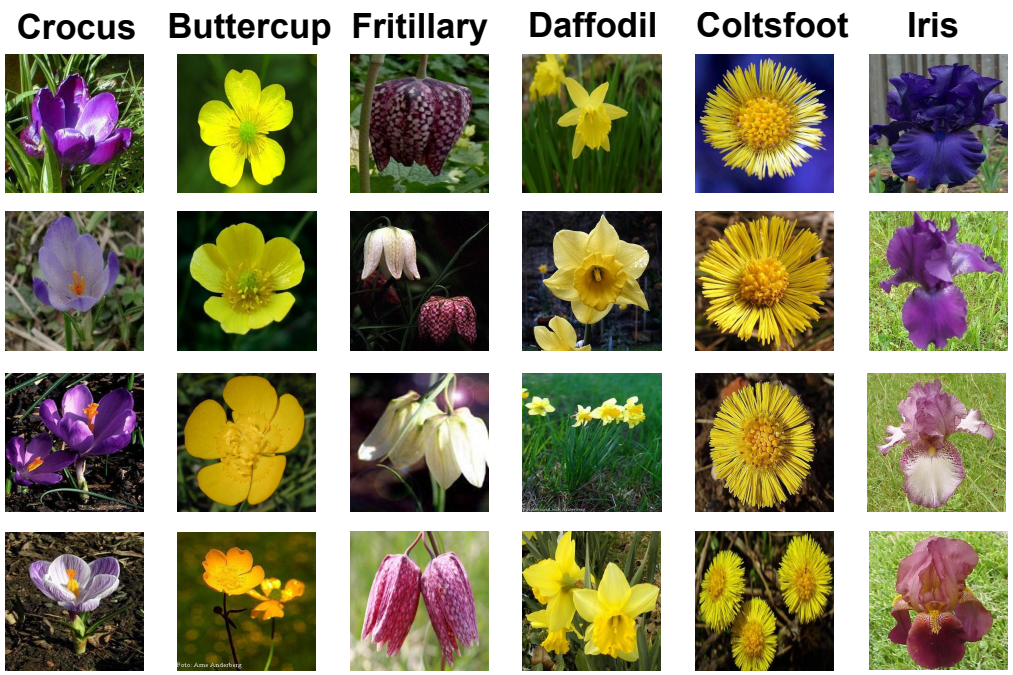


Figure 3.5: Visual examples of 6 most queried flower categories, from left to right, 1st to 6th.

the model's performance.

3.4 Discussion

The effectiveness of active learning is shown in the previous experiments, however, problems in active learning remain. For example, the queried samples are selected based on the current model, but the model is not accurate enough at the early steps. Although this issue can be mitigated by introducing the pre-trained weights, it can still fail to select the most informative samples and lead to bad performance in some complicated tasks. In practice, active learning may need a real-time system for label acquisition to involve the human annotators during model training. It may also require some algorithms to recognize the expertise of the annotators to avoid invalid labels.

3.5 Conclusion

This chapter demonstrates how active learning can be applied to reduce the labeling efforts. The experiments are conducted on flower classification using the VGG flower dataset, which we consider as the first step towards plant phenotyping. Different learning models and acquisition functions are explored which demonstrate the effectiveness and robustness of active learning.

The experimental results indicate that applying active learning can significantly reduce the number of labeled data (e.g. by 30%) without sacrificing performance. It also has the potential to guide data selection as it can indicate the difficulty of learning a certain class. We expect active learning should also be able to expand to other plant phenotyping tasks than flower classification such as leaf counting and segmentation. Despite the problems described in Section 3.4, active learning is worth promoting in plant phenotyping, given the high price and limited availability of good quality training sets in the area.

Chapter 4

Semi-supervised Counting with Adaptive Mean Teacher

4.1 Introduction

As introduced and discussed in last chapter, active learning is able to reduce the demand of labeled samples by exploring the most informative data. However, an oracle (e.g. human annotator) is necessary, which increases the complexity and uncertainty of the active learning pipeline. In this and the next chapter, *semi-supervised learning*, which also aims to leverage minimum labeled data to obtain maximum performance without any external annotator involved, is explored instead. Here, we evaluate semi-supervised learning on density-estimation (heatmap-regression) based counting.

As discussed in Chapter 2, *consistency regularization* is an important concept in semi-supervised learning methods. Consistency regularization assumes that a small perturbation on the same input and learning model does not change the output. Thus, one can add some random noise to an unlabeled sample and force the learning model to predict consistently on these noisy inputs. In practice, researchers perturb the input with Gaussian noise or other image transformation techniques, and perturb the learning model with dropout, before applying a loss function to force the consistency between the predictions on the perturbed counterparts. Mean Teacher [140] is one of the most famous consistency-regularization methods, proposed in 2017. It builds a teacher model, which is the exponential moving average (i.e. historical ensemble) of the learning model (i.e. student model), to supervise the student model with the consistency loss. The details of Mean Teacher will be introduced in Section 4.2.2.

The Mean Teacher approach assumes the teacher model to become more reliable as training proceeds, so it ramps up the weight of the consistency loss until a given epoch. This assumption is generally true but in practice we find the improvement of the reliability of the teacher model

is not monotonous – it can be affected by the difficulty of the input or some bad historical models. Therefore, an *adaptive consistency weight update method* is proposed here and evaluated on two separate tasks: aphid counting - a plant pest phenotyping problem, and the more general and well studied domain of crowd counting.

The proposed method evaluates a teacher model on the labeled training data at every training iteration using some metrics (e.g. Mean Absolute Error (MAE)), before converting this evaluation score to a consistency weight. Although the evaluation is performed on the *labeled* data at each batch, the result is able to estimate the reliability of the teacher model on the *unlabeled* data as well. This is based on the assumption that a model should perform similarly across data from the same distribution, i.e. in our case the labeled and the unlabeled data.

A batch sampler, which guarantees sufficient labeled samples at each batch to stabilize the training in semi-supervised learning, is also used in the proposed adaptive weight update method to increase the reliability of the evaluation. Experiments on aphid counting show the superiority and convenience of the adaptive consistency weight update method in Mean Teacher.

The contributions of this chapter are:

1. exploring the performance of Mean Teacher on heatmap-regression-based counting tasks (i.e. aphid counting and crowd counting) rather than on classification tasks where it is initially and more often discussed;

2. proposing and evaluating a new adaptive (hyper-parameter-free) consistency weight update method based on the performance of the

teacher model per iteration;

3. presenting a new plant phenotyping dataset – in each image, different aphids are point-annotated at their location with an image-level confidence label.

4.2 Related Work

4.2.1 Crowd Counting

4.2.1.1 Introduction

Among all the counting tasks, crowd counting may be the one that has been explored most thoroughly because the images are commonplace, and usually include complex background and a large number of targets. The model developed for crowd counting can be transferred to other counting tasks. In general, crowd counting algorithms can be categorized into detection-based, regression-based and density-estimation-based methods [96, 130].

Detection-based methods aim to detect all the humans existing in a single image, and sum the number of detection boxes to obtain the human counts [81]. In early research, a combination of a feature classifier (e.g. Support Vector Machine (SVM) [25] and Random Forest [41]) and a sliding-window detector is deployed [35] to detect and count people. These methods generally work well in low-density images and they can predict the positions of the humans at the same time. In dense scenarios, however, detection-based approaches may be inaccurate, especially in cases with heavy occlusion. Some researchers propose to only detect part of the human body [89, 155, 37], for example, just the head, to promote the detection

precision in highly congested scenes. Recently, some deep-learning-based detectors which have demonstrated superior performance in the generic object detection tasks (e.g. Faster R-CNN [87]) are also applied to further increase the detection accuracy for crowd counting.

To further address the occlusion issue, regression-based approaches attempt to directly predict the number of counts. Early methods typically consist of a low-level feature extractor (e.g. texture feature extractor [17], an area or perimeter feature extractor [16, 22, 120]) and a separate regression backend (e.g. linear regression [109]). In recent literature for deep-learning-based methods, directly regressing the number of counts is formulated as a weakly-supervised learning task; for example, Yang et al. [162] introduce an extra inter-image count sorting network to obtain more counting-related features. Pre-training and transfer learning techniques can also be adopted to boost the power of the feature extractor. A common problem in regression-based methods is that the spatial information is not well represented. Besides, the supervision signal – only the count – is sometimes insufficient, which makes it difficult to optimise the learning model, especially in complex scenarios where a lot of ambiguous objects appear (e.g. counting people on a street with many cars).

Density-estimation-based methods [77, 110] are often mentioned in recent works. This approach aims to solve the crowd counting problem by mapping the local features with the target density maps. A density map is composed of many Gaussian blobs centered at every object (e.g. human heads); the pixel values of each Gaussian blob sum to one. Therefore, summing all the pixel values within a density map generates the final count of the corresponding image. Density-estimation-based methods incorporate the spatial relationship between the objects and also mitigates the problem of heavy occlusion. In addition, compared to directly regressing the number

of counts or the coordinates of objects, it is easier to learn a Gaussian representation because it is a continuous distribution. The Gaussian blobs are usually generated from the point-annotations on the targets; this may, however, misrepresent the size of individual object because of camera angle and perspective. To ease this problem, size-adaptive Gaussian kernels [167], which calculate the distance between the point-annotations to adjust the size of individual Gaussian kernel, and perspective maps are applied [127].

4.2.1.2 Congested Scene Recognition Network

Congested Scene Recognition Network (CSRNet [85]) is used in this chapter as the base model for both aphid counting and crowd counting. It is a density-estimation-based crowd counting approach proposed in 2018, which consists of a pre-trained VGG-16 feature extractor (i.e. the first 13 layers of VGG-16) and a dilated convolutional backend. The main contribution of CSRNet is introducing dilated convolution to crowd counting for capturing larger receptive fields.

In deep learning, the receptive field measures the size of the patch (area) of the input which contributes to the output feature at any layer [6]. For example, a convolutional kernel is usually considered to have a limited receptive field as it only sees local features; on the contrary, a fully-connected layer has the largest receptive field because it can access every value of an input. A larger receptive field is important for crowd counting because it covers various sizes of target and broader spatial relationship. One may directly use larger or more convolutional kernels to increase the receptive field. For example, Zhang et al. [167] proposes a multi-branch CNN, varying the size of convolutional kernels at each branch, to predict the crowd density maps. Ideally, crowd scenes of different congestion levels

should be dealt with by different branches, but this dramatically increases the number of parameters. [85] also shows that the multi-branch design is inefficient because each branch learns almost identical features regardless different congestion levels of the evaluated crowd images.

Adding more pooling layers can also improve the receptive field at the cost of reducing resolution of the final output. Using up-sampling operations such as transpose convolution and bi-linear interpolation along with pooling layers can expand the receptive field while reserving the output resolution, but again, the number of parameters increases and the details of the input may be lost.

A 2-D dilated convolutional operation is defined as:

$$y(m,n) = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} x(m + r \times i, n + r \times j) w(i,j), \tag{4.1}$$

where y and x represent the output and input respectively, w is a convolutional kernel of size (M, N), and r is the dilation rate. Normal convolution can be considered as a special case of dilated convolution, when $r = 1$. Figure 4.1 shows the an intuitive comparison of normal convolution ($r = 1$) and dilated convolution, which indicates that the receptive field can be expanded without increasing the number of parameters.

The architecture of CSRNet is shown in Figure 4.2. The original paper of CSRNet [85] explores different dilation rates r at the backend while $r = 2$ is adopted in the following experiments because it is reported as the best in [85].

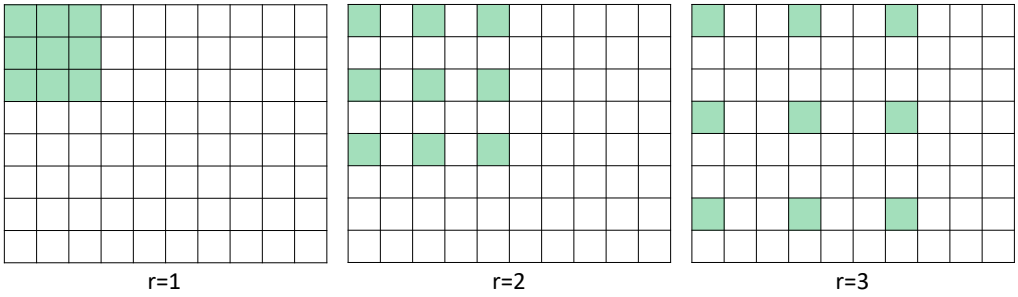


Figure 4.1: The example shows a 3×3 convolution kernel (green) with different dilation rate r. With the same number of parameters, dilated convolution ($r > 1$) captures larger region from the input than normal convolution ($r = 1$).

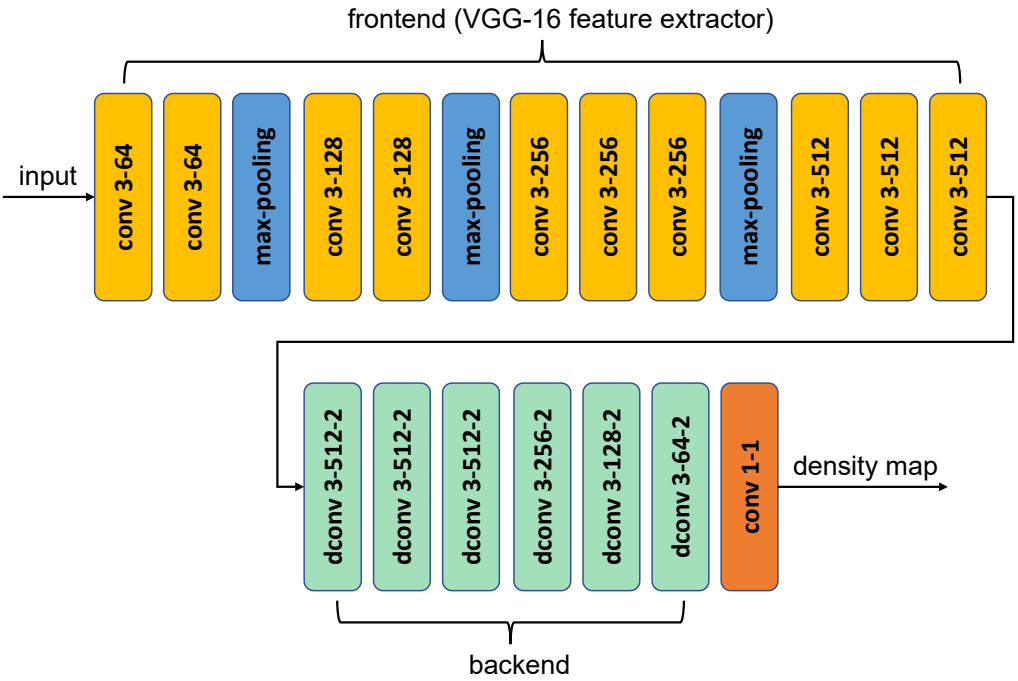


Figure 4.2: The architecture of CSRNet, which consists of a pre-trained VGG-16 feature extractor frontend, a dilated convolution backend, and a 1×1 convolution output head. The normal convolution layers (yellow block) are shown as "conv-(kernel size)-(number of kernels)" and the dilated convolution layers (green blocks) are denoted as "dconv-(kernel size)-(number of kernels)-(dilation rate)". Each convolution and dilated convolution layer is followed by a ReLU activation except the final 1×1 convolution output layer.

4.2.2 Mean Teacher

In the previous section we introduced CSRNet, which is used as the heatmap regression model for crowd counting and aphid counting in this chapter. To reduce the number of labels, saving annotation efforts, we apply a semi-supervised learning scheme, Mean Teacher, to CSRNet.

Mean Teacher [140] is a representative example of a number of consistency-regularization methods in semi-supervised learning. Consistency regularization assumes that the model should predict consistently on the inputs that are slightly perturbed from the same data. As shown in Figure 4.3, Mean Teacher implements a consistency loss function to force the student model θ and the teacher model θ' to predict the same label on an input.

In practice, the student model and teacher model often share an identical architecture but differ from each other by applying some noise to each model (η and η' in Figure 4.3, for example, Dropout). The input to each model is also varied by applying Gaussian noise or other data augmentation techniques to create stronger regularization.

The consistency loss can be formed as a Mean Square Error (MSE) between the predictions of the student model and the teacher model, which is defined as:

$$J(\theta) = \mathbb{E}_{x,\eta,\eta'}\left[\|f(x,\theta',\eta') - f(x,\theta,\eta)\|^2\right] \tag{4.2}$$

The MSE consistency loss essentially applies a larger penalty to larger differences between the predictions from the student model and the teacher model, to force both models to predict the same on the perturbed inputs. Together with a supervision loss $S(\theta)$ for the labeled data, the final loss of

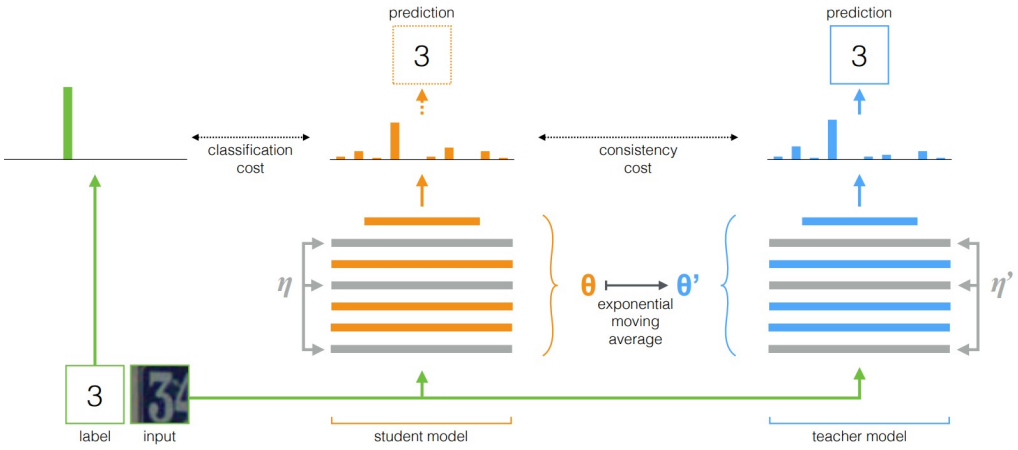


Figure 4.3: Structure of the Mean Teacher model. The weights of the teacher model (θ') is an exponential moving average (EMA) of the student's weights (θ). The models are applied with noise (η and η'), respectively. The aim is to force the student model to predict consistently as the teacher model on the same input. Image source: [140].

Mean Teacher can be defined as:

$$L(\theta) = S(\theta) + \lambda J(\theta), \tag{4.3}$$

where λ is a weight to balance the losses. For labeled data, both $S(\theta)$ and $J(\theta)$ are updated, while for unlabeled data, $S(\theta)$ is zero so only $J(\theta)$ is calculated. In the experiments of this chapter, both $S(\theta)$ and $J(\theta)$ are set as MSE between heatmaps.

Only the student model θ is optimized by back-propagating the final loss $L(\theta)$ and the teacher model θ' is updated as an exponential moving average (EMA) of θ (hence why the method is called Mean Teacher), which is defined as:

$$\theta'_t = \alpha\theta'_{t-1} + (1-\alpha)\,\theta_t, \tag{4.4}$$

where t represents the index of iteration and α is a smoothing weight to avoid the contribution of θ_t (i.e. the current student model) becoming too small.

In fact, Mean Teacher is not the first model to use exponential moving average in consistency regularization. As introduced in Section 2.5.2.3, another consistency-regularization-based semi-supervised learning approach, Temporal Ensembling [74] utilizes the EMA of the historical *predictions* to supervise the learning model. However, Mean Teacher considers using EMA on predictions is slow-paced, especially for large datasets, because each prediction can only be updated once per training epoch. Also, it is difficult to implement EMA on predictions for online training with a continuous data stream. Therefore, Mean Teacher updates the teacher model as the EMA of the student model after every training iteration, to be more up-to-date.

To note, the teacher model is a true historical average of the student model until a preset upper bound of α is reached; α is fixed to the preset upper-bound value during the rest of updates. This means that the contribution from the student model at the current iteration to update the teacher model is fixed after a certain number of iterations. In practice, α is set close to 1 (e.g. 0.99 or 0.999 [140]) to allow more intense updates to the teacher model at earlier training stages (when the teacher model is not reliable) and moderate update at the later training stage (when the teacher model becomes stable).

In addition to the consistency weight λ and the EMA decay term α, a batch sampler and ramp-up function are often used for better performance. A batch sampler samples a fixed portion of labeled data in each training batch regardless of how much the labeled data account for in the entire dataset, to make sure the model is capable of learning something confident (i.e. labeled) in each training batch.

The teacher model is assumed to perform better at the later training

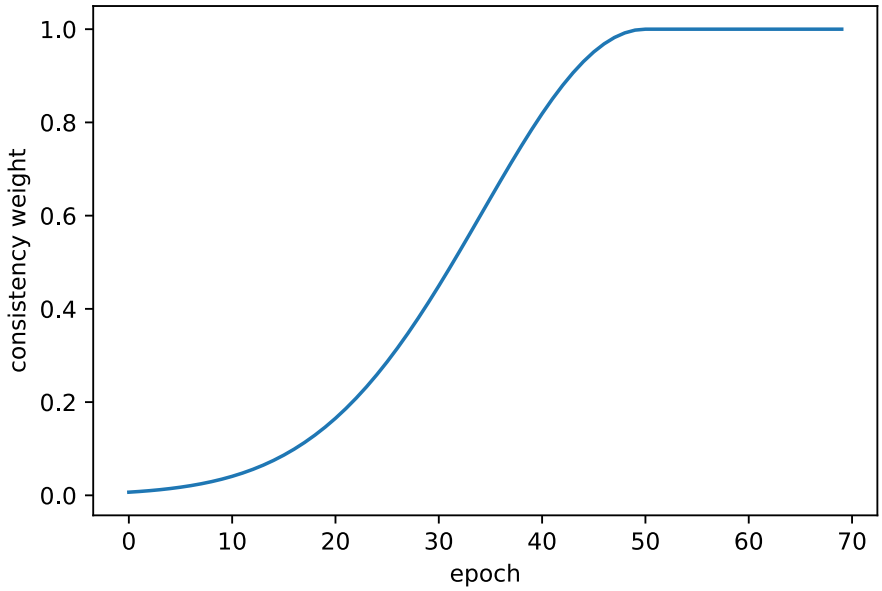


Figure 4.4: An example of using the sigmoid ramp-up function on the consistency weight. The consistency weight is increased from zero to its final value $\lambda = 1$ during the first 50 training epochs, and then remains fixed during the rest of training.

epochs than at the early training epochs because it is a historical ensemble of the student model. Therefore, a ramp-up function, which gradually increases the consistency weight from zero (at the beginning of training) to its final value λ (at a preset number of training epoch), is used to fit this assumption. A sigmoid ramp-up function is used in the following experiments so the consistency weight is varied in different epochs as:

$$\lambda'(e) = \exp(-5(1 - \frac{min(e, E)}{E})^2)\lambda \tag{4.5}$$

where e represents the current training epoch, and E is a hyper-parameter representing the final ramp-up epoch when the consistency weight reach the preset largest value λ. Figure 4.4 shows an example of the sigmoid ramp-up function [141].

4.2.3 Limitations of the Ramp-up Function

As mentioned, the usage of a ramp-up function to update the consistency weight by training epoch is based on the assumption that the later teacher models are more reliable than the earlier ones. This is generally true but the teacher model may not always improve. In fact, the performance of the teacher model fluctuates during training. This is because the optimization of the current student model is not always better than the previous ones, which affects its historical ensemble (i.e. the teacher model). Besides, searching for the best value of E can be tedious. Hence, in this chapter we propose an adaptive consistency weight updating method, discussed below.

4.3 Method

4.3.1 Adaptive Consistency Weight

Different from the ramp-up function, which continuously increases the consistency weight over training epochs, here an *adaptive* weight update method, which is based on an estimation of the performance of the current teacher model, is proposed.

As stated earlier, a batch sampler is often used in semi-supervised learning approaches to maintain a fixed portion (e.g. 50%) of labeled data in every training batch regardless of the percentage of labeled data across the whole dataset. Here we propose to evaluate the performance of the teacher model on the labeled training data in each batch, and then automatically change the consistency weight based on this evaluation result. In this chapter, we use mean absolute error (MAE) to evaluate the performance of the teacher model because MAE is a common metric in count-

ing tasks. The MAE is further developed to the batch-normalized MAE (NMAE), which not only represents the average model performance over a small group of images, but also can be simply transferred to a weight within $[0, 1]$. Below we illustrate how to calculate the adaptive consistency weight from the MAE.

Mean Absolute Error (MAE) is an important metric in counting tasks, which is defined as:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|Y_i - \hat{Y}_i|, \tag{4.6}$$

where N denotes the number of images, Y_i denotes the true count in image i and $\hat{Y}_i$ denotes the predicted count of the same image. Different from the supervision and consistency loss functions, which are formed as MSE between the heatmaps, here the adaptive consistency weight is developed from the MAE between the *numbers of counts* because:

1. the goal is to assess the performance of the teacher model, where calculating the MAE of counts (rather than the pixel errors) is a common practise in counting tasks;

2. it is more convenient to transfer a count error (rather than an error map) to an adjustable value – the adaptive consistency weight.

A normalized MAE is further defined as:

$$NMAE = \frac{MAE}{\overline{Y}}, \tag{4.7}$$

where $\overline{Y} = \frac{1}{N}\sum_{i=1}^{N} Y_i$ is the average number of the ground-truth counts over N images. Hence, the normalized MAE is expressed as:

$$NMAE = \frac{\sum_{i=1}^{N}|Y_i - \hat{Y}_i|}{\sum_{i=1}^{N} Y_i + \epsilon}, \tag{4.8}$$

where ϵ is a small value (e.g. 10^{-8}) to avoid division by zero, and $NMAE \in [0, \infty]$. In addition, $NMAE > 1$ (i.e. the error is even larger than the average ground-truth count) indicates that the model is far below expectation. We further limit NMAE to $[0, 1]$ when transferring it to a consistency weight, which will be introduced later.

Different from the conventional MAE, the normalized MAE provides a percentage error, which is better for comparing the model performance over different groups of data varying the ground-truth counts. For example, if we have two aphid counting datasets with the average aphid counts of 20 and 100 respectively, and we also have a trained model with the MAE of 5 and 10 respectively over the former and the latter datasets, it is difficult to tell where the model performs better directly from the MAE. However, the normalized MAE of the former dataset is $\frac{5}{20} = 0.25$ and $\frac{10}{100} = 0.1$, indicating that the model is more precise over the second dataset. This property of the normalized MAE is beneficial to continuously evaluating the model over different batches of training data because the average count of each batch may vary a lot, considering the batch size is often much smaller than the size of the entire training set.

To note, in each training batch, the normalized MAE is only calculated over the labeled data (because it can not be estimated on the unlabeled data, which have no true counts) using the teacher model (denoted as $NMAE(t)$). Hence the consistency weight λ' is updated at every training iteration as:

$$\lambda' = (1 - min(NMAE(t), 1))\lambda, \tag{4.9}$$

where λ is a hyper-parameter indicating the maximum value of the consistency weight. The former weighted term $w = 1 - min(NMAE(t), 1)$ increases as $NMAE(t)$ decreases (i.e. the teacher model is more reliable)

and vice versa; $w \in [0, 1]$.

A model is assumed to perform equally well or poorly on either the labeled or unlabeled images because both of the labeled and unlabeled data are sampled from the same distribution. As the teacher model is a moving average of the student model, one may argue that the teacher model is likely to fit the labeled data better because their supervision signals are more accurate than the unlabeled ones. However, the evaluation on the labeled training data can still place the upper-bound expectation of the performance on the unlabeled data, especially when the percentage of labeled data per batch is relatively large (e.g. 50%).

We now evaluate the performance of the adaptive consistency weight on two complementary datasets: a novel aphid plant pest dataset, and a more standard crowd counting dataset.

4.4 Datasets

4.4.1 Aphid Dataset

4.4.1.1 Initial Data with Multiple Resolutions

The aphid dataset is provided by School of Bioscience at University of Nottingham including 199 images taken by smart phone with resolution of 1800×4000 (41 images), 4000×1800 (44 images), 1816×4032 (1 image), 4032×1816 (89 images) and 4032×3024 (24 images). The images capture aphids and other insects on wheat plants, which are captured in a glasshouse environment. Individual images can contain one single plant, multiple plants, or parts of a plant; the insects can appear in different

parts of the plant (e.g. wheat spikes and leaves); there are different types of insects and insects in different stages of life (so they have different appearance and size); the photograph angles, lighting, distance between the object and the camera, focus area, and background of the images also vary dramatically. This is therefore a challenging dataset.

The images are annotated by myself using LabelMe [119]. Each image is point-annotated at the centers of the aphids with wings, aphids without wings, dead aphids, and other insects respectively; an image-level label indicating the annotation difficulty (i.e. easy/middle/hard, which can be affected by the size of the targets, the level of the occlusion of the targets, the degree of blur of the image, etc, which may be related to the accuracy of my annotations) of the wingless aphids (because they are the majority), which can be affected by the factors like the level of occlusion and congestion of the targets, is also provided per image. The annotating process took 17 hours 35 minutes (on average 5 minutes 18 seconds per image), producing 199 image-level labels and 15,778 point-annotations (in average 79.29 points per image). Some visual examples of each category are presented in Figure 4.5. The average count per category is summarized in Table 4.1.

Table 4.1: Average counts of different categories per image in the original (before cropped) aphid dataset

Category	Average count
Aphids with wings	1.3
Aphids without wings	68.7
Dead aphids	8.98
Other insects	0.31

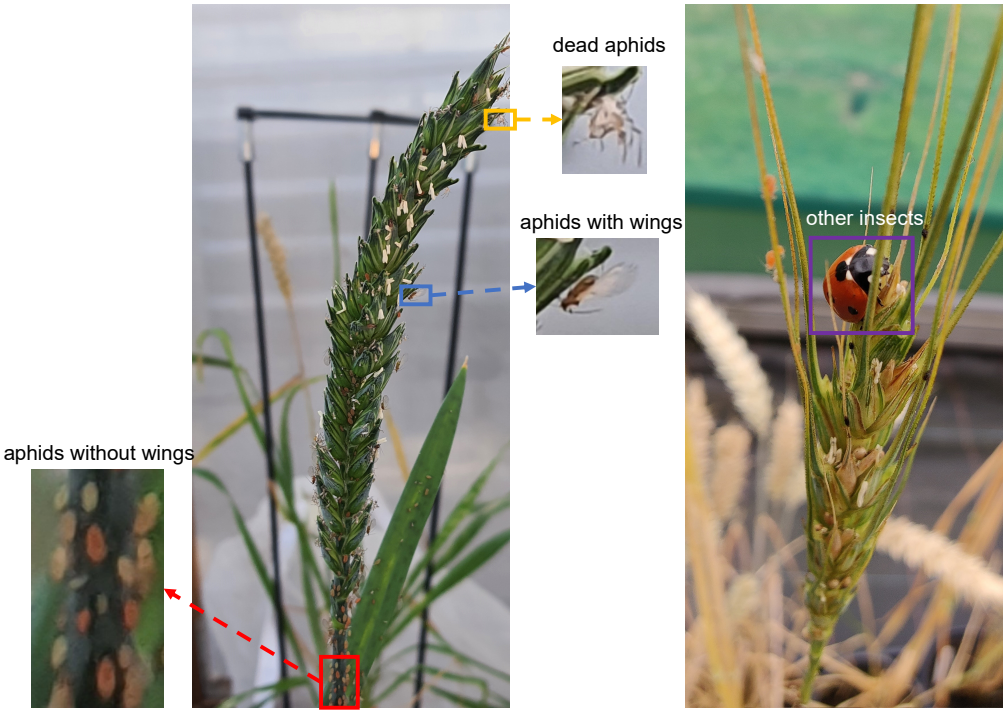


Figure 4.5: Examples of different classes of the original aphid dataset. We crop the images to 1024×768 and train the model to count the aphids with (blue box) + without (red box) wings together in this chapter (details are described below).

4.4.1.2 Cropped Data with 1024×768 Resolution – Used in the Experiments of this and next Chapter

The experiments of this and next chapter only attempt to count the aphids with wings and without wings together (i.e. as a single category, they will be referred to "aphids" below) because they are of more interest to the plant scientists and they dominate the counts (shown in Table 4.1). The annotations of dead aphids and other insects, and the image-level confidence label will be used in future work.

Table 4.2: Statistics of the aphid dataset used in the following experiments. Num is the number of images, Max, Min and Avg count represent the maximum, minimum and average number of aphids in the images respectively.

	Resolution	Num	Max count	Min count	Avg count
Training	1024x768	939	104	0	21.8
Validation	1024x768	255	118	0	20.9

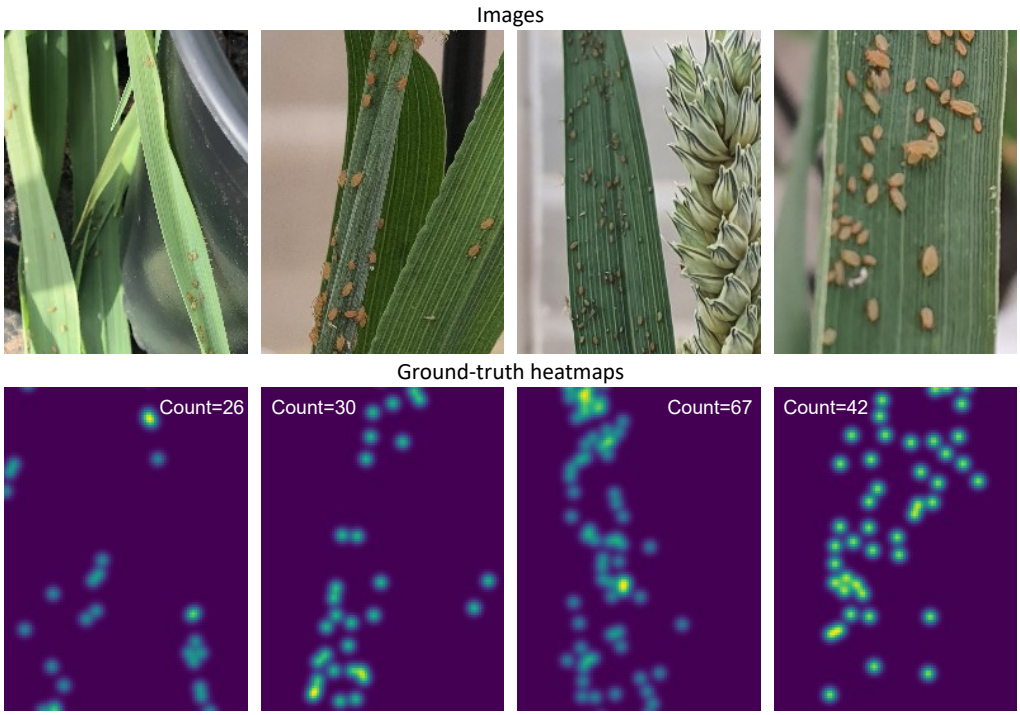


Figure 4.6: Examples of the cropped aphid dataset. Original images are shown at the top and the corresponding true heatmaps are presented at the bottom. Each aphid count (shown at the corner of the heatmap) is the sum of all the pixels of the corresponding heatmap.

In the experiments of this and next chapter, some images are discarded because they have too much background visible; the remaining images are first center-cropped at the target-dense area (e.g. on the wheat plant) and then randomly cropped to 1024×768-resolution images to reduce memory demand. This results in a training set with 939 images and a validation set with 255 images of 1024×768 resolution. This cropped dataset is referred to "aphid dataset" below. Table 4.2 summarizes the statistics of the aphid dataset. For density-estimation based aphid counting, a Gaussian kernel with 15×15-pixel size is applied to every target in each image to generate the corresponding ground-truth heatmap; the heatmap is 8x downsampled to 128×96 to save computation while preserving accuracy. Figure 4.6 shows some images from the aphid dataset and the associated density maps.

4.4.2 ShanghaiTech Crowd Dataset

ShanghaiTech [167] is a crowd counting dataset released in 2016 which consists of 2 parts: Part_A is constructed mostly of images with a large number of people and images in Part_B are relatively less crowded. Table 4.3 summarizes the statistics of the ShanghaiTech crowd dataset. The images in ShanghaiTech dataset are point-annotated on each human's head and the corresponding ground-truth heatmaps are also generated by applying Gaussian kernels on them. Following the same settings as previous researchers [167, 85, 92], a geometry-adaptive kernel (i.e. automatically adjusting the Gaussian kernel size for each point annotation based on the distance to its neighbour point annotations) is adopted to generate the density maps for part A and a fixed kernel (15×15-pixel size) is used for part B. To note, only part B is evaluated here as extended experiments to aphid counting. Figure 4.7 shows some example images and the corresponding density maps of ShanghaiTech part B.

Table 4.3: Statistics of the ShanghaiTech crowd dataset. Num is the number of images, Max, Min and Avg count represent the maximum, minimum and average number of crowds in the images respectively.

		Resolution	Num	Max count	Min count	Avg count
Part A	Training	various	300	3139	33	535
	Test	various	182	2262	62	426.6
Part B	Training	768x1024	400	578	12	122.8
	Test	768x1024	316	539	9	123.7

4.5 Experiments

This section compares the performance between label-only training, standard Mean Teacher, and Mean Teacher with the proposed adaptive consistency weight update approach. Here, label-only training refers to training



Figure 4.7: Examples of ShanghaiTech Part B crowd dataset. Original images are shown at the top and the corresponding true heatmaps are presented at the bottom, with the count of people (i.e. sum of all the pixels of the heatmap) shown at the bottom right.

the model *only* with labeled data – different from semi-supervised learning, where unlabeled data is also considered. The experiments are first performed on aphid counting and then extended to crowd counting to evaluate the generalization.

4.5.1 CSRNet in Mean Teacher

CSRNet is used as the base model across the experiments; if not specified, batch normalization is applied before ReLU at each block of the backend to improve performance. Mean Square Error (MSE) is used as the supervision loss and consistency loss.

In Mean Teacher, the base model is simultaneously optimized by the supervision loss (between the student predictions and the ground-truth labels) and the consistency loss (between the student predictions and the teacher predictions). However, as the teacher model may be unreliable, especially at the early training stage, mixing the supervision loss and the consistency loss may mislead the student model. In the initial paper of Mean Teacher [140], the authors attempt to decouple the supervision task

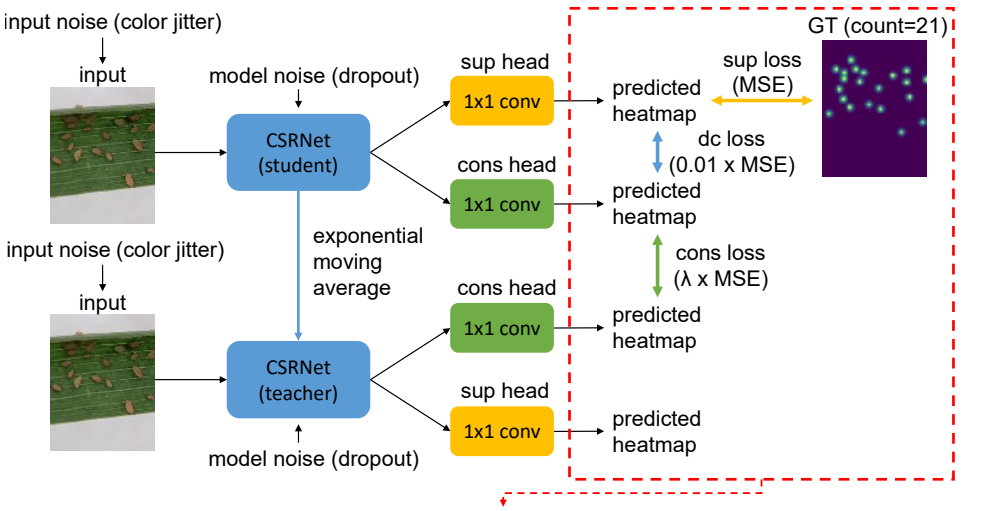


Figure 4.8: Using CSRNet in Mean Teacher for heatmap regression. The sigmoid ramp-up function and the proposed adaptive consistency weight dynamically changes the weight of the consistency loss λ.

and the consistency task by using two output heads, where one is supervised by the supervision loss and the other one is supervised by the consistency loss. We term the former as a supervision head and the latter as a consistency head. An additional weighted MSE is applied between the heads so by tuning the weight, which is usually quite small (as in [140]), one can control the coupling level of the supervision task and the consistency task.

Although using two output heads is optional, we adopt this setting on CSRNet – an additional 1×1 convolution layer is added to the end of the model and an MSE loss weighted by 0.01 is applied between the two heads. Figure 4.8 indicates how we use CSRNet as the base model of Mean Teacher. In evaluation stage, we compare the predictions from both heads of the student model and the teacher model with the ground truth, and report the best.

4.5.2 Aphid Counting

4.5.2.1 Implementation Details

Table 4.4 indicates the training setting of the label-only and semi-supervised learning experiments. In terms of data augmentation, for label-only experiments, the training images are first randomly cropped to 512×384 and then horizontally flipped with a 0.2 probability; for semi-supervised learning experiments, the images are further randomly jittered in brightness, contrast and saturation, where the jitter intensity is uniformly drawn from $[0.6, 1.4]$.

In terms of batch size, different values of batch size are tested and the one producing the least error is adopted. For label-only experiments, the batch size is set to 24; for semi-supervised learning experiments, the batch size is set to 16, with 8 labeled images and 8 unlabeled images forced by a batch sampler. The original architecture of CSRNet is adopted for the label-only experiments, and a CSRNet with a dropout layer of 0.5 probability added after the VGG-16 frontend is used in the semi-supervised learning experiments. An Adam optimizer with learning rate of 10^{-5} and weight decay of 5×10^{-4} is used to optimize all the models.

Table 4.4: Different training settings between label-only and semi-supervised aphid counting experiments.

	Data augmentation	Batch size	Model
Label-only	crop+flip	24	CSRNet
Semi	crop+flip+colorjitter	16 (8 labeled)	CSRNet+dropout

Every model is validated on the same original validation set (i.e. 255 images with 1024×768 resolution, no augmentation) after every training epoch. For each model, we record the best Mean Absolute Error (MAE) and the best Root Mean Square Error (RMSE) achieved in validation.

MAE is defined as:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|Y_i - \hat{Y}_i|, \tag{4.10}$$

and RMSE is defined as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2}, \tag{4.11}$$

where N denotes the number of images, Y_i denotes the true count in image i and $\hat{Y}_i$ denotes the predicted count of the same image.

MAE and RMSE are common metrics to measure how the learning model perform in the regression tasks (e.g. heatmap-regression based counting tasks). Their definitions show that MAE equally weights all the errors before calculating the average, while RMSE applies larger weights to the larger errors (because they are squared).

To reduce the randomness from the model (i.e. from model initialization and optimization) and the dataset (i.e. for $< 100\%$ labeled data, there may exist easier or more difficult subsets), for each evaluated approach, we conduct repeated experiments over 5 runs and report the average and standard deviation of the best validation MAE and RMSE. For 100% labeled data, 5 runs means training the model on the same training data 5 times; for $< 100\%$ labeled data, 5 runs means training the model on 5 different splits of the corresponding number of labeled images. We report the average (over 5 runs) best MAE and RMSE achieved in validation as the performance of each method.

4.5.2.2 Training with Labeled Images Only

First of all, we evaluate training a CSRNet for aphid counting, with only labeled data. We conduct experiments varying the number of training images as follows: 939 (100%), 720 (76.7%, denoted as 75% for easier interpretation, and being more consistent with later crowd counting experiments), 480 (51.1%, denoted as 50%), 240 (25.6%, denoted as 25%), and 96 (10.2%, denoted as 10%) to explore the performance change of reducing the labels.

Figure 4.9 reports the validation MAE and RMSE of training the CSRNet with different number of labeled images only (i.e. without using Mean Teacher). In general, the validation error (i.e. MAE and RMSE) increases as the number of labeled images reduces. This increase becomes notable at 10% labeled data, where the error rate is about 150% of that at full training data (i.e. MAE: 7.1 vs 4.6, and RMSE: 4.6 vs 3). Hence, below we aim to apply and upgrade Mean Teacher to improve the scenario of 10% labeled training data.

4.5.2.3 Standard vs Adaptive Mean Teacher

Successively, we evaluate training the CSRNet for aphid counting with 10% labeled data using different training schemes:

- Label-only: training on 10% labeled data only.
- MT: training the standard Mean Teacher on 10% labeled data and 90% unlabeled data; we empirically set the ramp-up epoch = 5, the EMA decay term $\alpha = 0.999$, and the consistency weight $\lambda = 1$.
- Adaptive MT: training Mean Teacher with the proposed adaptive consistency weight update method on 10% labeled data and 90% un-

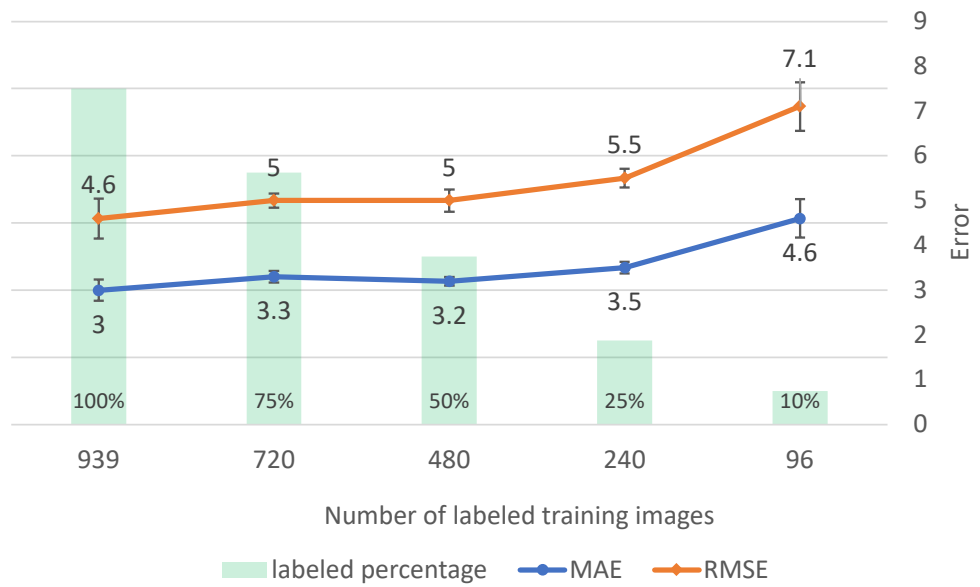


Figure 4.9: Validation MAE and RMSE (lower is better) of aphid counting with different number of labeled training images. The green bars indicate the percentage of the labeled data to the entire dataset. Each pair of the validation MAE (bottom blue line) and RMSE (top orange line) is shown with mean (i.e. the value next to the associated point) and standard deviation (i.e. the error bar on the associated point) over 5 runs.

labeled data; we empirically set the EMA decay term $\alpha = 0.999$ and the consistency weight $\lambda = 1$, which are the same as the experiments of the standard Mean Teacher.

Figure 4.10 reports the validation errors of the different training approaches. Same as before, the validation results are averaged from 5 runs on different splits of 10% training data. Figure 4.10 shows that the proposed adaptive approach consistently outperforms training on both 10% labeled data only, and against the standard Mean Teacher approach. Compared to label-only, the proposed method reduces the validation MAE and RMSE by 17.4% (from 4.6 to 3.8) and 11.3% (from 7.1 to 6.3) respectively; while the original Mean Teacher only decreases MAE by 6.5% (from 4.6 to 4.3), whilst RMSE remains the same. Besides, rather than manually setting the ramp-up epoch, the proposed approach is also hyper-parameter-free, which

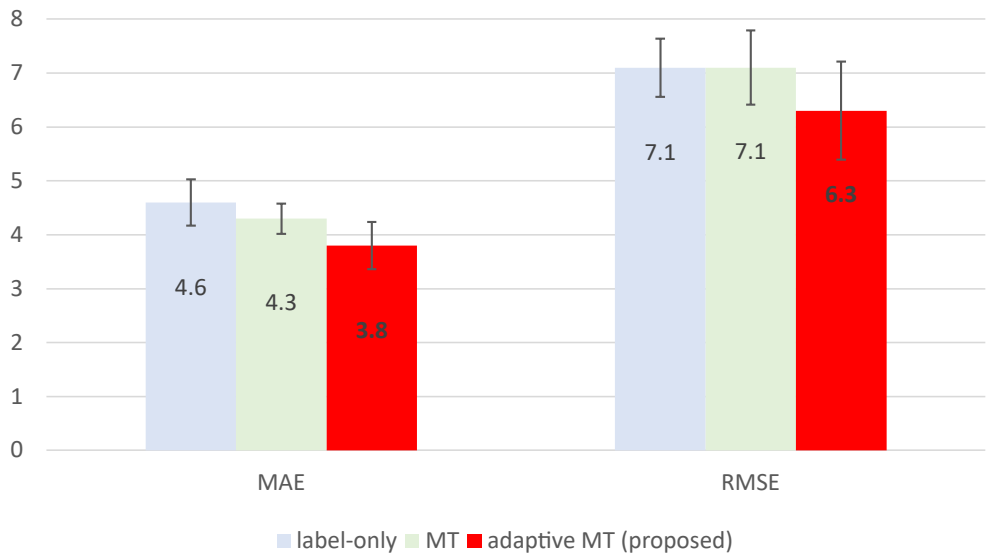


Figure 4.10: Validation MAE (left) and RMSE (right) of aphid counting on 10% labeled training data using different training schemes. Each pair of MAE and RMSE is reported in mean (i.e. the value associated with the bar) and standard deviation (i.e. the error bar) from training on 5 different splits of 10% labeled data.

is more convenient and less subjective.

In addition, to explore the size of the effect, we also conduct a two-tailed Student's t-test to verify the validation results shown in Figure 4.10. The two-tailed student's t-test indicates the statistical significance of if the means from two groups are equal (or unequal). This is realized by comparing the p-value (i.e. the output of the student's t-test) with a preset threshold, which is set to 0.05 in our experiments – $p \leq 0.05$ indicates strong evidence (i.e. statistical significance) of the two means are unequal, and vice versa. The two-tailed student's t-test can be further divided to *paired* and *unpaired* t-test, where the former compares the means of the same item under two different conditions, and the latter compares the means of two independent conditions.

Here we conduct the two-tailed *paired* student's t-test to verify the validation results shown in Figure 4.10 to show if the performance of various

training schemes is significantly different. We use a paired t-test because we compare two different approaches (i.e. two conditions) over *each* of the 5 different training data splits ; in other words, the same split is used in both conditions. The t-test results are shown in Table 4.5, where almost all the p-values are less than 0.05 (i.e. significant at the 0.05 level); only one p-value is greater than 0.05 (RMSE, label-only vs MT (ramp=5)), because these two methods essentially have the same average RMSE (Figure 4.10 blue and green bars on RMSE). Therefore, the t-test results statistically confirm the superiority of the proposed adaptive consistency weight over the standard ramp-up function and not using Mean Teacher.

Table 4.5: P-values of the two-tailed paired student's t-test for aphid counting, verifying the results shown in Figure 4.10. The p-values that are less than the 0.05 significance level is bolded.

	p-value (MAE)	p-value (RMSE)
label-only vs MT (ramp=5)	**0.029**	0.742
label-only vs MT (adaptive)	**0.004**	**0.04**
MT (ramp=5) vs MT (adaptive)	**0.017**	**0.015**

4.5.3 Crowd Counting

The aphid counting experiments show the effectiveness of the proposed adaptive consistency weight update methods in a specific plant phenotyping domain. Here, the generality of the proposed approach will be verified on a more generic task – crowd counting.

4.5.3.1 Implementation Details

The crowd counting experiments are conducted on ShanghaiTech part B (called "ShB" below) using CSRNet. Table 4.6 indicates the training setting of the label-only and semi-supervised learning experiments on crowd

counting.

Table 4.6: Training settings for the label-only and semi-supervised crowd counting experiments. It is the same setup as for aphid counting, except batch size is 16 in the label-only experiments because it generates the best performance compared to the other candidate values of batch size

.

	Data augmentation	Batch size	Model
Label-only	crop+flip	16	CSRNet
Semi	crop+flip+colorjitter	16 (8 labeled)	CSRNet+dropout

An Adam optimizer with learning rate of 10^{-5} and weight decay of 5×10^{-4} is used in the label-only experiments. Another Adam optimizer with the same learning rate and weight decay is applied to all the semi-supervised learning experiments, but the learning rate is divided by 5 at epoch 10 and 200, respectively. Although Adam automatically changes the learning rate during training based on the preset learning rate, we experimentally find that Mean Teacher on ShB is difficult to converge when the preset learning rate is large – that is why we manually reduce the learning rate as noted. Every method is validated on the test set of ShB (i.e. we use this test set as a validation set) after every training epoch, and we record the best MAE and RMSE achieved in validation.

For every method in the crowd counting experiments, we conduct 5 runs of repeated experiments on the *same* split of data – while in the aphid counting experiments, they are 5 *different* splits – to reduce the randomness caused by the model initialization and optimization. It is also convenient to compare the standard and adaptive Mean Teacher models with the literature [94], where the authors also evaluate Mean Teacher on ShanghaiTech part B. Although they may use a different base model and a different split of labeled training data than us, it is still valuable to compare the percentage improvement of the model performance, between their work and ours. We report the average (over 5 runs) best MAE and

RMSE achieved in validation as the performance of each method.

4.5.3.2 Training with Labeled Images Only

Same as aphid counting, the CSRNet is first evaluated without Mean Teacher, varying the number of training samples – 400 (100%), 300 (75%), 200 (50%), 100 (25%) and 40 (10%), respectively. The results are reported in Figure 4.11. Figure 4.11 shows that the performance of the learning model becomes worse as the number of training images reduces. The decrease of model property becomes notable at 10% training data and hence it is used to demonstrate semi-supervised methods below.

4.5.3.3 Batch Normalization over 100% and 10% Labeled Data

Batch normalization is often used to accelerate the training of CNNs. As mentioned before, batch normalization is applied at the backend of the CSRNet used in the experiments in this chapter. Here, we first want to evaluate the impact of using batch normalization over 100% and 10% labeled training data on crowd counting, which may indicate a simple solution to dealing with insufficient training data before using any semi-supervised learning approach. We evaluate batch normalization on crowd counting rather than aphid counting because we consider batch normalization as a more general technique of CNNs, and hence it should be verified on a more generic task – here, crowd counting.

We train the CSRNet with and without batch normalization on 100% (400 images) and 10% (40 images) labeled training data only for 2,000 epochs, with their validation results shown in Figure 4.12. Figure 4.12 shows that within 2,000 training epochs, batch normalization improves the

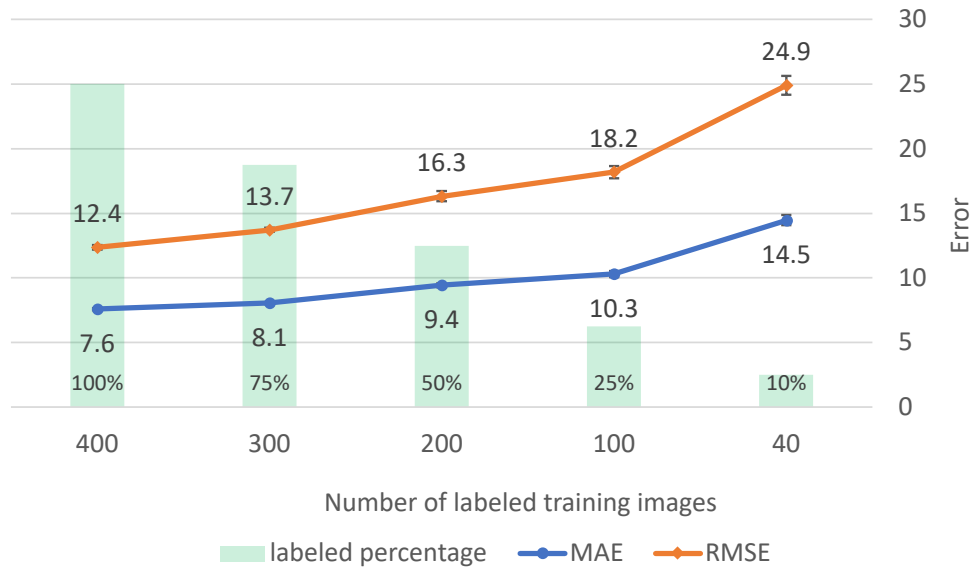


Figure 4.11: Validation MAE and RMSE (lower is better) of crowd counting with different number of labeled training images. The green bars indicate the percentage of the labeled data to the entire dataset. Each pair of the validation MAE (bottom blue line) and RMSE (top orange line) is shown with mean (i.e. the value below/above the associated point) and standard deviation (i.e. the error bar on the associated point) over 5 runs.

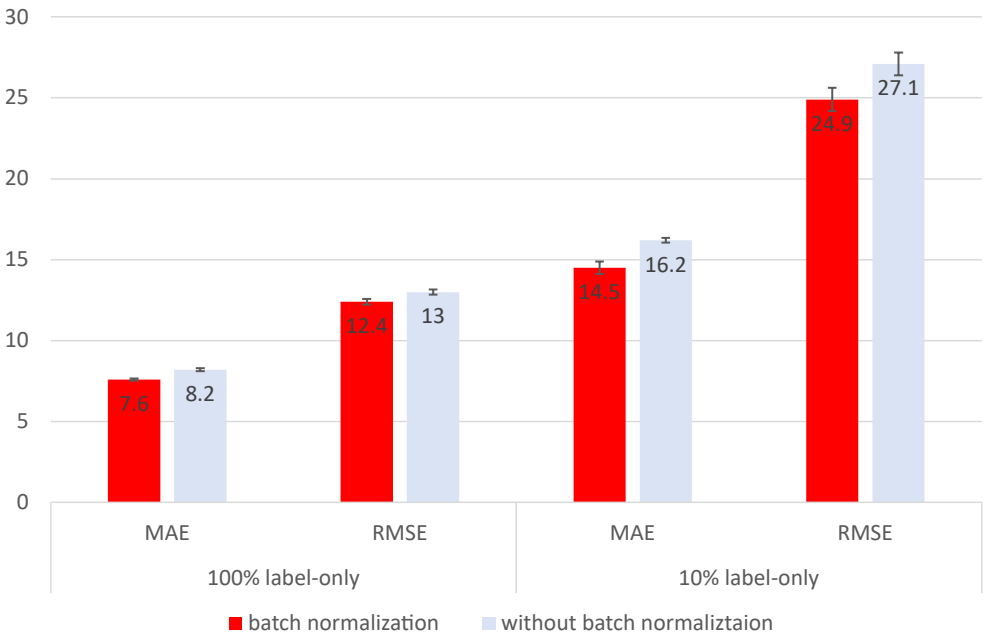


Figure 4.12: Validation MAE and RMSE of training the CSRNet with or without batch normalization on 100% and 10% labeled samples, denoted in mean (i.e. values in the bars) and standard deviation (i.e. error bars). The results are averaged from 5 runs of the model trained on the same split of the training data. Batch normalization (if in use) is applied before ReLU at each block of the backend.

model over both 100% and 10% labeled data, without using any semi-supervised learning approach. This improvement is particularly obvious for the less-labeled-data scenario – using batch normalization reduces the MAE and RMSE over 10% labeled data by 10.5% (i.e. MAE reduced from 16.2 to 14.5) and 8.1% (i.e. RMSE reduced from 27.1 to 24.9) respectively, while the corresponding reduction rates for 100% labeled data is 7.3% (i.e. MAE reduced from 8.2 to 7.6) and 4.6% (i.e. RMSE reduced from 13 to 12.4) respectively.

To note, 2,000 training epochs is quite a large number of epochs for training the CSRNet on ShB. Specifically, the models in the repeated runs barely improve after 1,000 epochs. Therefore, here we conclude that batch normalization is particularly useful for training a CNN with fewer labels in limited but substantial time, which is worth applying before using any semi-supervised learning method. In the remainders of this chapter, batch normalization is still used at the backend of the CSRNet, for both label-only and semi-supervised experiments.

4.5.3.4 Standard vs Adaptive Mean Teacher

Successively, we evaluate both the standard and the proposed adaptive Mean Teacher on semi-supervised crowd counting. The EMA decay term α is empirically set to 0.999 and the maximum value of consistency weight λ is set to 1, for both standard and adaptive Mean Teacher. Since the proposed adaptive consistency weight can be considered as an substitute of the ramp-up function, we also evaluate Mean Teacher without using any weight changing strategy. Figure 4.13 presents the validation errors of training the CSRNet for crowd counting on 10% labeled data (and 90% unlabeled data if applicable) with:

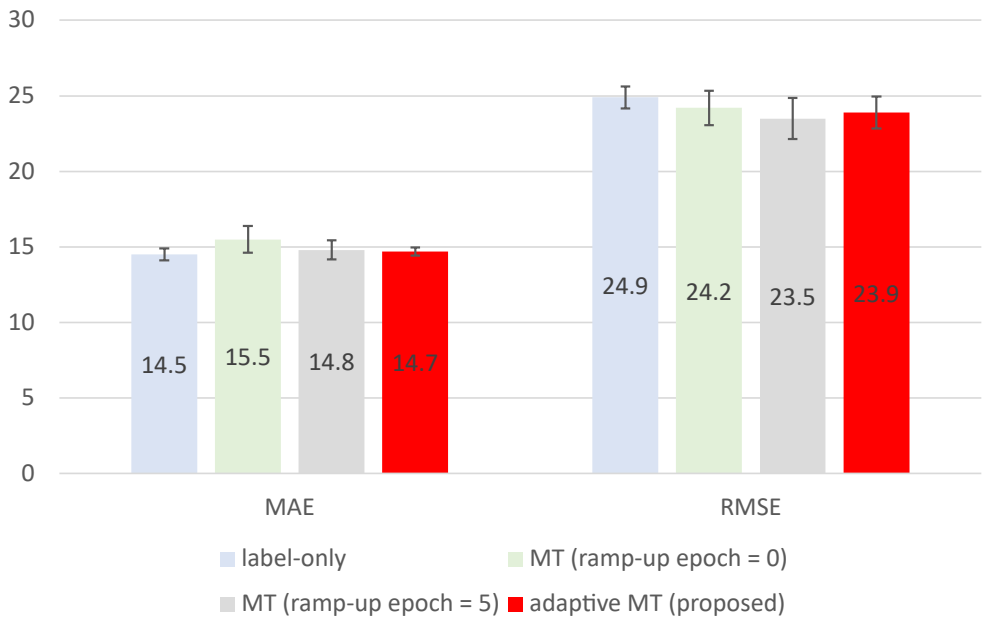


Figure 4.13: Validation MAE (left) and RMSE (right) of crowd counting on 10% labeled training data using different training schemes. Each pair of MAE and RMSE is reported in mean (i.e. the value associated with the bar) and standard deviation (i.e. the error bar) from 5 runs of training on the same 10% labeled data.

- Label-only: training on 10% labeled data only.
- MT (ramp-up epoch = 0): using Mean Teacher with a fixed consistency weight $\lambda = 1$ all the time.
- MT (ramp-up epoch = 5): using standard Mean Teacher with sigmoid ramp-up function, where the ramp-up epoch is set to 5.
- Adaptive MT: using Mean Teacher with the proposed adaptive consistency weight.

The results of using Mean Teacher in Figure 4.13 first indicate that dynamically changing the consistency weight is an improvement because the model that uses the fixed consistency weight (i.e. ramp-up epoch = 0) suffers the largest errors. To note, below standard Mean Teacher refers to the version with ramp-up epoch = 5 rather than 0.

Also, the proposed adaptive method does not dominate the performance in semi-supervised crowd counting – it slightly outperforms the standard Mean Teacher (i.e. ramp-up epoch = 5) in MAE, but fails in RMSE – although the difference is also marginal. It is difficult to draw a simple conclusion that the adaptive Mean Teacher is better or worse than the standard Mean Teacher, due to the inconsistency between the MAE and RMSE. However, here we argue that a method is superior only if it leads in both the metrics, like the adaptive Mean Teacher in semi-supervised aphid counting, and hence we consider that the proposed adaptive consistency weight does not improve Mean Teacher, at least over the ShB dataset.

4.5.3.5 Mean Teacher vs Label-only

In addition, from Figure 4.13, comparing label-only with all the Mean Teacher results, we find that Mean Teacher, even with the proposed adaptive method, is not very effective on ShB – it fails to reduce MAE, though it does manage to reduce RMSE. This time, the MAE of using the standard or adaptive Mean Teacher is *slightly worse* than training on 10% labeled data only, but, conversely, the improvement in RMSE is relatively large.

Why is there a difference between MAE and RMSE performance? As introduced in the definition of MAE and RMSE, MAE equally weights all the errors, but RMSE weights the larger errors more. The marginal change of MAE (i.e. label-only vs standard MT vs adaptive MT = 14.5 vs 14.8 vs 14.7) indicates that applying Mean Teacher (standard or adaptive) may not be able to reduce the average errors. On the other hand, the relatively clear reduce of RMSE (i.e. label-only vs standard MT or adaptive MT = 24.9 vs 23.5 or 23.9) indicates that using Mean Teacher (standard or adaptive) may help to make the model not being too wrong – as the maximum value

of the errors may be reduced.

However, we still consider that applying Mean Teacher does not ease the fewer-label (i.e. 10% labeled data) scenario on crowd counting because only one metric is improved.

As with aphid counting, we also conduct two-tailed Student's t-tests to explore significance of the results shown in Figure 4.13 to ascertain if any of the small difference are in fact significant. However, here we use two-tailed *unpaired* t-test, assuming *unequal variance* in different groups. The unpaired t-test is used here because the different approaches are trained on the *same* split of data for 5 times, and we consider the performance of the different methods is independent i.e. does not rely on the split of the data.

The t-test results are shown in Table 4.7. None of the p-values indicates significance at the 0.05 level. These t-test results again confirm the ineffectiveness of both the standard and adaptive Mean Teacher on semi-supervised crowd counting, using 10% labeled training data of ShB.

Table 4.7: P-values of the two-tailed two-sample unequal variance student's t-test for crowd counting, verifying the results shown in Figure 4.13

	p-value (MAE)	p-value (RMSE)
label-only vs MT (ramp=0)	0.064	0.292
label-only vs MT (ramp=5)	0.365	0.085
label-only vs MT (adaptive)	0.382	0.107
MT (ramp=0) vs MT (ramp=5)	0.210	0.389
MT (ramp=0) vs MT (adaptive)	0.113	0.614
MT (ramp=5) vs MT (adaptive)	0.690	0.654

While not convincing, these results are consistent with previous research. Liu et al. [94] also compared training using only labeled data with the standard Mean Teacher on 10% labeled data on ShB; though note the exact labeled samples may be different from the experiments here due to random split. However, they report only a 1.3% (15.8 to 15.6) improve-

ment in MAE and a 2% (25 to 24.5) improvement in RMSE, which is also not obvious. The results from [94] and our experiments show that Mean Teacher, even with the proposed adaptive consistency weight, is not capable enough to adapt to the scenario with 10% labeled data in ShB; a more powerful semi-supervised learning method will be needed.

4.6 Discussion

The experiments show that the proposed adaptive consistency weight is a more convenient (i.e. no hyper-parameters to set) and more effective method than the original ramp-up function in Mean Teacher, however, only in aphid counting – as Figure 4.10 shows. Moreover, faster convergence is observed with the proposed approach.

Figure 4.13 shows that the performance of Mean Teacher is not satisfying enough in crowd counting – the standard and the proposed adaptive Mean Teacher can only improve the RMSE, with the MAE being slightly worse than directly training on the 10% labeled data. This may be relevant to the complexity of the dataset – ShB has greater average counts (123.7 vs 20.9) and more complex background than the aphid dataset.

In addition, using the exponential moving average of the student model as teacher leads to coupling of the student and teacher model, which means that the student and the teacher model become very similar at later stages of training. We observe that the models are easier to improve when they have larger performance gaps during training, so this coupling limits further improvement of the models. This coupling issue is also reported in [62] where the authors find the Euclidean distance between the weights of the teacher model and the student model is small, and hence they propose to

train both models independently (terming them "dual student" models), instead of using exponential moving average to create the teacher model.

The proposed adaptive consistency weight update method also has its disadvantages. First, it aims to weight the consistency loss based on how good the teacher model predicts on the unlabeled data. However, this performance is estimated, by necessity, on a limited amount of labeled data, which may not be representative. Second, the normalized MAE used to tune the consistency weight favors the images of fewer targets, which may result in the model tending to learn from the easier images (i.e. the images with fewer targets).

4.7 Conclusion

In this chapter, we attempt to apply Mean Teacher to the heatmap-regression based aphid counting and crowd counting tasks, to reduce the demand of labeled data. Conventionally, a ramp-up function, for instance, a sigmoid ramp-up function, is used in Mean Teacher, to increase the consistency weight by the time of training, with the assumption of the teacher model being more and more reliable. However, as the teacher model is updated rapidly (i.e. in every training iteration) as the exponential moving average of the student model, we find the performance of the teacher model generally increases, but fluctuates. Thus, we propose an adaptive consistency weight update approach, which is based on the up-to-date evaluation of the teacher model using the normalized MAE, to replace the standard ramp-up function.

In aphid counting, the proposed adaptive consistency weight outperforms the standard ramp-up function – when training on 10% labeled data,

compared to not using Mean Teacher, using the standard Mean Teacher with the sigmoid ramp-up function reduces the MAE by 6.5% (i.e. from 4.6 to 4.3) without reducing RMSE, while using Mean Teacher with the adaptive consistency weight reduces the MAE and RMSE by 17.4% (i.e. from 4.6 to 3.8) and 11.3% (i.e. from 7.1 to 6.3), respectively. The details are shown in Figure 4.10.

However, neither the standard nor the adaptive Mean Teacher achieves satisfying results on semi-supervised crowd counting. As shown in Figure 4.13, compared to training on only the 10% labeled data, using the standard or the adaptive Mean Teacher achieves slightly worse results in MAE (i.e. label-only vs standard MT vs adaptive MT = 14.5 vs 14.8 vs 14.7), although the RMSE is obviously reduced (i.e. label-only vs standard MT vs adaptive MT = 24.9 vs 23.5 vs 23.9).

The reduction of the RMSE with the plateau of MAE may indicate that applying Mean Teacher reduces the variance of the counting errors, but we still consider that Mean Teacher is not effective in semi-supervised crowd counting because MAE is not reduced. Similarly, we also conclude that the proposed adaptive consistency weight does not improve Mean Teacher in semi-supervised crowd counting, because it only achieves slightly better MAE than using the standard sigmoid ramp-up function (i.e. adaptive vs standard = 14.7 vs 14.8), with slighly worse RMSE (i.e. adaptive vs standard = 23.9 vs 23.5).

Beyond semi-supervised learning, we also show that using batch normalization effectively improves the performance of a model trained on less labeled data – using 10% labeled data in crowd counting as an example, as shown in Figure 4.12.

In summary, the proposed adaptive consistency weight is a more effec-

tive and more convenient (i.e. without any hyper-parameter) substitute of the standard ramp-up function in Mean Teacher, in the domain of aphid counting. We also conclude that both the standard and the adaptive Mean Teacher is not able to generalize well to semi-supervised crowd counting – which may be due to the high complexity of the dataset.

However, we aim to develop a more generalizable and more robust method that is also capable of addressing the semi-supervised crowd counting task. This may also show the potential of the approach to be effectively adapted to other tasks with more complex settings, for example heavier occlusion, noisier background, and more objects, in plant phenotyping. Thus, we will explore a more up-to-date semi-supervised learning method – FixMatch, and adapt it to heatmap-regression based counting tasks, starting with crowd counting.

Chapter 5

Semi-supervised Counting with Monte-Carlo-Dropout FixMatch and Shadowout

5.1 Introduction

In last chapter, we propose the adaptive Mean Teacher, which significantly improves aphid counting in the 10% labeled data setting. However, it does not achieve satisfactory results on semi-supervised crowd counting, where only the RMSE is improved. Thus, in this chapter, we aim to develop a more robust approach that is capable to address semi-supervised heatmap-regression based counting in both domains.

Mean Teacher ([140], proposed in 2017) discussed in last chapter is a consistency-regularization approach, which forces the predictions on the slightly perturbed versions of the same input to be consistent. Beside consistency regularization, pseudo-labeling is another trending method in semi-supervised learning, which transfers the model predictions to pseudo labels to supervise the learning model. Therefore, in this chapter, we explore a more up-to-date semi-supervised learning method, FixMatch [132] (proposed in 2020), which uses stronger data augmentation (than Mean Teacher) and combines consistency regularization with pseudo-labeling.

FixMatch has demonstrated SOTA (or close to SOTA) performance in many generic classification datasets including CIFAR-10/100 [70], SVHN [99], STL-10 [23] and ImageNet [30]. The idea of FixMatch is simple – for an unlabeled image, first generating a pseudo-label from the model prediction on its weakly-augmented version, and then enforcing the consistency between the pseudo-label and the model prediction on its strongly-augmented version. As FixMatch is proposed for classification, the pseudo-label is one-hot label transferred from the model prediction, only when the greatest predicted probability is greater than a preset threshold. This prevents performance decreasing from unreliable predictions.

Different from image classification, where the prediction confidence can be represented by the greatest predicted probabilities, the prediction in heatmap regression is nothing but a single image, where the confidence is difficult to estimate without a reference. Hence, in this chapter, we propose to use Monte Carlo Dropout to estimate the uncertainty of the predicted heatmap; the low-uncertainty predictions will be transferred to pseudo-labels. Monte Carlo Dropout computes the mean and variance of the prediction to represent its uncertainty, which is realized by turning on dropout at model evaluation and forward the input multiple times. In addition, a data augmentation technique, Shadowout, which randomly changes the brightness of multiple random areas of an image, is proposed specifically for heatmap regression tasks. The experiments on both crowd counting and aphid counting demonstrate the effectiveness of the proposed methods.

The contributions of this chapter are:

1. adapting FixMatch to heatmap regression using Monte Carlo Dropout;
2. introducing and evaluating a heatmap-regression-specific data augmentation method – Shadowout, on crowd counting and aphid counting.

5.2 Related Work

5.2.1 FixMatch

FixMatch [132] is a semi-supervised learning method integrating pseudo-labeling with consistency regularization. In FixMatch, the consistency is built between the predictions of the *weakly-augmented* version and the

strongly-augmented counterpart of the same input. Here, weak augmentation (e.g. vertical and horizontal flip) usually refers to the image transformation that does not significantly affect the image quality or shift the image. Strong augmentation, however, may dramatically change the color (e.g. altering brightness or hue, or even inverting the color) or block some areas (e.g. Cutout [32]) of the image. *RandAugment* [27] and *Cutout* [32] are used simultaneously in FixMatch as strong augmentation.

RandAugment samples a number of different strong transformations (e.g. altering brightness or hue, inverting the color, etc.) and applies them to the input with various intensities. A list of image transformations from RandAugment and some visual examples of each transformation used in this chapter will be shown in the experiment section in Table 5.1 and Figure 5.3, respectively.

Cutout (visual examples are shown later in Figure 5.2) is implemented after RandAugment, which directly blocks a random area of the input. Cutout is generally good for image classification because it prohibits the learning model from overfitting some patterns or parts of the targets, by covering different areas of the images during training. However, it may cause inconsistency between the input and the desired output in heatmap regression with multiple small objects. The limitation of Cutout will be further discussed in 5.2.2.

As shown in Figure 5.1, the prediction of the weakly-augmented version is converted to an one-hot pseudo-label when it is considered as above confidence, i.e. the greatest predicted probability $\geq$ the preset threshold (the dotted line line at the prediction on the top row in Figure 5.1). This pseudo-label is then used to supervise the output of the strongly-augmented version of the same input. In image classification tasks, the confidence of

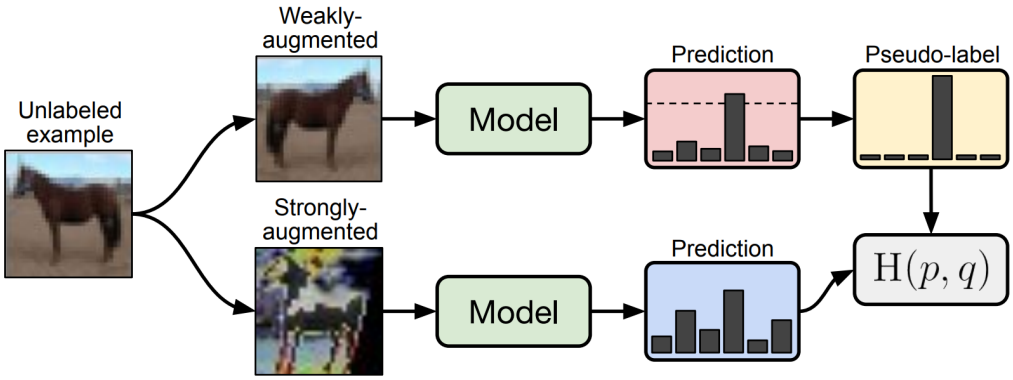


Figure 5.1: FixMatch algorithm on image classification. For any unlabeled sample, a weakly-augmented version is predicted by the model (top). Only when the greatest value among the assigned probabilities exceeds the threshold (dotted line), the prediction is transferred to a one-hot pseudo-label. With a cross-entropy loss ($H(p, q)$), this pseudo-label is used to supervise the model trained on the strongly-augmented version of the same example (bottom). Image source: [132].

the prediction can be implemented by thresholding the predicted probabilities, for example, if any of the predicted probabilities among all the categories is greater than 0.6, this prediction is deemed as confident; the acquisition functions in active learning (discussed in Chapter 3) can also be used to estimate the confidence as well. This confidence estimation, however, is biased to the model – the model can be misled during training and predict incorrect labels with high confidence.

Same as Mean Teacher, the final loss function L in FixMatch also consists of the supervision loss L_{sup} and the consistency loss L_{cons}, defined as:

$$L = L_{sup} + \lambda L_{cons}, \tag{5.1}$$

where λ is a weight to balance the losses. For labeled data, the supervision loss is expressed as:

$$L_{sup} = \mathbb{E}\left[H(p, p(y|\alpha(x)))\right], \tag{5.2}$$

where p is the true label, $p(y|\alpha(x))$ is the prediction on a labeled sample x with a weak augmentation $\alpha(\cdot)$, and $H(\cdot)$ represents the cross-entropy operation in classification task. Setting $q = p(y|\alpha(x))$, which is the output on the weak augmentation input, for the same input x, the consistency loss L_{cons} is defined as:

$$L_{cons} = \mathbb{E}\left[\mathbb{1}(\max(q) \geq \tau) H(\hat{q}, q(y|A(x)))\right], \tag{5.3}$$

where τ is the confidence threshold, $\hat{q} = arg\,max(q)$, representing the one-hot pseudo-label transferred from q, and $q(y|A(x))$ is the output on x with stronger augmentation $A(\cdot)$.

Although pseudo-labeling is designed for unlabeled data, in practice, in addition to being supervised by the ground-truth label, the labeled samples are also considered as part of the unlabeled samples and involved in the process of pseudo-labeling and the calculation of consistency loss. Hence, for labeled data, both L_{sup} and L_{cons} are calculated, while for unlabeled data, only L_{cons} is considered.

Although the idea of consistency regularization is key to FixMatch and Mean Teacher, they are distinct from each other:

- FixMatch introduces different levels of transformation for the input (i.e. strong and weak augmentation), while Mean Teacher usually uses weaker transformation. The application of the stronger augmentation pushes the learning model to be more robust to predicting consistently on the same input.

- FixMatch includes a confidence-filter and pseudo-label process before consistency regularization, while Mean Teacher does not. This is where FixMatch combines pseudo-labeling with consistency regular-

ization. Only the predictions with high confidence can be transferred to pseudo-label (e.g. one-hot label) and to supervise the learning model, preventing the inaccurate pseudo-labels corrupting the learning model.

- FixMatch trains a single model, while Mean Teacher utilizes a teacher model, which is an exponential moving average (EMA) of the student model, to supervise the student model.

5.2.2 Limitations of FixMatch

The key contributions of FixMatch are (1) improving pseudo-labels by thresholding the prediction confidence, and (2) using stronger augmentation such as Cutout for better regularization. However, for heatmap regression task, for example, the crowd and aphid counting tasks in this chapter, these key components of FixMatch become invalid to some extent.

First, unlike image classification, it is difficult to estimate the confidence from a single prediction on heatmap regression. Second, Cutout benefits image classification because it prevents the model from overfitting a certain feature of a class by randomly blocks an area of the image, but it is ill-posed for counting tasks because it causes inconsistency between the input and the expected density map. For example, if Cutout blocks 5 aphids in an image containing 12 aphids, then the expected heatmap should just sum to 7 rather than 12, but without the ground truth (e.g. for unlabeled images), we can never be sure of the real count of the image.

Therefore, here we propose to estimate the uncertainty of the predicted heatmaps using Monte Carlo dropout to address the former issue, and a new data augmentation method, which we term Shadowout, to accommodate

the latter problem.

5.3 Method

5.3.1 Shadowout

As introduced before, Cutout [32], which is adopted in the initial proposal of FixMatch [132], may be inappropriate for counting because its blocking mechanism leads to inconsistent number of counts between the input and the desired output. Here we propose Shadowout as a strong data augmentation to be used in heatmap-regression based counting tasks.

The motivation behind this is as follows. In plant phenotyping, especially for in-field research, shading from the background (e.g. clouds or trees) and occlusion between the research objects (e.g. leaves or aphids) often appear in multiple areas of the image. Inspired by this, we propose Shadowout to randomly change the brightness from 0 (i.e. totally black) to 1 (i.e. original brightness) of random regions on the image. Specifically, for an image, Shadowout samples a random number $n \in [0, N]$ of square areas of random size $s \in [S_1, S_2]$, before altering the brightness in each square with the intensity uniformly sampled from 0 to 1.

Figure 5.2 shows some examples to illustrate the difference between Cutout (2nd row), Random Cutout (3rd row) and Shadowout (4th row). Cutout usually blocks a single large region (e.g. quarter of an image), but the proposed Shadowout randomly tunes the brightness on multiple small regions with different size. Random Cutout is same as Shadowout except the pixel values at each random regions are set to zero. These three methods will be compared with each other in the aphid counting experiments below.

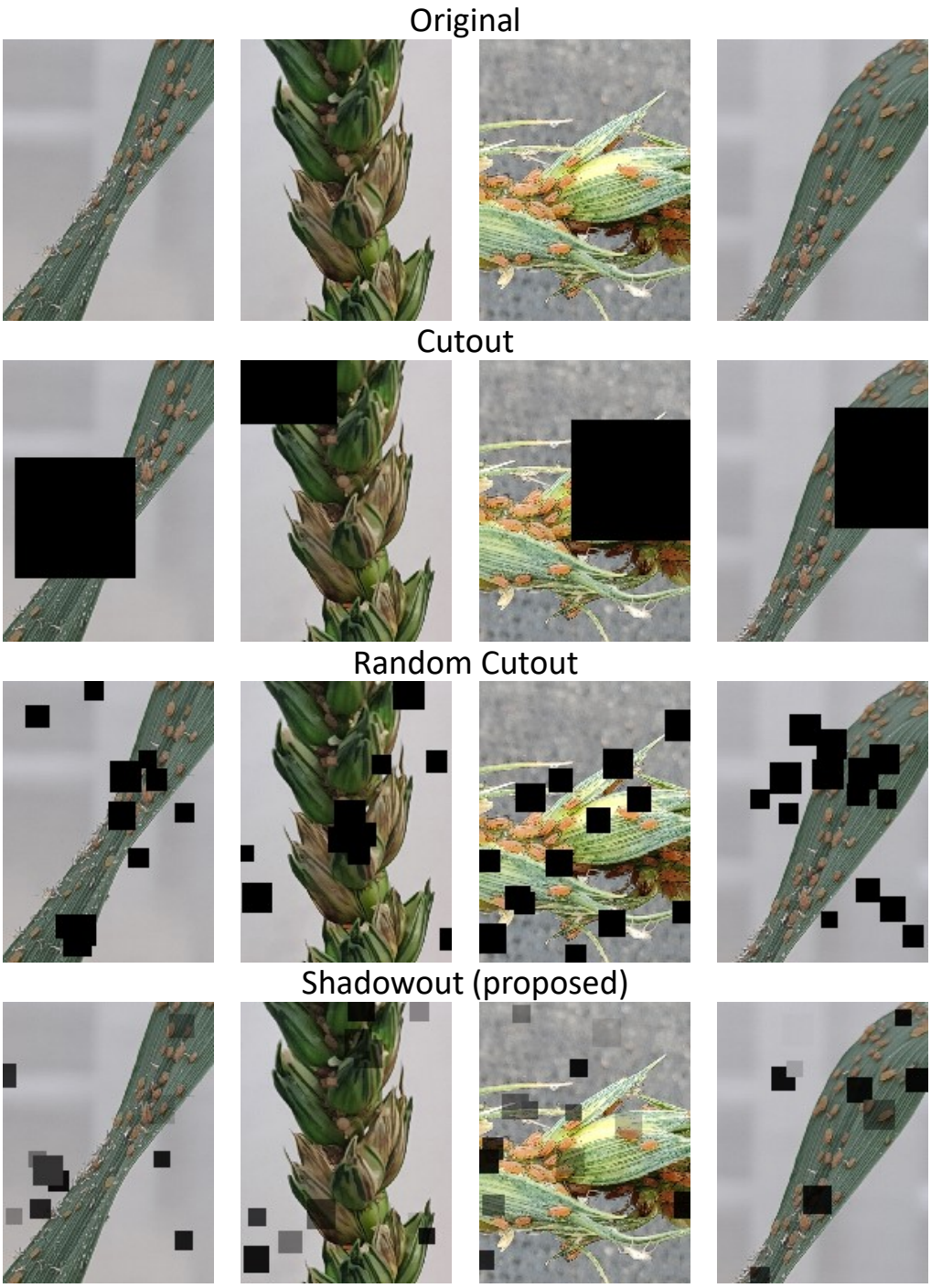


Figure 5.2: Different occlusion-based image transformation methods applied on four aphid images. The original images are shown at the top row, and the effects of applying different transformations to these images are shown at the remaining rows, respectively.

Shadowout is different from the typical implementation of Cutout in two aspects: (1) Shadowout acts to more than one area with small size, rather than a single large square; (2) Shadowout randomly changes the brightness in each square, instead of blacking-out the selected region. These two traits are designed to fit the density-estimation based counting tasks – they usually include many (small) targets to predict. Therefore, covering more small regions without affecting the consistency between the input and output, is beneficial.

5.3.2 FixMatch with Monte Carlo Dropout

Monte Carlo Dropout is proposed in [40] to estimate the uncertainty of deep learning models. The idea is to turn on dropout at the evaluation stage and forward the same input N times to have N predictions. To note, the dropout and batch normalization are supposed to be switched off in evaluation stage, but here only dropout is turned on so different features are dropped at each time to create multiple sub-models. The mean μ and variance σ^2 are calculated from the predictions to represent the uncertainty of the model on the input.

Inspired by the original version of FixMatch, which selects the confident sample by thresholding its highest predicted probability and transfers its prediction to one-hot pseudo-label, here we propose to use the variance σ^2 and mean μ from Monte Carlo Dropout as the uncertainty indicator and pseudo-label respectively to adapt FixMatch to density estimation tasks. Specifically, at the beginning of each training iteration, for each image x, the mean map $\mu_{\alpha(x)}$ and variance map $\sigma^2_{\alpha(x)}$ of its weakly-augmented version $\alpha(x)$ is predicted from the current model using Monte Carlo Dropout; if the sum of all the pixel values on $\sigma^2_{\alpha(x)}$ is less than a pre-defined threshold

τ, $\mu_{\alpha(x)}$ will be used as the pseudo-label to form the prediction consistency between $\alpha(x)$ and the corresponding strongly-augmented version $A(x)$; otherwise, the consistency loss is not calculated.

The final loss function is defined as $L = L_{sup} + \lambda L_{cons}$, same as in Equation 5.1, where λ is a hyper-parameter to balance the losses. In heatmap regression task, mean square error (MSE) is usually used as loss function so here MSE between heatmaps is adopted in both supervision loss and consistency loss, same as the loss functions for Mean Teacher implemented in Chapter 4. The supervision term L_{sup} (for labeled data only) is defined as:

$$L_{sup} = \mathbb{E}\left[|y - M(\alpha(x), \theta))|^2\right], \tag{5.4}$$

where y represents the true density map for x and $M(\cdot, \theta)$ is the learning model. The consistency term L_{cons} (for both labeled and unlabeled data) is defined as:

$$L_{cons} = \mathbb{E}\left[\mathbb{1}(\sigma^2_{\alpha(x)} \leq \tau)\left|\mu_{\alpha(x)} - M(A(x), \theta)\right|^2\right] \tag{5.5}$$

5.4 Experiments

To note, below we term the FixMatch with Monte Carlo Dropout as *MC-FixMatch*, to differ from the original version of FixMatch proposed in [132]. By default, *Shadowout* is used in MC-FixMatch.

In this section, we evaluate the proposed MC-FixMatch on semi-supervised crowd counting and aphid counting, respectively. The experiments begin with crowd counting because Mean Teacher does not achieved satisfying results on crowd counting as shown Chapter 4.

For both crowd counting and aphid counting, we compare training MC-FixMatch on 10% labeled data, with the other training schemes, namely label-only, standard Mean Teacher (using the sigmoid ramp-up function), and adaptive Mean Teacher, which have been evaluated in Chapter 4. We expect that the proposed MC-FixMatch should at least improve the label-only scenario for both crowd counting and aphid counting, to demonstrate its effectiveness and robustness. In addition, we also explore MC-FixMatch with different values of the uncertainty threshold τ. Moreover, we compare the performance of MC-FixMatch using Shadowout with using its substitutes such as Cutout and Random Cutout.

5.4.1 Implementation Details

5.4.1.1 MC-FixMatch and Shadowout

CSRNet (one output head) with batch normalization at its backend is adopted as the base model of MC-FixMatch. A dropout layer is added after the frontend of CSRNet to perform Monte Carlo Dropout.

Monte Carlo Dropout is performed at the beginning of each training iteration with the number of forwarding times set to 10. Multiple values of the uncertainty threshold τ (i.e. 0.01, 0.1, 1, ∞) are first evaluated on crowd counting, and then we empirically set $\tau = 0.05$ in aphid counting by examining the number of selected pseudo-labels per training iteration.

Weak augmentation is not applied (i.e. directly using input x as weakly-augmented version $\alpha(x)$) because it often contains flip operation, which causes inconsistency between the pseudo-label and the prediction of the strongly-augmented input $A(x)$. RandAugment following by Shadowout is used as strong augmentation. The operations that cause shifting in

RandAugment are removed for the same reason of not using weak augmentation. The details of our implementation of RandAugment is described in Table 5.1 and some visual examples of each transformation is shown in Figure 5.3.

For Shadowout (visual examples shown above at the bottom row of Figure 5.2), the number of regions n is sampled from $[0, 16]$, and the size of each region is sampled from $[30, 60]$.

5.4.1.2 Training and Validation

For a fair comparison with the other training schemes, the proposed MC-FixMatch is trained and evaluated on the exactly same datasets in Chapter 4 – that is, we use the same training data and validation data of ShanghaiTech part B (ShB) and the aphid dataset as Chapter 4. Specifically, for crowd counting, we train MC-FixMatch 5 times on the *same* split (i.e. the identical split used in Chapter 4) of 10% labeled data from ShB; for aphid counting, we train MC-FixMatch on the 5 *different* splits (i.e. the identical splits used in Chapter 4) of 10% labeled data from the aphid dataset, respectively. We report the average best MAE and RMSE in validation over 5 runs.

The hyper-parameters for training MC-FixMatch on crowd counting and aphid counting are very similar in this chapter. Specifically, the original training images are first randomly cropped to a quarter of the original size (i.e. for ShB, this is from 768x1024 to 384x512; for aphid, this is from 1024x768 to 512x384) and horizontally flipped with 0.2 probability. This flip and crop aim to expand the training dataset, which is not relevant to the data augmentation used in MC-FixMatch. For Mc-FixMatch, the strong augmentation (details in Table 5.1) followed by the proposed Shadowout

Table 5.1: List of image transformations from RandAugment used in our implementation. Rotating, shearing and translating are removed because they cause displacement. Every time, $N = 2$ transformations are randomly selected from the list, and each transformation has a 50% chance to be applied to the image with the intensity (if applicable) randomly drew from the associated parameter range.

Transformation	Description	Parameter	Range
Autocontrast	Maximizes the image contrast by setting the darkest (lightest) pixel to black (white).	-	-
Brightness	Adjusts the brightness of the image. $B = 0$ returns a black image, $B = 1$ returns the original image.	B	$[0.05, 0.95]$
Color	Adjusts the color balance of the image like in a TV. $C = 0$ returns a black and white image, $C = 1$ returns the original image.	C	$[0.05, 0.95]$
Contrast	Controls the contrast of the image. $C = 0$ returns a gray image, $C = 1$ returns the original image	C	$[0.05, 0.95]$
Equalize	Equalizes the image histogram.	-	-
Identity	Returns the original image.	-	-
Posterize	Reduces each pixel to B bits.	B	$[4, 8]$
Sharpness	Adjusts the sharpness of the image, where $S = 0$ returns a blurred image, and $S = 1$ returns the original image.	S	$[0.05, 0.95]$
Solarize	Inverts all pixels above a threshold value of T.	T	$[0, 255]$

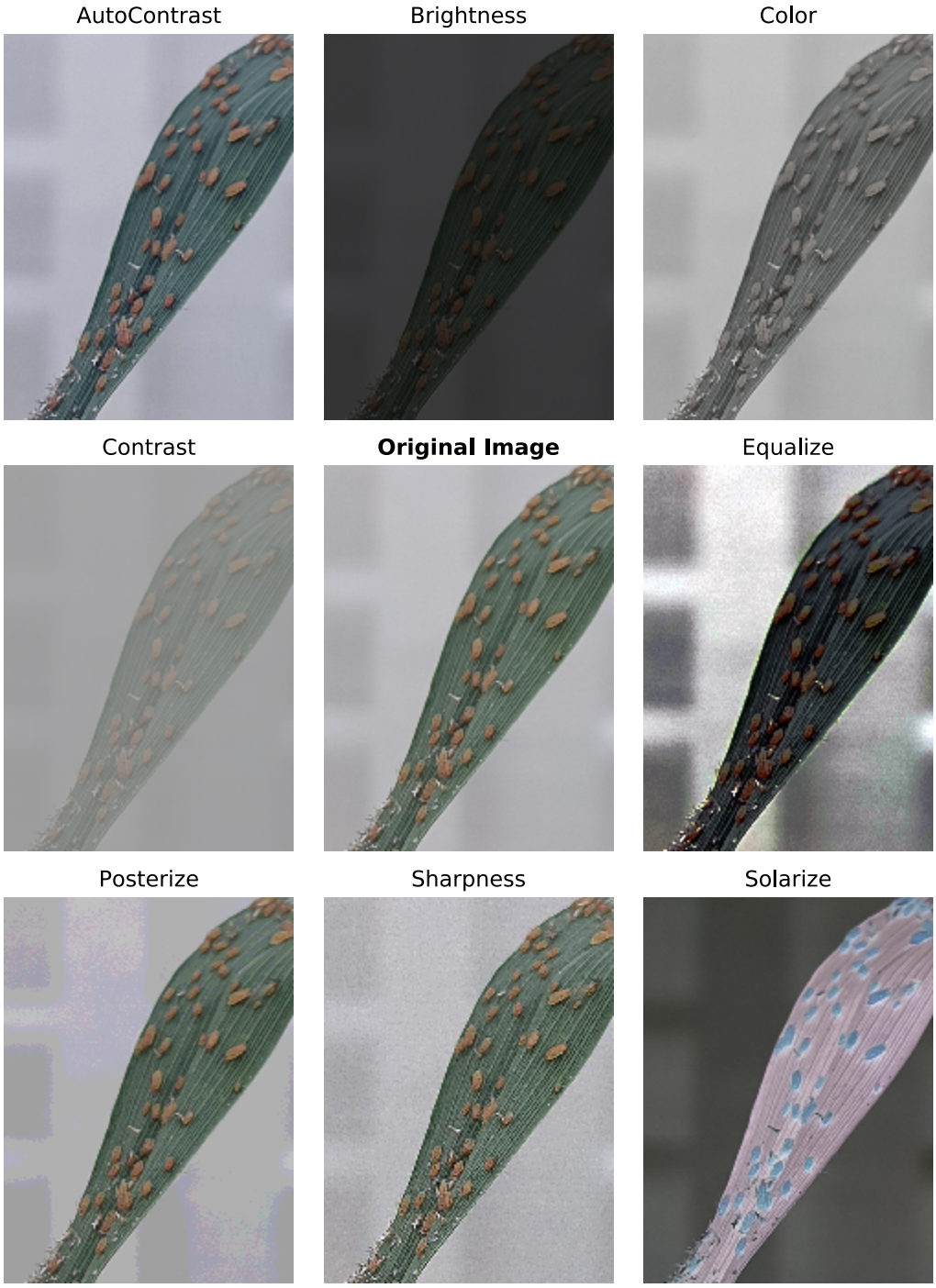


Figure 5.3: Visual examples of every transformation in RandAugment used in our experiments (details described in 5.1). The original input image is shown in the middle.

is applied after the crop and flip, to create the strongly-augmented input. The batch size is set to 16, consisting of 8 labeled images and 8 unlabeled image in every batch. An Adam optimizer with a learning rate of 10^{-5} and weight decay of 5×10^{-4} is used to optimize MC-FixMatch. For crowd counting only, we empirically divide the learning rate by 10 at the 400th training epoch.

5.4.2 Crowd Counting

5.4.2.1 MC-FixMatch + Shadowout vs Other Training Schemes

We first compare the performance of the proposed MC-FixMatch + Shadowout, with the other training schemes that have been evaluated in Chapter 4, using 10% labeled (and the other 90% unlabeled if applicable) training data of ShB. Figure 5.4 shows the validation errors of these training schemes:

- Label-only (evaluated in Chapter 4): directly training the CSRNet on 10% labeled data only;
- MT (evaluated in Chapter 4): standard Mean Teacher with ramp-up epoch = 5, EMA decay $\alpha = 0.999$ and consistency weight $\lambda = 1$;
- Adaptive MT (proposed and evaluated in Chapter 4): adaptive Mean Teacher with EMA decay $\alpha = 0.999$ and consistency weight $\lambda = 1$;
- MC-FixMatch + Shadowout (proposed in this chapter): FixMatch using Monte Carlo Dropout plus Shadowout; we empirically set the uncertainty threshold $\tau = 1$.

As shown in Figure 5.4, compared to label-only, the proposed method

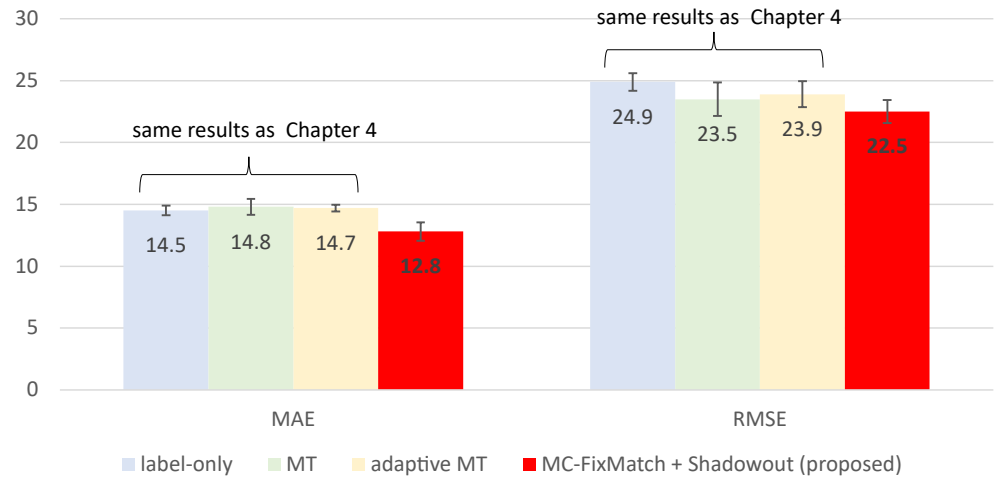


Figure 5.4: Validation MAE (left) and RMSE (right) of crowd counting on 10% labeled training data using different training schemes. Each pair of MAE and RMSE is reported in mean (i.e. the value associated with the bar) and standard deviation (i.e. the error bar) from 5 runs of training on the same 10% labeled data (i.e. same data as in Chapter 4).

– MC-FixMatch + Shadowout reduces the MAE by 11.7% (i.e. from 14.5 to 12.8) and the RMSE by 9.6% (i.e. from 24.9 to 22.5). Besides, the performance of MC-FixMatch + Shadowout consistently outperforms the standard and the adaptive Mean Teacher approaches in terms of both MAE and RMSE. We conclude that MC-FixMatch + Shadowout is capable to improve semi-supervised crowd counting, on 10% labeled ShB.

We also conduct a two-tailed unpaired (assuming unequal variance) Student's t-test (same as Chapter 4) to verify if there is any significant improvement after applying MC-FixMatch + Shadowout. The t-test results are presented in Table 5.2. The results from Table 5.2 again demonstrate the superiority of MC-FixMatch + Shadowout – the improvement in both MAE and RMSE is significant (at 0.01 level) compared to label-only. Recalling Table 4.7 in Chapter 4, which shows the t-test results between label-only and the standard/adaptive *Mean Teacher*, there is no significant (at 0.05 level) improvement after applying Mean Teacher in terms of both MAE and RMSE.

Table 5.2: P-values of the two-tailed unpaired unequal variance student's t-test for crowd counting, verifying the results shown in Figure 5.4. Here we use FM to represent MC-FixMatch + Shadowout.

	p-value (MAE)	p-value (RMSE)
label-only vs FM	**0.005**	**0.002**
MT (ramp=5) vs FM	**0.002**	0.232
MT (adaptive) vs FM	**0.004**	0.069

In addition, the training time of the explored semi-supervised learning methods are shown in Table 5.3.

Table 5.3: Training time on crowd counting of Mean Teacher (ramp=5), adaptive Mean Teacher, and MC-FixMatch + Shadowout. The training time is calculated by $n \times t$, where n is the index of the iteration that the model converges and t is the time (seconds) needed to train the model per iteration. The results are averaged from 5 runs and normalized by the training time of Mean Teacher.

	MT	adaptive MT	MC-FixMatch
training time	1	0.94	3.29

5.4.2.2 Changing the Uncertainty Threshold τ

The uncertainty threshold τ is a key hyper-parameter of the proposed MC-FixMatch. It is assumed to affect the performance of the learning model – it should not be too small (i.e. filtering most samples) or too large (i.e. allowing all the samples). Below the effect of setting different values of τ is evaluated in crowd counting, which is a more studied domain than aphid counting, as we wish to consider MC-FixMatch a more generic model for heatmap regression and not limited only to plant phenotyping.

Figure 5.5 shows the number of selected pseudo-labels per training iteration under different values of τ. In all the reported values of τ, the number of selected pseudo-labels increases (until reaching the batch size 16) as training proceeds, indicating that the learning model becomes more

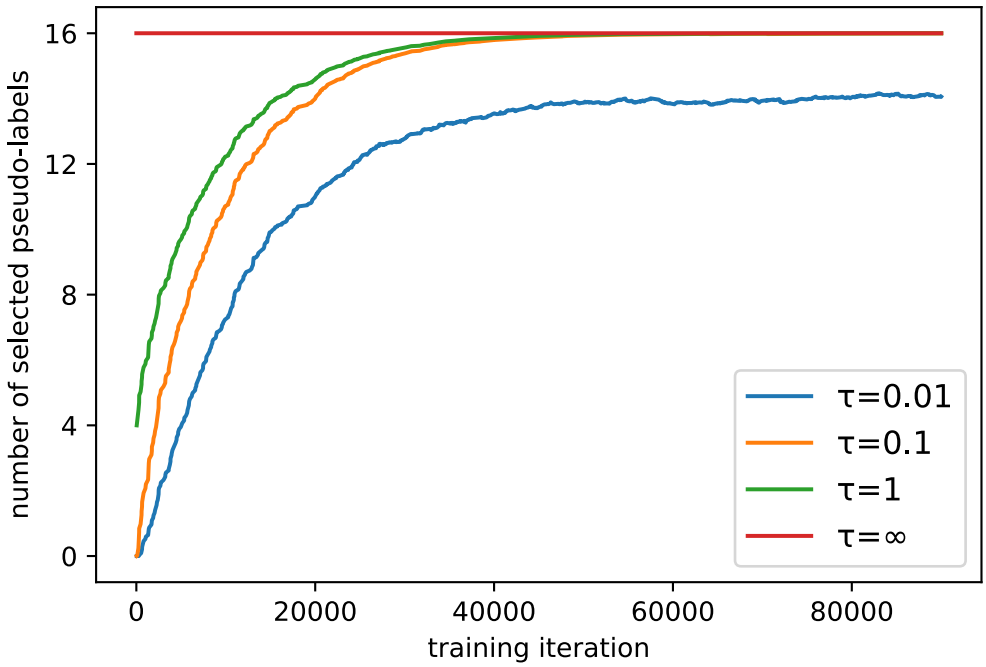


Figure 5.5: The number of selected pseudo-labels per training iteration on 10% labeled ShB, using different uncertainty thresholds τ.

and more reliable. As expected, when the criterion of uncertainty is too strict (i.e. $\tau = 0.01$), many predictions are deemed as uncertain even at later training stages, and hence the number of selected pseudo-labels is always lower than the batch size. When the criterion of uncertainty is too loose (i.e. $\tau = \infty$), all the pseudo-labels are allowed to be used from the very beginning of the training process. Only when the criterion of the uncertainty is properly set (i.e. $\tau = 0.1$ or 1), the number of the selected pseudo-labels gradually increases until reaching the batch size. Below we evaluate MC-FixMatch with different uncertainty thresholds.

Figure 5.6 compares the validation MAE and RMSE of using different τ. As expected, $\tau = 0.01$ achieves the worst performance, which is even worse than training on labeled data only, because many unlabeled samples are dumped without providing any information. $\tau = 0.1$ and $\tau = 1$ achieve comparable results, although $\tau = 1$ outperforms $\tau = 0.1$ in both metrics. Surprisingly, $\tau = \infty$ does not damage the performance too much

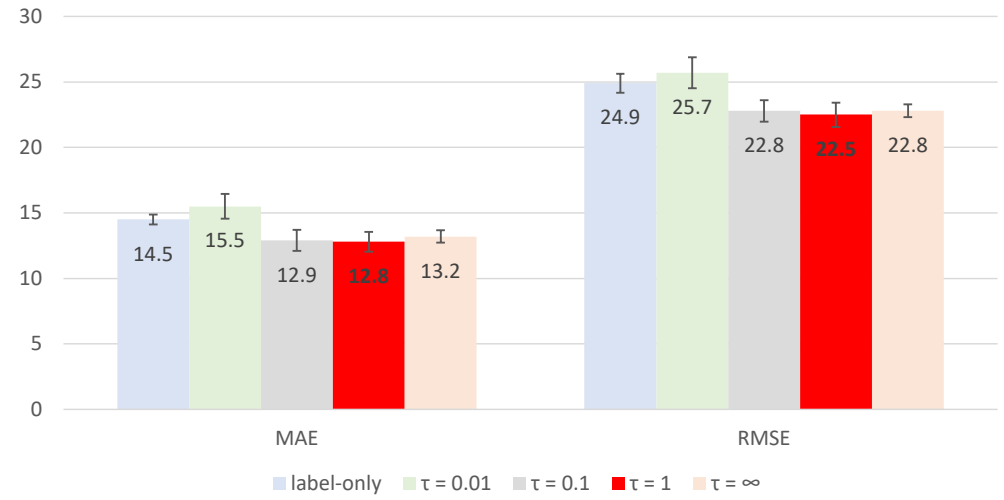


Figure 5.6: Validation MAE and RMSE of MC-FixMatch (+ Shadowout) varying the uncertainty threshold τ, trained on 10% labeled data of ShB. The label-only bar is for the convenience of comparison. The results are reported as averaging 5 runs on the same split of data.

– it achieves the worse MAE but the same RMSE as $\tau = 0.1$; this may be because the pseudo-label is essentially the mean of multiple predictions, which counteracts the noise and is relatively accurate.

We conduct the two-tailed unpaired (assuming unequal variance) Student's t-test for the results in Figure 5.6. The t-test results shown in Table 5.4 confirms our finding – the performance of using very a small uncertainty threshold (e.g. $\tau = 0.01$) is significantly worse than the other evaluated values at 0.01 level (1st to 3rd rows), but the performance gap between $\tau = 0.1$, 1 and ∞ is not significant (4th to 6th rows).

Table 5.4: Student's t-test results for Figure 5.6, verifying the significance of the performance change when applying different τ to MC-FixMatch on crowd counting.

τ	p-value (MAE)	p-value (RMSE)
0.01 vs 0.1	**0.002**	**0.003**
0.01 vs 1	**0.001**	**0.002**
0.01 vs inf	**0.003**	**0.003**
0.1 vs 1	0.912	0.676
0.1 vs inf	0.485	0.989
1 vs inf	0.395	0.620

5.4.3 Aphid Counting

5.4.3.1 MC-FixMatch + Shadowout vs Other Training Schemes

Similar to the crowd counting experiments, we also compare the proposed MC-FixMatch + Shadowout with the other training schemes, on 10% labeled aphid dataset. Figure 5.4 shows the validation errors of the different training schemes:

- Label-only (evaluated in Chapter 4): directly training the CSRNet on 10% labeled data only;
- MT (evaluated in Chapter 4): standard Mean Teacher with ramp-up epoch = 5, EMA decay $\alpha = 0.999$ and consistency weight $\lambda = 1$;
- Adaptive MT (proposed and evaluated in Chapter 4): adaptive Mean Teacher with EMA decay $\alpha = 0.999$ and consistency weight $\lambda = 1$;
- MC-FixMatch + Shadowout (proposed in this chapter): FixMatch using Monte Carlo Dropout plus Shadowout; we empirically set the uncertainty threshold $\tau = 0.05$.

As shown in Figure 5.7, the proposed MC-FixMatch + Shadowout improves on the label-only baseline by 13% in MAE (i.e. from 4.6 to 4) and 11.3% in RMSE (i.e. 7.1 to 6.3). It is also better than the standard Mean Teacher with sigmoid ramp-up, where the MAE and RMSE is reduced by 7% (i.e. MAE reduced from 4.3 to 4) and 11.3% (i.e. RMSE reduced from 7.1 to 6.3), respectively. However, it is worse than the adaptive Mean Teacher in terms of MAE.

The two-tailed paired Student's t-test is conducted to verify the statistical significance of the semi-supervised aphid counting results shown in

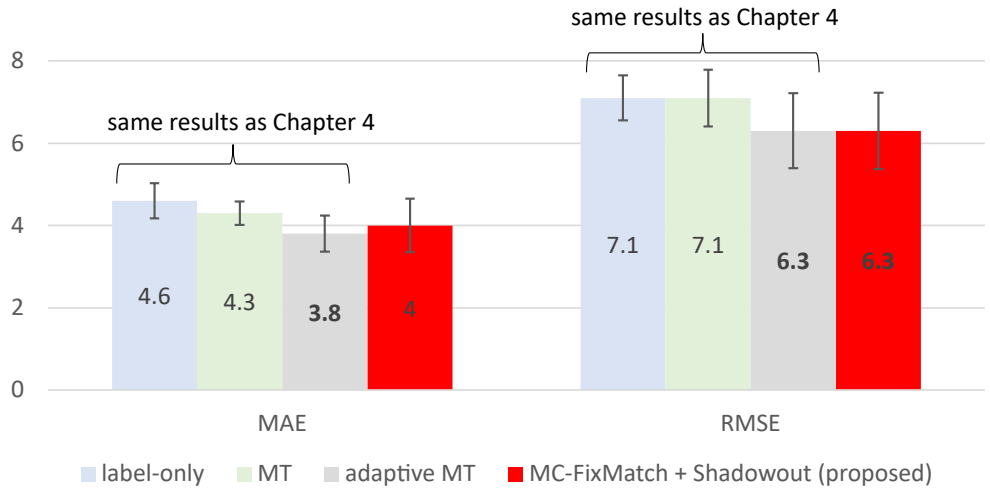


Figure 5.7: Validation MAE (left) and RMSE (right) of aphid counting on 10% labeled training data using different training schemes. Each pair of MAE and RMSE is reported in mean (i.e. the value associated with the bar) and standard deviation (i.e. the error bar) from training on 5 different splits of 10% labeled data (i.e. identical splits as in the aphid counting experiments of Chapter 4).

Figure 5.7. The t-test results presented in Table 5.5 indicates the statistical significance (at 0.05 level) of the performance improvement after using MC-FixMatch + Shadowout, compared to label-only (in both MAE and RMSE, 1st row) and the standard Mean Teacher (only in RMSE, 2nd row), although there is no significance compared to the adaptive Mean Teacher proposed in Chapter 4.

As mentioned at the end of Chapter 4 and the beginning of this chapter, we aim to develop a more *generalizable* approach that is capable of addressing both semi-supervised aphid counting and crowd counting. Compared to the adaptive Mean Teacher, which only reaches satisfactory results on aphid counting, the proposed MC-FixMatch + Shadowout in this chapter shows the effectiveness in both domains, and hence our goal is satisfied.

In addition, the training time of the explored semi-supervised learning methods are shown in Table 5.6.

Table 5.5: P-values of the two-tailed paired Student's t-test results for Figure 5.7, comparing FM (i.e. MC-FixMatch + Shadowout) with the other training schemes on semi-supervised aphid counting.

	p-value (MAE)	p-value (RMSE)
label-only vs FM	**0.009**	**0.025**
MT (ramp=5) vs FM	0.173	**0.016**
MT (adaptive) vs FM	0.437	0.741

Table 5.6: Training time on aphid counting of Mean Teacher (ramp=5), adaptive Mean Teacher, and MC-FixMatch + Shadowout. The training time is calculated by $n \times t$, where n is the index of the iteration that the model converges and t is the time (seconds) needed to train the model per iteration. The results are averaged from 5 runs and normalized by the training time of Mean Teacher.

	MT	adaptive MT	MC-FixMatch
training time	1	0.46	1.91

5.4.3.2 Ablation Study on Shadowout

We have already demonstrated the effectiveness and robustness of the combination of MC-FixMatch and Shadowout, in semi-supervised crowd counting and aphid counting. We have also shown how different uncertainty thresholds τ affect MC-FixMatch in the more generic crowd counting. Here, the proposed Shadowout is compared with its alternatives on 10% labeled aphid counting, because it is inspired by natural shadows (e.g. from leaves or trees) and occlusion, common in plant phenotyping scenarios.

In addition to Cutout, which is applied to the original FixMatch and seems to be more suitable for classification than heatmap regression, we also compare Shadowout with not using any occlusion-based transformation, and Random Cutout, which is a combination of Cutout and Shadowout. The visual examples of these occlusion-based data augmentation are shown in Figure 5.2, and the hyper-parameters of each transformation are shown below:

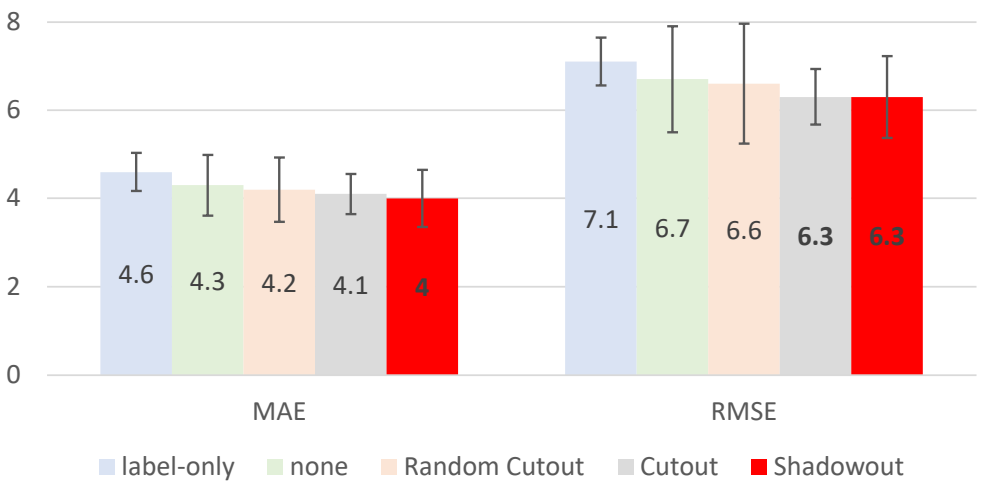


Figure 5.8: Validation MAE (left) and RMSE (right) of using different occlusion methods with MC-FixMatch. Random crop, random flip and RandAugment are still applied before the evaluated occlusion-based transformation. Each pair of MAE and RMSE is reported in mean (i.e. the value associated with the bar) and standard deviation (i.e. the error bar) from training on 5 different splits of 10% labeled aphid data (i.e. identical splits as Chapter 4).

- None: not using any occlusion augmentation;
- Cutout: randomly block a 220×220-size (approximately a quarter of an image) region;
- Random Cutout: randomly block n regions ($n \in [0, 16]$) with $s \times s$ resolution ($s \in [30, 60]$);
- Shadowout (proposed): randomly change the brightness (from 0 to 1) of n regions ($n \in [0, 16]$) with $s \times s$ resolution ($s \in [30, 60]$).

To note, following the same setting as the original paper of FixMatch [132], Cutout occludes the selected regions in grey (i.e. pixel value is set to $(127, 127, 127)$ for RGB images) rather than black (i.e. pixel value=$(0, 0, 0)$), while Random Cutout and Shadowout are set to block the selected areas in black.

Figure 5.8 compares the validation errors of using different occlusion

methods with MC-FixMatch. As shown in Figure 5.8, the proposed Shadowout consistently outperforms the others on both MAE and RMSE. However, the performance gap between Cutout and Shadowout is small – Shadowout achieves slightly better MAE and the same RMSE as Cutout. We assume Cutout is not optimal for heatmap regression, especially for the scenarios with many small objects, because it occludes a single large area of the input, which may cause inconsistency between the input and the output (e.g. different number of objects). However, the results in Figure 5.8 indicate that this inconsistency may actually be a tolerable noise to the heatmap regression model. It is also noticeable that using any occlusion augmentation listed at Figure 5.8 improves the performance as "none" achieves the worst MAE and RMSE.

Table 5.7: Student's t-test results for Figure 5.8, comparing the performance gap between different occlusion-based data augmentation on semi-supervised aphid counting.

	p-value (MAE)	p-value (RMSE)
None vs Random Cutout	0.886	0.861
None vs Cutout	0.497	0.343
None vs Shadowout	0.196	0.261
Random Cutout vs Cutout	0.606	0.577
Random Cutout vs Shadowout	0.287	0.646
Cutout vs Shadowout	0.545	0.914

The two-tailed paired Student's t-test is conducted to verify the statistical significance between the results shown in Figure 5.8. The t-test results, which are presented in Table 5.7, however, indicate that there is no significant difference between any of the evaluated transformations, despite the visible trend in favour of Shadowout.

5.5 Discussion

The experiments on aphid counting and crowd counting have demonstrate that combining FixMatch with Monte Carlo Dropout is able to perform and generalize well on semi-supervised learning heatmap regression tasks. As defined in [63], uncertainty in deep learning can be classified to epistemic uncertainty and aleatoric uncertainty. The former represents the uncertainty of the learning model, which can be estimated by Monte Carlo Dropout; the latter, capturing the uncertainty (e.g. noise) of the data (i.e. images and labels), however, is not properly represented in this chapter.

Aleatoric uncertainty in heatmap regression can be predicted by adding an extra output head to the learning model and integrating this uncertainty output into a MSE loss, which is defined as:

$$L_{alea}(\theta) = \frac{1}{D}\sum_{i}\frac{1}{2}\hat{\sigma}_i^{-2}||y_i - \hat{y}_i||^2 + \frac{1}{2}\log\hat{\sigma}_i^2, \tag{5.6}$$

where θ denotes the weights of the learning model, D is the pixel resolution of the output $\hat{y}_i$ (e.g. the predicted aphid density map of 64×48 resolution), y_i is the ground truth, and $\hat{\sigma}_i^2$ is the predicted variance from the additional output head. The second term $\frac{1}{2}\log\hat{\sigma}_i^2$ is to prohibit the model from predicting infinite variance (so that the loss can easily be small).

In practice, $s_i = \log\hat{\sigma}_i^2$ is predicted on the additional output head rather than $\hat{\sigma}_i^2$ because it is more numerically stable, so the loss function is further defined as:

$$L_{alea}(\theta) = \frac{1}{D}\sum_{i}\frac{1}{2}\exp(-s_i)||y_i - \hat{y}_i||^2 + \frac{1}{2}s_i \tag{5.7}$$

The predicted aleatoric uncertainty is sample-dependent and can be

added together to the uncertainty calculated from Monte Carlo Dropout as shown before to form a more complete uncertainty. This may estimate a more accurate uncertainty for thresholding and further improve the performance of FixMatch.

Another approach to integrate input uncertainty is to apply different data transformation on the input at each forward pass of Monte Carlo Dropout. For example, one can apply a weaker Shadowout which tunes the brightness level of the input from 0.6 to 1. This creates variations on both the input and the learning model (i.e. by applying dropout) so the variance generated from Monte Carlo Dropout is able to represent the uncertainty from both sides. Compared to computing the aleatoric uncertainty, this method does not require an additional output head, and the integrated uncertainty relates to the difficulty of the input image (e.g. background complexity and number of targets), which is also useful in semi-supervised learning.

5.6 Conclusion

In this chapter, we propose MC-FixMatch and Shadowout for semi-supervised heatmap regression. MC-FixMatch incorporates Monte Carlo Dropout with FixMatch to address the uncertainty estimation for heatmap regression. Shadowout, which is inspired by the natural shading and occlusion in plant phenotyping, is designed to randomly change the brightness of multiple random regions of an image. Shadowout can corrupt the input image without breaking the consistency between it and its desired output (e.g. the number of targets), which is specially useful for counting tasks.

The proposed approaches improve on both semi-supervised aphid count-

ing and crowd counting, with 10% labeled data. This demonstrates the effectiveness and robustness of the proposed methods in this chapter. As comparison, adaptive Mean Teacher proposed in Chapter 4 only achieves satisfying results on aphid counting. Here we have fulfilled the goal set at the end of Chapter 4 – to develop a more generalizable semi-supervised approach.

Chapter 6

Learning to Localise and Count With Incomplete Point-Annotations

6.1 Introduction

This chapter has led to a publication at the Workshop on Interactive Labeling and Data Augmentation for Vision (ILDAV) in the International Conference on Computer Vision (ICCV) 2021 (Section 1.4 2).

As discussed in the previous chapters, the quality and quantity of training data are important in deep learning, but annotating data is time-consuming and expensive. Active learning (discussed in Chapter 3) and semi-supervised learning (discussed in Chapter 4 and 5) attempt to reduce the labeling cost by using fewer labeled training samples. The difference is that active learning begins with a small labeled set, and gradually expand the labeled set by actively querying the labels of the most informative unlabeled samples from the human annotators within the training process; while semi-supervised learning uses a fixed labeled set and explores all of the available unlabeled data without extra manpower.

In recent years, another approach to reduce the labeling complexity – weakly-supervised learning, is becoming more and more popular in deep learning. Instead of decreasing the number of labeled samples, weakly-supervised learning aims to utilize "weaker" labels to train the deep neural networks, for instance, learning an object recognition task with image-level class labels rather than bounding boxes.

In counting and localisation tasks, people usually need to point-annotate every relevant object on images, which can be hundreds or thousands of points per image, which is costly. Moreover, human annotators may miss some objects due to fatigue and subjectivity, which leads to low-quality annotations. To address these two problems, a novel scenario – learning with fewer point-annotations *per image*, is explored in this chapter (some

example images will be shown in Figure 6.3). This is a scenario in between *weakly-supervised learning* and *semi-supervised learning*. On one hand, it explores incomplete labels per image; on the other hand, it also reduces the total number of annotations. This per-image reduction of annotations can be imposed intentionally by asking the annotators to label a portion (e.g. 50%) of what they see in each image to reduce the labeling efforts, while it can also be caused by unintentionally missing labels from the annotators. In this chapter, a simple yet effective approach – asymmetric mean square error (AMSE) is proposed to adapt to the intentional reduction and counteract the unintentional missing of per-image annotations, which are demonstrated on heatmap-regression based wheat spikelet localisation and crowd counting.

Mean square error (MSE) is widely used as a loss function in heatmap regression tasks, for example, object counting and localisation. As discussed in Chapter 4 and Chapter 5, a *heatmap* is usually generated by applying a Gaussian kernel on every point-annotation. However, if some of the point annotations are removed, a neural network trained with MSE will easily overfit the Gaussian areas around the remaining labeled points. Replacing MSE with the proposed loss function, AMSE, can force the learning model to enhance the response from the false-negative areas (i.e. areas where the targets are incorrectly unlabeled), and to cope with the class imbalance between background and targets when the annotations are significantly reduced.

In this chapter, experiments on two tasks, heatmap-regression-based crowd counting and wheat spikelet localisation are conducted to demonstrate the effectiveness and robustness of the proposed AMSE. Specifically, different percentages of the reduction of per-image annotations, which is termed the "drop rate", ranging from minor (10% drop) to extreme (90%

drop), are evaluated. The only hyper-parameter of AMSE (β) is tuned to adapt to different drop rates. Results show that an asymmetric loss function is effective across different models and datasets, even in very extreme cases with limited annotations provided (e.g. 90% reduction of the original annotations). Whilst tuning of the key parameter is required, setting conservative parameter values can help more realistic situations, where only small amounts of data have been missed by annotators.

The main contributions of this chapter are:

1. introducing a novel and efficient annotation concept, which helps account for accidental or purposeful partial annotation per image;
2. a description and evaluation of the proposed AMSE loss function to counteract performance loss from this reduced number of annotations per image;
3. an exploration of the sensitivity of the single tuning parameter β of AMSE over two diverse datasets and two deep neural network architectures;
4. demonstration of competitive performance on public crowd counting and wheat spikelet localisation datasets in the presence of substantially (from 10% to 90%) reduced annotations.

The results show that when the reduction of annotations per image is less than 50%, the proposed method can achieve comparable performance to the baseline (i.e. training on a fully-annotated dataset with MSE loss); when the reduction is greater than 70%, the proposed method perform better than the model trained on the same reduced dataset with MSE loss. However, selection of β and management of the training regime plays an important role.

6.2 Related Work

There are two scenarios of incomplete annotations discussed in this chapter – unintentionally missing annotations (e.g. by mistake or due to fatigue) and intentionally dropping annotations (e.g. as a faster labeling scheme). Although both tasks are accomplished by the proposed asymmetric loss function (with different hyper-parameters), the recent works are discussed separately below, terming noisy labels and incomplete labels, respectively. The neural networks used in the experiments are also introduced in this section.

6.2.1 Learning with Noisy Labels

The quality and quantity of training data and its annotations are crucial in deep learning, but the annotation process is expensive and time-consuming. With the increasing demand of labeled data, recent work focuses on improving the labeling efficiency and reducing the price of annotating images, for example, using crowdsourcing [163]. Crowdsourcing has become a popular method to acquire labels for large-scale image datasets, especially after the ImageNet dataset [30] was released. However, imperfect labels are commonly generated in any human-label process, especially via crowdsourcing. These inaccurate labels are often produced by non-experts due to carelessness, fatigue or inexperience, leading to a need to suppress or handle noisy or missing labels [43, 82, 83].

Some fundamental experiments of how deep neural networks react to noisy labels were carried out recently. For example, Zhang et al. [165] swap and delete labels on different datasets including CIFAR10 [70] and ImageNet [30] to demonstrate that deep models can learn the general patterns

of the data in early epochs but soon overfit the local patterns. Krueger [72] and Arpit [7] find that deep neural networks can practically memorize all training information including noisy (i.e. misclassified) and missing labels. At the application level, Jiang et al. [57] propose a step-by-step curriculum learning model that learns the easier patterns first and then generalizes to the more difficult cases. Li et al. [80] propose a method to perform a meta-learning update before a gradient update, where they generate synthetic noisy labels in the training process to simulate real-life annotation noise. Goldberger and Ben-Reuven [45] use the expectation maximization (EM) algorithm to estimate the correct labels and retrain the network after a certain number of epochs. Northcutt et al. [108] use confident learning to estimate label errors and refine learning model by assuming the dependency between the noise and true labels.

6.2.2 Learning with Incomplete Labels

Labeling fewer targets per image can dramatically reduce the difficulty and effort of annotating images, especially those with a lot of interested objects. This is different from what often happens in semi-supervised learning where the number of labeled images, rather than the number of labels in an image, is decreased. It is also different from typical scenarios in weakly-supervised learning using coarse-grained supervision, where pixel-level annotations are reduced to image-level labels (i.e. easier labels).

However, to the best of our knowledge, very limited amount of literature attempts to deal with the scenarios with incomplete pixel-level labels. For example, Nguyen et al. [101] propose a loss function to learn two energy maps representing target pixels and background pixels, respectively, to refine the incomplete pixel-level annotations in binary segmentation, and

Wang et al. [150] propose to propagate the incomplete bounding box per image using a positiveness-focused object detector to for detection-based object counting.

6.2.3 Limitations of Previous Work on Noisy or Incomplete Labels

Recent literature favors noisy labels (e.g. randomly assigned labels) over incomplete labels, and image-level labels (e.g. classification labels) over pixel-level annotations (e.g. point-annotations). Besides, many existing approaches either require tuning multiple hyper-parameters or need to modify the architecture of the learning model, which may increase the complexity of optimization.

These limitations motivate us to develop an effective yet simple method (both computationally and intuitively), to cope with intentionally- and unintentionally-missing pixel-level annotations per image. The proposed method, AMSE, is architecture-independent and has only one tuning hyper-parameter. The experiments below demonstrate the effectiveness and robustness of the proposed method on heatmap-regression-based localisation and counting tasks, with a larger number of objects of interest per image than typically occur in previous work of learning with incomplete pixel-level labels.

6.2.4 Context-Aware Network

The Context-Aware Network (CAN, [92]) is used as the base model in crowd counting experiments. The Context-Aware Network is one of the best

performing crowd counting models to date. A similar structure is adopted from the original Context-Aware Network paper [92], except AMSE is used to supervise the partially-annotated tasks. As the major contribution of this chapter is a model-independent loss function, below the mechanism of the Context-Aware Network is briefly introduced for readers' convenience, with more details found in [92].

In computer vision, contextual information refers to the relationship between a pixel and the other pixels. In density-estimation-based crowd counting, perspective distortion (e.g. due to the position of camera) can significantly affect the accuracy of local crowd density estimation. Figure 6.1 illustrates the architecture of the Context-Aware Network used in the experiments of this chapter. The Context-Aware Network attempts to represent the rapid change of local perspective using multi-scale contextual information, by inserting a contextual block (Figure 6.1 bottom) between the frontend and backend of CSRNet (introduced in Chapter 4).

The contextual block first computes the multi-scale feature maps from the previous VGG feature maps f_v using average pooling with different kernel sizes. The feature map of certain scales can be deemed as the average representation at the particular sub-area of f_v. For example, a 1×1 feature map generated by average pooling represents the average feature of the entire f_v (first row of Figure 6.1 bottom); and a 2×2 feature map generated by average pooling represent the average feature of every quarter of f_v (second row of Figure 6.1 bottom). These multi-scale feature maps are further processed by 1×1 convolution (being learnable and more expressive) and bilinear upsampling (scaling the feature maps of different scales to the same size for further computation). Therefore, the process of computing

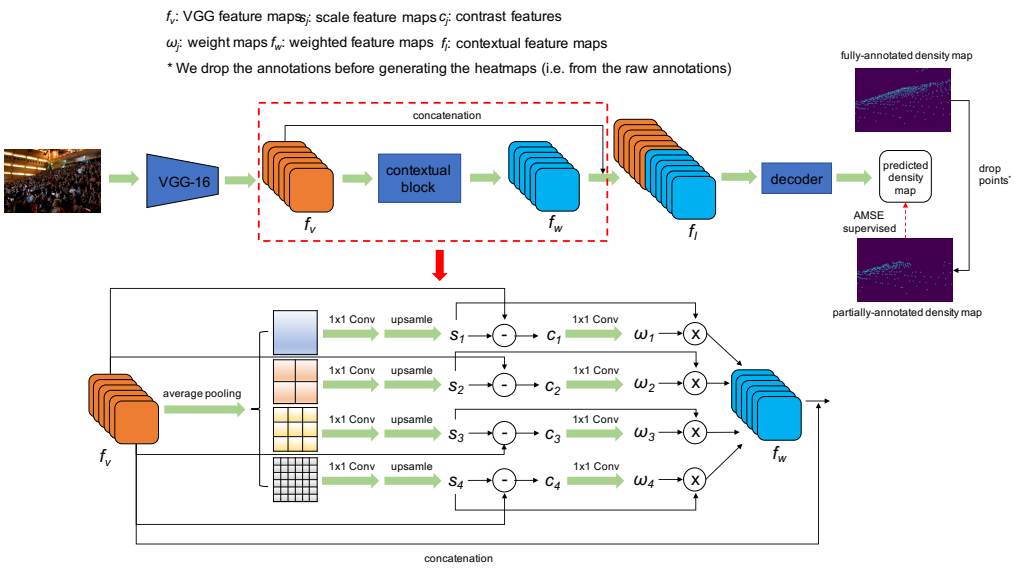


Figure 6.1: Architecture of the Context-Aware Network. **Top**: general structure of Context-Aware Network. AMSE is used to supervise the network so it can learn from partially annotated data. As can be seen, the contribution of this chapter is the AMSE and the handling of partially-annotated data on the right hand side - the rest of the architecture remains the same as [92] **Bottom**: details of the contextual block.

the scale feature map s_j can be represented as:

$$s_j = U_{bi}(F_j(P_{avg}(f_v, j), \theta_j)), \tag{6.1}$$

where $j \in 1, 2, ..., N$ and N is the number of average pooling operations; $P_{avg}(f_v, j)$ average pools f_v into $n(j) \times n(j)$ blocks; specifically, $N = 4$ is adopted, where $n(j) \in 1, 2, 3, 6$; F_j represents a 1×1 convolution operation with the parameters θ_j; and U_{bi} represents the bilinear upsampling operation.

After computing the scale feature maps, the contrast feature c_j at each scale is generated by $c_j = s_j - f_v$. Here c_j is a context feature representing the difference of a pixel and its local neighbours. This difference indicates the saliency of each pixel at multiple scales, capturing the rapid change of perspective at various local areas of the image.

Similar to the attention mechanism, the scale-specific weight map ω_j is first computed from c_j before applied back to s_j to obtain the context-weighted scale feature map f_ω, as follows:

$$\omega_j = F_{sa}^j(c_j, \theta_{sa}^j), \tag{6.2}$$

where F_{sa}^j represents a 1×1 convolution layer with the parameters θ_{sa}^j connected with a sigmoid activation function.

$$f_\omega = \frac{\sum_{j=1}^{N} \omega_j \bigodot s_j}{\sum_{j=1}^{N} \omega_j}, \tag{6.3}$$

where $\bigodot$ represents element-wise product of two matrices.

f_ω essentially incorporates the information of where the learning model should focus on in different scales. Finally f_ω is concatenated (channel-wise) with the VGG features f_v and sent to the decoder to predict the density map.

6.2.5 Stacked Hourglass Network

The Stacked Hourglass Network was first introduced to perform human pose estimation, but it is also popular for object localisation. We conduct the wheat spikelet localisation experiments based on the dataset proposed by [111]. Following the similar setting in [111], where the authors simultaneously localise spikes and spikelets, and classify awned and unawned plants using the Stacked Hourglass Network [100], we also deploy a four-block Stacked Hourglass Network with intermediate supervision, while only one task – wheat spikelet localisation is performed. The architecture of Stacked Hourglass Network used in the wheat spikelet localisation experiments in this chapter is shown in Figure 6.2. As the major contribution of

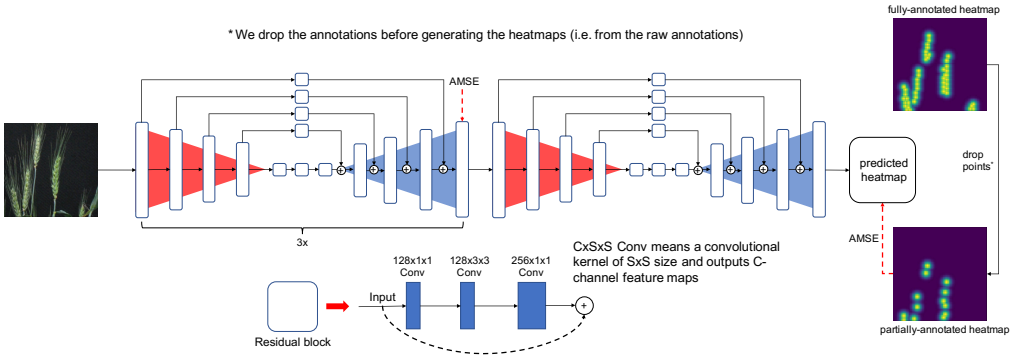


Figure 6.2: Architecture of a four-block Stacked Hourglass Network. **Top**: general structure of the Stacked Hourglass Network, based on [111]; each white block in the figure represents a residual block (**bottom**). The method proposed in this thesis uses AMSE (indicated on the right of the figure) to supervise the model so it can learn from partially annotated data.

this chapter is a model-independent loss function, below the mechanism of Stacked Hourglass Network is briefly introduced for readers' convenience, with more details can be found in [100] and [111].

The Stacked Hourglass Network stacks multiple hourglass sub-networks, with intermediate supervision performed after each. Each hourglass sub-network is an encoder-decoder structure, where the encoder extracts features by downsampling (max pooling and convolution), while the decoder upsamples these features back to the original resolution using nearest neighbor upsampling, and reconnecting the corresponding encoder features. The hourglass sub-network is designed as such to extract sufficient semantic representation from downsampling as well as preserve enough spatial information from upsampling and reconnection – the spatial information is crucial in localisation tasks. By stacking several hourglass sub-networks, the Stacked Hourglass Network is able to better capture local and global relationship, and to refine the outputs by intermediate supervision which shows performance improvement beyond a straightforward loss at the end of the network [100].

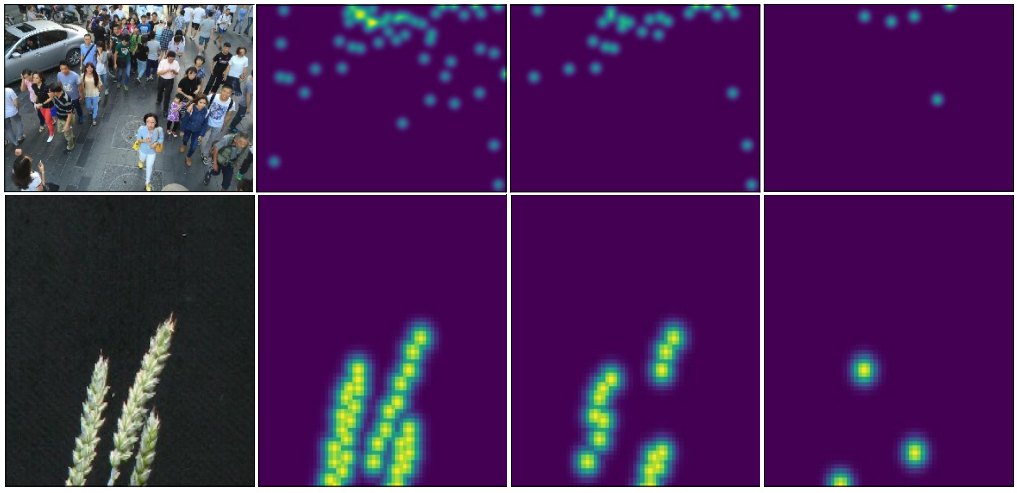
6.3 Method

6.3.1 Problem Definition

The aim of this section is training a deep network with competitive performance using a reduced number of annotations per image. This reduction of annotations can be caused by two possible scenarios: 1) *intentional* reduction in annotation quality, as a result of having a learning system which is able to cope with a quicker, less accurate annotation regimes, or 2) *accidental* errors and oversight on the part of the annotator trying to achieve "perfect" labelling. The first of these scenarios could be the result of instructing an annotator to only annotate approximately half of the objects they can see in the image, for example. If a neural network can be trained successfully in this regime, the time consuming process of labeling a dataset can be reduced whilst still acquiring high quality annotations for those that are given.

In contrast, in the second scenario the exact percentage of missing annotations on each image is difficult to estimate because these missing annotations are the result of *unknown* errors. The first, intentional scenario is more likely to produce more extreme missing data amounts, whereas the second, accidental scenario is more likely to produce more conservative drops in annotation data. The proposed AMSE is potentially valuable for both of these scenarios as different drop rates (again, the percentage data reduction) are explored.

The AMSE approach presented in this Chapter is developed to tackle deep learning-based heatmap regression problems, for object localisation and counting, on 2D RGB images. These images are point-annotated, from which a Gaussian kernel is applied to generate the heatmaps representing



(a) Input (b) Original heatmap (c) 50% reduced (d) 90%reduced

Figure 6.3: A graphical overview of reduced annotations: upper row: ShanghaiTech Part B [167], bottom row: ACID wheat dataset [111]. (a) input image, (b) original instance heatmap image, (c) and (d) show the same heatmap as (b) but with 50% and 90% of annotations dropped respectively. The aim is to train a well-performing network using dropped annotations as represented in (c) and (d).

locations and densities. Different drop rates $dr \in [0, 1]$ are applied to the fully labeled dataset to imitate the scenarios of incomplete annotations. The drop rate dr is applied equally to all the ground-truth heatmaps in the dataset and the annotations in each ground-truth heatmap are randomly dropped subject to dr. For example, if $dr = 0$, the ground-truth heatmaps are unchanged, but if $dr = 1$, then there will be no annotation in any ground-truth heatmap of that dataset. When $dr = 0.5$, there are 50% annotations dropped randomly in *each* the ground-truth heatmap.

The conducted experiments examine the AMSE performance when training a deep learning model on such a dropped dataset, and compare the performance to (1) baseline MSE methods with zero drop out (i.e. complete annotations), and (2) MSE methods with different drop rates. Figure 6.3 illustrates heatmaps with different drop rates for comparison.

6.3.2 AMSE vs. MSE

Mean square error (MSE) is expressed as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2, \tag{6.4}$$

where Y_i is the ground truth and $\hat{Y}_i$ is the predicted value.

Asymmetric MSE (AMSE) is used to counteract the effect of the reduced (missing) annotations, which is defined as:

$$AMSE = \frac{1}{N}\sum_{i=1}^{N}\{[\beta + sign(Y_i - \hat{Y}_i)](Y_i - \hat{Y}_i)\}^2, \tag{6.5}$$

where $\beta \in [-1, 1]$ is a hyper-parameter set manually, and $sign(\cdot)$ is expressed as:

$$sign(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{cases} \tag{6.6}$$

The difference between MSE and AMSE is illustrated in Figure 6.4: MSE maps the input x equally for both $x < 0$ and $x > 0$, while AMSE favours one side than the other. As a loss function, this imbalance allows us to punish the model with bias, and hence leads to better performance. In this chapter, x is set to $x = Y_i - \hat{Y}_i$, where Y_i represents ground truth and $\hat{Y}_i$ stands for prediction from the model. To counteract the reduced-annotation scenarios, $\beta > 0$ (Figure 6.4 left) is adopted because it applies a higher penalty for the scenarios of $\hat{Y}_i < Y_i$ and forces the neural network to over-predict. To note, when β in AMSE is set to 0, AMSE is equivalent to MSE.

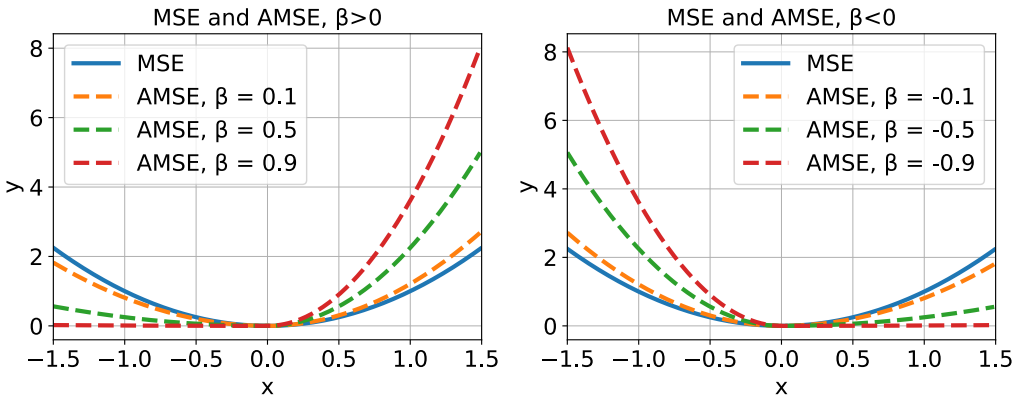


Figure 6.4: A comparison between MSE and AMSE. In both graphs, the blue solid line represents MSE, and the dashed lines show the behaviour of AMSE under different parameters (noted as β). The graph on the left shows AMSE under $\beta > 0$, where the left part of each curve is suppressed, while the graph on the right shows the opposite.

Below we discuss the use case of AMSE as a loss function to handle incomplete point-annotations in this chapter. Setting the complete number of the point-annotations (i.e. fully and correctly labeled targets) in an image to M, in the case of reduced annotations, $dr \times M$ annotations are dropped (i.e. missed in annotation) so only $(1 - dr) \times M$ annotations are left. A ground-truth heatmap is generated by applying a Gaussian kernel over the point-annotations, which results in Gaussian (hot-spot) regions centered at the corresponding points. The values within the hot-spot areas are > 0, and the remaining areas of the generated heatmap are zero-valued, which represents the background. Reducing annotations creates a lot of false negatives because some desired areas (i.e. the hot-spot areas) are treated as background. This causes locations to be lost and counts underestimated in object localisation and counting tasks respectively.

The neural network, therefore, is expected to *overestimate* the predictions especially in the false-negative areas. In other words, the model should predict larger $\hat{Y}_i$ than the ground truth Y_i. As mentioned, x in Figure 6.4 is set to $(Y_i - \hat{Y}_i)$. A greater loss should be allocated for $(Y_i - \hat{Y}_i) > 0$; and when $(Y_i - \hat{Y}_i)$ grows, a larger loss should be applied. This will guide

the neural network to learn a larger $\hat{Y}_i$. As a result, the model learns to avoid *underestimation*, which in practice cancels out the effects from reduced annotations. Therefore, $\beta > 0$ is adopted here to deal specifically with reduced-annotation cases. This parameter, β, is the key and the only tuning parameter of AMSE. As shown in Figure 6.4, when β grows, the curve becomes more unbalanced. It is reasonable to assume from this that the ideal value of β is related to the drop rate dr for a particular dataset; this is explored experimentally in the remainder of this chapter.

6.4 Experiments

The experiments of this chapter are conducted on two contrasting tasks – crowd counting and wheat spikelet localisation to explore AMSE (varying β) when different portions of point-annotations are dropped. Below we first introduce the datasets, learning models and implementation details, followed by the experiments and the relevant analysis in detail.

6.4.1 Datasets

6.4.1.1 ACID Wheat Dataset

The Annotated Crop Image Database (ACID) [111] is used for wheat spikelet localisation. ACID is constructed from high-resolution (1956x1530) wheat plant images annotated on spikelets and ears (individual wheat grains). In the experiments of this chapter, only the point-annotations on spikelets (92.3 per image in average) are used. The stats of the ACID dataset are summarized in the first row of Table 6.1. We use 364 images from ACID to form the training set and 78 different images to form the

validation set.

Table 6.1: Stats of the ShanghaiTech crowd dataset and ACID wheat dataset. The average count represents the average number of people or spikelets of the images in the corresponding dataset. To note, in the wheat spikelet localisation experiments, the images from ACID are randomly cropped to 256x256 resolution, so the average count reduces accordingly.

	resolution	number of images	average count
ACID	1956x1530	442	92.3
ShA	various	482	494.1
ShB	768x1024	716	123.2

To evaluate the scenarios of incomplete annotations, the annotations in each ground truth of the training set are randomly dropped before generating the corresponding heatmap, while the ones in the validation set remain unchanged. The dropped portions are decided by the drop rate $dr \in [0, 1]$ as discussed in Section 6.3. To note, rather than dropping from the generated heatmaps, dropping directly from the original point annotations is carried out to simulate the real-life case where the annotators omit some annotations, either as unintentional error, or as part of a "quick but messy" annotation scheme, where annotators may be told that annotating less than 100% of instances is acceptable.

After dropping the annotations in the training set, a fixed-size Gaussian kernel is applied to the point-annotations to generate the ground-truth heatmaps. Figure 6.5 left shows two examples of the images from ACID and their true heatmaps, which are generated from the complete annotations.

6.4.1.2 ShanghaiTech Crowd Dataset

ShanghaiTech crowd dataset [167] is used for crowd counting. ShanghaiTech consists of 2 parts: part A is constructed mostly of images with

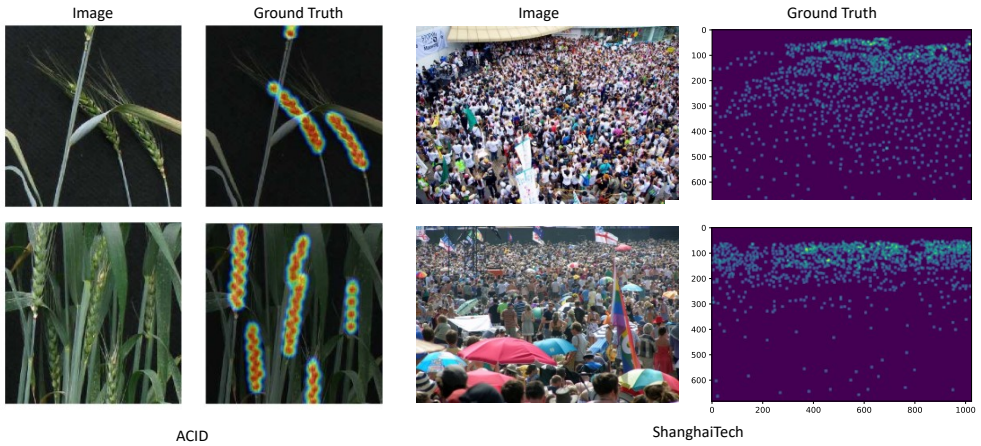


Figure 6.5: Example images and the corresponding ground truth from the datasets used in this paper. Left: ACID dataset (image source: [111]). Right: ShanghaiTech dataset (image source: [167]).

a large number of people (average 494.1), and images in part B (also used in Chapter 4 and Chapter 5) are relatively less crowded (average 123.2). Each human head in every image of the dataset is point-annotated. Part A consists of 482 images with various resolution, which are further divided into 300 training images and 182 validation images; part B contains 716 images of 768x1024 resolution, which are further split into 400 training images and 316 validation images. The stats of the ShanghaiTech crowd dataset are summarized in the second and third rows of Table 6.1.

Same as ACID, the point-annotations in each image of the training sets are randomly dropped subject to dr, while those of the validation sets remain untouched. After dropping the point-annotations, a geometry-adaptive kernel is adopted to generate the density maps for part A and a fixed kernel is used for part B, Following the same settings as previous researchers ([167, 85, 92]). Figure 6.5 right shows two examples of the images from ShanghaiTech crowd dataset and their true heatmaps, which are generated from the complete annotations.

6.4.2 Implementation Details

The original structure of Context-aware Network (CAN) [92] is used to perform crowd counting on the ShanghaiTech dataset [167]. This results in a size of the predicted heatmap of $\frac{1}{8\times8}$ of the input size. The training batch size is set to 1 for part A because the images of part A vary in size; the training batch size is set to 4 for part B to balance the performance and computational power. The detailed implementation parameters of crowd counting are shown in the second and third rows of Table 6.2.

The Stacked Hourglass Network (SHN) [100] is used to perform the spikelet localisation on the ACID dataset [111]. A similar Stacked Hourglass Network structure as used in [111] is adopted, which is to say four-stacked with intermediate supervision, and one output head to predict the heatmap. This leads to the size of the predicted heatmap is $\frac{1}{4\times4}$ of the input size. Each image in either training or validation set is cropped to 10 different 256×256-resolution sub-images, to enlarge the dataset as well as saving computational consumption. The training batch size in wheat spikelet localisation is set to 6. Table 6.2 first row shows more implementation details of the wheat spikelet localisation experiments.

Table 6.2: Training parameters of wheat spikelet localisation (ACID-SHN) and crowd counting (ShA/ShB-CAN). In the optimizer part, lr denotes learning rate, wd denotes weight decay, and mo denotes momentum.

	input res	output res	batch size	optimizer			
				name	lr	wd	mo
ACID-SHN	256×256	64×64	6	RMSProp	$2.5e-4$	0	0
ShA-CAN	various $(a \times b)$	$\frac{a}{8} \times \frac{b}{8}$	1	SGD	$1e-7$	0	0.95
ShB-CAN	768×1024	96×128	4	Adam	$1e-5$	$5e-4$	0

Importantly, only the annotations from the training sets are dropped, while those of the validation sets remain the same as the original dataset (i.e. 100% annotated). Despite dropped annotations, the neural networks

are expected to localise or count *all* instances if possible. Different values of AMSE parameter β are applied to the pairs of datasets and models with various drop rates dr to explore the relationship between β and dr, and the resulting effect on performance.

6.4.3 Crowd Counting with AMSE

In the crowd counting experiments, each model is trained on dropped data and validated on unseen and undropped data every epoch to view performance. The predicted counts are calculated by summing up the values through each pixel of the heatmaps. The validation results are presented as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), which are the same as Chapter 4 and Chapter 5.

In the following experiments, "ShanghaiTech Part A" is abbreviated to "ShA", "ShanghaiTech Part B" is abbreviated to "ShB", and, as introduced before, the Context-Aware Network is abbreviated to "CAN". For example, "ShA-CAN" represents the Context-Aware Network trained and validated on ShanghaiTech Part A.

6.4.3.1 Experiment 1: AMSE on Fully-annotated Data

First, the effects of AMSE on the original, fully-annotated dataset ($dr = 0$) are explored to show performance with "perfect" annotations. Ideally the model trained on AMSE should perform broadly as well as baseline MSE in this scenario. The validation results, reported in MAE and RMSE (lower is better), are shown in Table 6.3. Generally speaking, the errors become larger as β grows. One exception is $\beta = 0.1$, which obtains even better results than $\beta = 0$ (MSE loss function). This may indicate that the original

Table 6.3: Validation MAE and RMSE (lower is better) for $dr = 0$ under different AMSE parameters β on ShanghaiTech dataset using Context-Aware Network: (a) validation results on ShanghaiTech Part A; (b) validation results on ShanghaiTech Part B. $\beta = 0$ is equivalent to using MSE loss function and $\beta \neq 0$ stands for using AMSE loss function in training. The best result (i.e. minimum) in each column is highlighted in bold.

(a) ShA-CAN Results ($dr = 0$)

β	MAE	RMSE
0	67.3	103.5
0.1	**64.7**	**99.6**
0.2	69.4	101.2
0.3	101.1	139.6
0.5	151.2	194.7
0.7	305.9	365.4
0.9	806.6	949.8

(b) ShB-CAN Results ($dr = 0$)

β	MAE	RMSE
0	8.6	13.7
0.1	**8.4**	**13.3**
0.3	8.6	13.7
0.5	8.9	14
0.7	11.2	16
0.9	28.5	32.1

data has already missed some annotations - although this is hard to verify.

From Table 6.3, it can be concluded that an undropped dataset is indeed sensitive to different β, but for a relatively small β, for example up to $\beta \leq 0.2$ in Part A and $\beta \leq 0.5$ in Part B, the performance is not affected significantly, and sometimes even achieves a slight improvement.

6.4.3.2 Experiment 2: AMSE on Incompletely-annotated Data

The next step is to create dropped datasets from the original ShanghaiTech dataset using $dr = 0.1, 0.3, 0.5, 0.7, 0.9$, and then apply various values of β to see how the model reacts to different drop rates. Figure 6.6 (reporting MAE) and Figure 6.7 (reporting RMSE) compare the validation results between using MSE and AMSE loss function in training on ShA-CAN. Figure 6.8 (reporting MAE) and Figure 6.9 (reporting RMSE) compare the corresponding validation results on ShB-CAN. Table 6.4 (ShA-CAN) and Table 6.5 (ShB-CAN) provide numerical results, showing the best validation MAE/RMSE achieved for each dr and the corresponding optimal β of

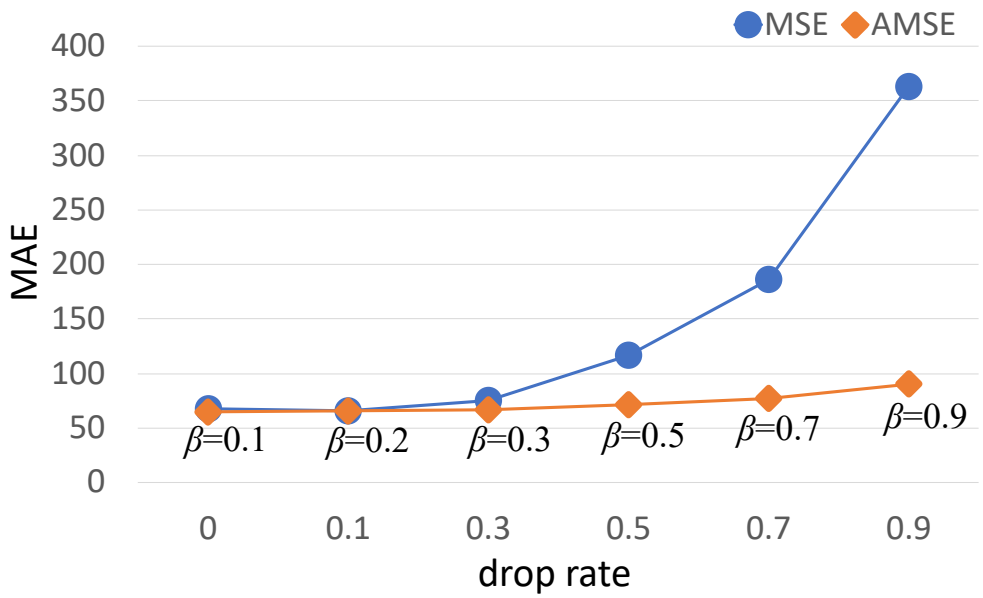


Figure 6.6: Validation MAE (lower is better) on ShA-CAN under different drop rates. Blue line: results obtained by using MSE loss function (i.e. $\beta = 0$) in training. Orange line: best results obtained by using AMSE loss function in training, with β value under each point showing when the corresponding result is achieved.

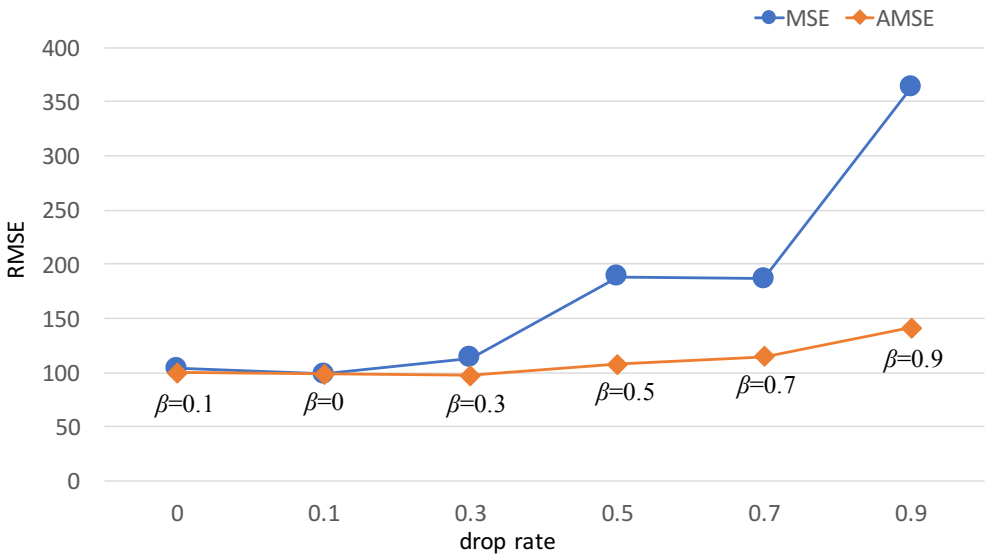


Figure 6.7: Validation RMSE (lower is better) on ShA-CAN under different drop rates. Blue line: results obtained by using MSE loss function (i.e. $\beta = 0$) in training. Orange line: best results obtained by using AMSE loss function in training, with β value under each point showing when the corresponding result is achieved.

the proposed method. As a comparison, the results achieved by training with the original MSE loss function are shown in the column of $\beta = 0$.

Table 6.4: Validation MAE/RMSE (lower is better) on ShA-CAN under different drop rates. Columns under $\beta = 0$: results obtained by training with MSE loss function. Columns under $\beta \neq 0$: best results obtained by training with AMSE loss function, with β value in bracket showing when the corresponding result is achieved.

dr	MAE		RMSE	
	$\beta = 0$	$\beta \neq 0$ (best β)	$\beta = 0$	$\beta \neq 0$ (best β)
0	67.3	**64.7** (0.1)	103.5	**99.6** (0.1)
0.1	65.7	**65.7** (0.2)	98.9	**98.9** (0)
0.3	75.3	**66.7** (0.3)	113.6	**97.6** (0.3)
0.5	116.9	**71.4** (0.5)	188.5	**107.4** (0.5)
0.7	186.3	**77.3** (0.7)	186.3	**114.8** (0.7)
0.9	363.2	**90.5** (0.9)	363.2	**141.4** (0.9)

Figure 6.6, which presents validation MAE in ShA-CAN, shows that as drop rate grows, the performance gap between MSE (blue line) and AMSE (orange line) becomes larger; particularly when $dr \geq 0.3$ AMSE significantly outperforms MSE. A similar trend is observed in Figure 6.7 where validation RMSE of ShA-CAN is shown. Table 6.4 indicates that when $dr \leq 0.3$, the results achieved by AMSE not only beat those achieved by MSE on the same drop rate but also are marginally better than the model trained on the undropped dataset with MSE loss function, i.e. $dr = 0$ and $\beta = 0$ in the tables. The results for $dr = 0.5$ show that with 50% data dropped, the performance only reduces 6.1% in MAE (67.3 to 71.4) and 3.8% (103.5 to 107.4) in RMSE compared to the baseline. Even in the very extreme cases, where 70% and 90% of annotations are dropped per image, the models only experience 14.9% (67.3 to 77.3) and 34.5% (67.3 to 90.5) dropped in MAE, respectively. Although these extreme cases are unlikely to happen in real life (unless annotators are instructed to behave this way), they still show the robustness and effectiveness of the proposed method. To note, the optimal β is selected by evaluating multiple values of β on a

particular dr and reporting the one that leads to the best validation result.

Table 6.5: Validation MAE/RMSE (lower is better) on ShB-CAN under different drop rates. Columns under $\beta = 0$: results obtained by training with MSE loss function. Columns under $\beta \neq 0$: best results obtained by training with AMSE loss function, with β value in bracket showing when the corresponding result is achieved.

dr	MAE		RMSE	
	$\beta = 0$	$\beta \neq 0$ (best β)	$\beta = 0$	$\beta \neq 0$ (best β)
0	8.6	**8.4** (0.1)	13.7	**13.3** (0.1)
0.1	9.2	**8.8** (0.1)	14.8	**14.4** (0.1)
0.3	18.1	**10** (0.5)	19.7	**15.9** (0.3)
0.5	48.4	**12.6** (0.8)	68.2	**17.9** (0.8)
0.7	75.5	**16.9** (0.95)	99.1	**26.8** (0.95)
0.9	102.7	**34** (0.95)	133.8	**50.8** (0.95)

Similar trends are also shown in ShB-CAN validation results. From Figure 6.8 and Figure 6.9 it can be observed that when $dr \geq 0.3$ the validation MAE and RMSE achieved by AMSE exceed those obtained by MSE loss function; and the gap in performance becomes larger when $dr \geq 0.5$. Table 6.5 shows that when $dr \leq 0.3$ the validation results obtained by AMSE are better than those achieved by MSE on the same drop rate, and also very close to the baseline (i.e. $dr = 0, \beta = 0$). When $dr \geq 0.5$, the performance of models trained with AMSE is far better than the ones trained with MSE. A general preference for increasing β with dr is also observed, although this pattern is slightly less consistent for the ShB-CAN.

To summarize, the results on ShA/ShB-CAN on various dr shows that AMSE outperforms MSE when annotations are dropped in training. Particularly, if one is asked to annotate 70% of points per image (ie. dr=0.3), using AMSE with a suitable β can still obtain performance on par with training on 100%-annotated data using MSE. If one wishes to further reduce annotations, the results of AMSE will still fall within an acceptable area until dr becomes extreme (i.e. $dr > 0.7$).

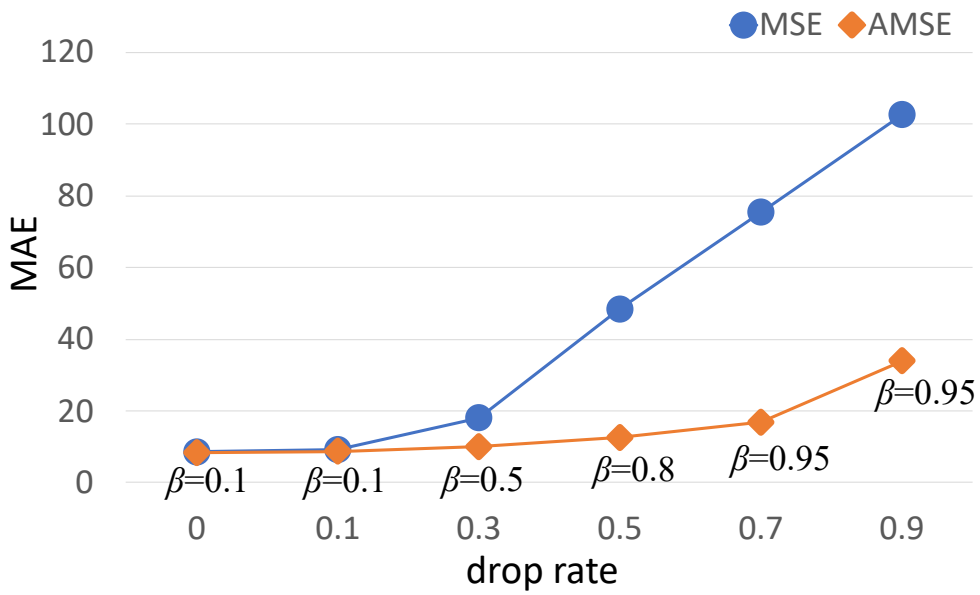


Figure 6.8: Validation MAE (lower is better) on ShB-CAN under different drop rates. Blue line: results obtained by using MSE loss function (i.e. $\beta = 0$) in training. Orange line: best results obtained by using AMSE loss function in training, with β value under each point showing when the corresponding result is achieved.

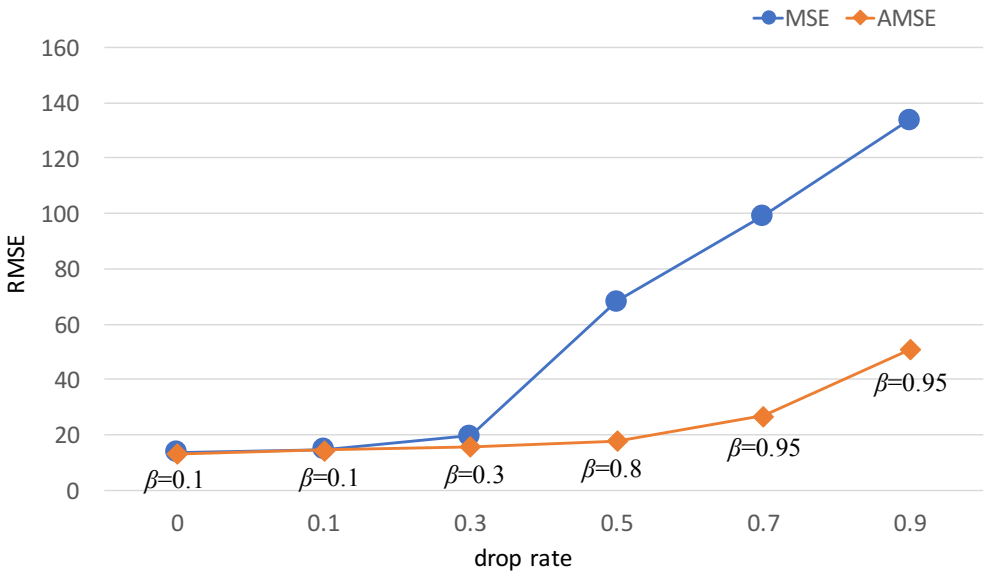


Figure 6.9: Validation RMSE (lower is better) on ShB-CAN under different drop rates. Blue line: results obtained by using MSE loss function (i.e. $\beta = 0$) in training. Orange line: best results obtained by using AMSE loss function in training, with β value under each point showing when the corresponding result is achieved.

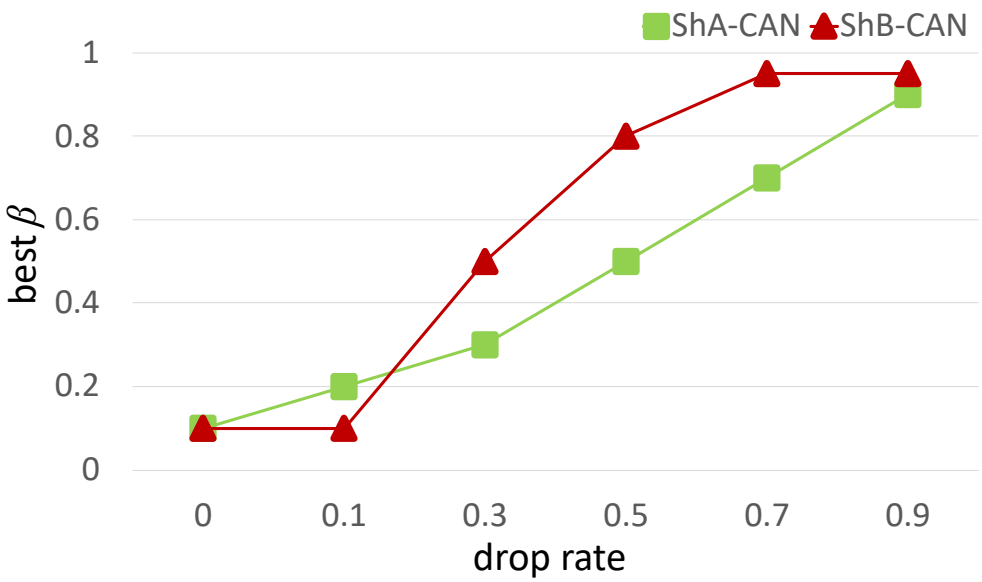


Figure 6.10: β value where the best validation MAE is achieved under each drop rate on ShA/ShB-CAN.

Plotting dr against β of the best MAE achieved (Figure 6.10) indicates that the optimal β value increases as the dr grows larger. This provides some hints in how to select a proper β in real life scenarios.

6.4.3.3 Experiment 3 (Repeated): AMSE vs MSE on $dr = 0.5$

To eliminate the contingency from the previous experiments caused by the random initialization of the learning model and the single split of the dropped-annotation datasets, and to reveal how differently the two loss functions, MSE and AMSE, behave on a dropped dataset, repeated experiments are conducted on ShA/ShB-CAN for a fixed drop rate, $dr = 0.5$. For ShA and ShB dataset respectively, five different folds are created by randomly dropping a different subset of the annotations per image (by applying different random seeds). The neural network is then trained using MSE and AMSE with the best performing β noted from earlier experiments above (i.e. $\beta = 0.5$ for ShA and $\beta = 0.8$ for ShB) on these folds.

The validation results of the model are reported by average error, bias and standard deviation, which are shown in Figure 6.11. MAE is a more common metric for model selection in crowd counting so RMSE is not reported here – its response is also very similar to MAE. *Bias* is introduced here to better reveal the mechanism of AMSE, which is defined as:

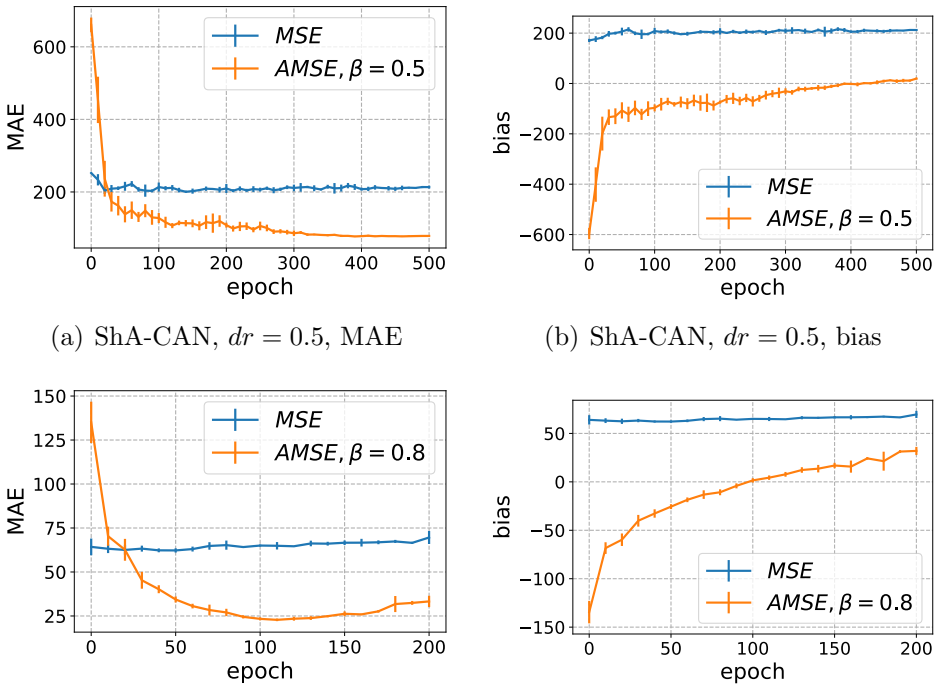
$$bias = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i), \tag{6.7}$$

where N denotes the total number of images, Y_i denotes the true number of crowd counts in image i and $\hat{Y}_i$ denotes the predicted crowd counts of the same image. Bias not only measures the absolute errors (i.e. such as MAE), but also reports over- or under-estimation of the count.

It is observed from Figure 6.11 panels (a) and (c) (reporting MAE), that the proposed method (AMSE, orange line) is robust to different random samples of annotations as the standard deviation of MAE is small; AMSE also consistently outperforms MSE (blue line), after sufficient training, using the same dropped dataset.

Figure 6.11 panels (b) and (d) illustrate the bias as defined in Equation (6.7): $bias < 0$ means the predicted count is greater than the true count (i.e. prediction is *overestimated*) and $bias > 0$ represents the opposite (*underestimation*). Figure 6.11 panels (b) and (d) which report validation bias indicates that:

- The proposed method (AMSE, orange line) starts from predicting overestimated counts and gradually forces the model to predict true counts during training before overfitting the dropped training data. However, eventually it leads the model to *under-count* if training proceeds.



(a) ShA-CAN, $dr = 0.5$, MAE

(b) ShA-CAN, $dr = 0.5$, bias

(c) ShB-CAN, $dr = 0.5$, MAE

(d) ShB-CAN, $dr = 0.5$, bias

Figure 6.11: Average validation MAE/bias of ShA/ShB-CAN on $dr = 0.5$ from repeated experiments. Under each parameter set, the curves show the mean and standard deviation (error bars) of 5 repeated experiments, with various random seeds to generate different dropped training data. All the curves are smoothed after every 10 epochs. (a)/(b) Average validation MAE/bias of ShA-CAN on $dr = 0.5$ with MSE and AMSE of $\beta = 0.5$ (i.e. optimal *beta*, below as the same); (c)/(d) Average validation MAE/bias of ShB-CAN on $dr = 0.5$ with MSE and AMSE of $\beta = 0.8$.

- The model trained with MSE (blue line) overfits the dropped data at the very beginning, and consequently predicts a steady underestimated count throughout training.

- The bias of AMSE crosses zero at around epoch 400 and 100 in ShA/ShB respectively, indicating when the model begins to undercount. This matches panel (a) and (c) where the curves show that the models plateau or start performing worse from the same points.

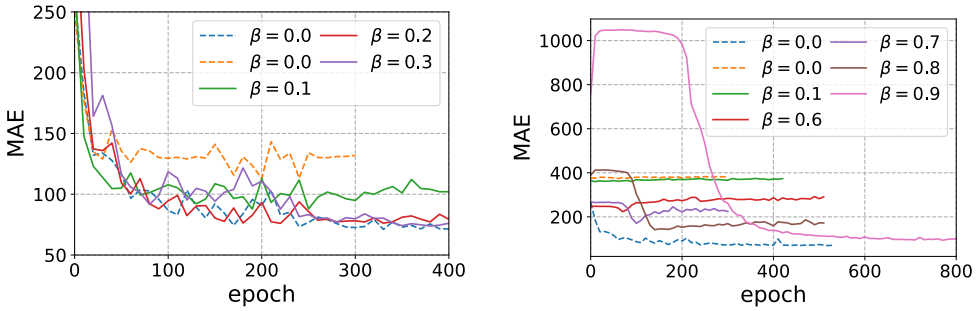
To summarize, AMSE begins with strong overestimation and gradually moves towards the true count as training progresses. MSE, however, already underestimates the count at the start, as expected, and never has

the opportunity to predict correctly. What is key to note is that over training with AMSE can also lead to underestimation if the model is allowed to overfit the training data. Therefore, the optimal strategy of using AMSE may be training the model for a sufficient amount of time, then selecting the model with lowest validation error - rather than simply training for a fixed number of epochs and selecting the final model as the best. The graphs suggest this could potentially be used to locate the zero-crossing point for the bias - where the model is optimally trained.

6.4.3.4 The Impacts of Different β of AMSE

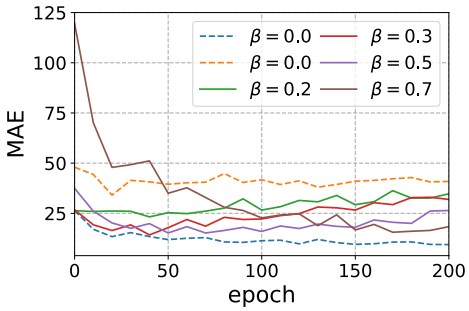
After demonstrating the effectiveness and robustness of the proposed AMSE, we also evaluated how varying the only parameter of AMSE – β influences the performance of AMSE on a particular dr. Clearly the improvement afforded by AMSE is highly dependant on a suitable choice of β. Figure 6.12 presents validation MAE curves of ShA/ShB-CAN with different β under dr=0.3 and 0.9, which imitates a realistic case of unintentional missing of a few annotations, and an extreme case of intentionally dropping many annotations respectively. Figure 6.12 shows that:

- Almost every β improves the performance given sufficient training time, while there is an optimal choice of β among all values for each dr.
- Under a given dr, a relatively high (not necessarily the highest) β makes the network start from a larger validation error and converge slower. However, the network can be trained longer before overfitting the reduced annotations and hence converge to a better end point. For example in Figure 6.12 (b), which shows the validation results of ShA-CAN under $dr = 0.9$, the pink solid line ($\beta = 0.9$) clearly

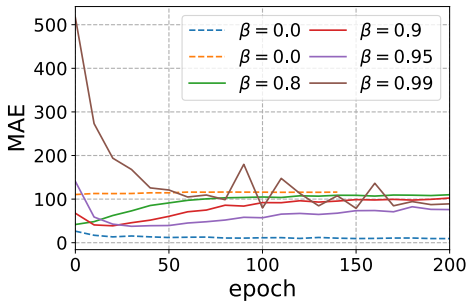


(a) ShA-CAN, $dr = 0.3$

(b) ShA-CAN, $dr = 0.9$



(c) ShB-CAN, $dr = 0.3$



(d) ShB-CAN, $dr = 0.9$

Figure 6.12: Validation MAE of ShA/ShB-CAN with various β on $dr = 0.3$ and $dr = 0.9$ through training: (a)/(b) validation MAE of ShA-CAN trained on $dr = 0.3/dr = 0.9$, (c)/(d) validation MAE of ShB-CAN trained on $dr = 0.3/dr = 0.9$. The results are averaged every 10th epoch. In each sub-graph, the blue dashed line (trained on fully-annotated data with MSE loss function) and the orange dashed line (trained on reduced data with MSE loss function) delineate the baseline boundaries of the experiments. The solid lines (trained on reduced data with AMSE loss function) are expected to fall between the dashed lines to show the effectiveness of AMSE; those closer to the blue dashed line show better performance.

converges later than the other parameters, and to a lower MAE.

- Under a given dr, choosing a too high β, may reduce performance, though it is still better than not using AMSE (orange dashed line). For example, in Figure 6.12 (d), which shows validation results of ShB-CAN under $dr = 0.9$, the brown solid line ($\beta = 0.99$) performs worse than the purple solid line ($\beta = 0.95$), while it is still better than the orange dashed line (i.e. using MSE).

To conclude, AMSE is able to compensate for reduced annotations

by allowing the network to train longer and learn better before overfitting. Choosing an exact β is more complex because it depends on different dr and data. In the next section, we consider applying AMSE on wheat spikelet localisation, where the positions of the objects matter.

6.4.4 Wheat Spikelet Localisation with AMSE

Spikelet localisation is a more complicated task than crowd counting because it aims to *localise* each spikelet instead of directly counting the targets. In the following experiments, each model is trained on data of reduced annotations and evaluated on the fully-annotated unseen set every five training epochs. Following the same setting as [111], the location of each spikelet is calculated by non-maximum suppression over the predicted heatmap and the validation results are presented as the F1 score, which is defined as:

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}, \tag{6.8}$$

where TP, FP and FN represent number of true positives, false positives and false negatives, respectively.

True positives are the predicted points on the positions of the corresponding ground-truth points; false positives are the predicted points that do not match any ground-truth point (i.e. predictions on the non-spikelet positions); false negatives are the ground-truth points that do not match any predicted counterpart. As perfect pixel-level predictions are difficult, in practice, we allow a one-pixel error for the predictions (the resolution of the predicted heatmaps is 64×64). To note, as F1 score is used here, higher results are better (opposite to the previous crowd counting experiments).

6.4.4.1 Experiment 1: AMSE on Incompletely-annotated Data

Similar to the experiments in crowd counting, the Stacked Hourglass Network is trained on the wheat spikelet localisation task (abbreviated to spikelet-SHN below) under regimes of $dr = 0.1, 0.3, 0.5, 0.7, 0.9$ using the AMSE loss function. Figure 6.13 and Table 6.6 compare the validation F1 score between training with MSE and AMSE. Here the impact of applying AMSE on a fully-annotated dataset (i.e. $dr = 0$) is not explored because it was demonstrated in the more generic crowd counting.

Table 6.6: Validation F1 score on spikelet-SHN under different drop rates. $\beta = 0$: results obtained by training with MSE loss function. $\beta \neq 0$: best results obtained by training with AMSE loss function, with β value in bracket showing when the corresponding result is achieved.

dr	F1-score (%)	
	$\beta = 0$	$\beta \neq 0$ (best β)
0	84.2	-
0.1	78.5	**80.2** (0.1)
0.3	65.9	**82.3** (0.7)
0.5	50.8	**78.7** (0.7)
0.7	27.2	**77.9** (0.9)
0.9	8.1	**70.3** (0.9)

Figure 6.13 shows that the validation F1 score of the model trained with MSE (blue line) worsens as the drop rate increases, while the performance of the model trained with AMSE (orange line) still holds steady. Table 6.6 shows that with 30% and 50% annotations dropped, training with AMSE only experiences a 2.3% (84.2 to 82.3) and 6.5% (84.2 to 78.7) performance drop compared to the baseline ($dr = 0, \beta = 0$); even under the extreme case where only 10% annotations remain (dr=0.9), the model trained with AMSE can still achieve 83.5% (70.3 / 84.2) of the performance of the baseline, while the model trained with MSE shows far below acceptable performance. It is also observed from Table 6.6 that the optimal β

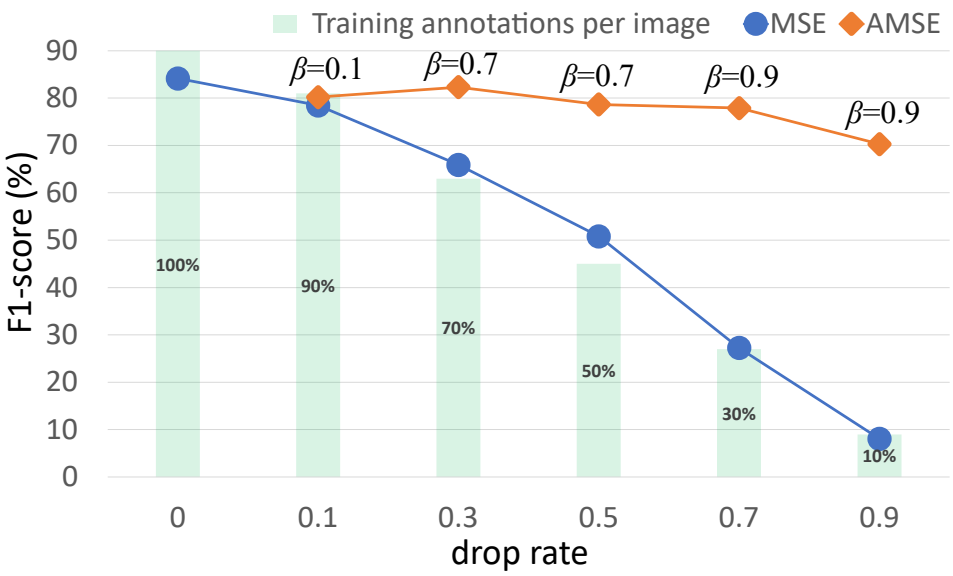
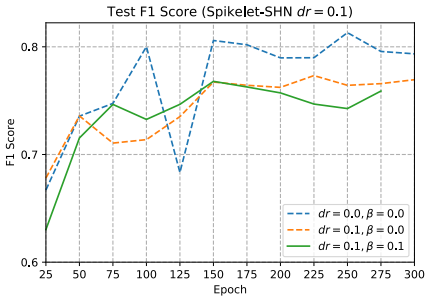


Figure 6.13: Validation F1 score (higher is better) on spikelet localisation under different drop rates. Blue line: results obtained by using MSE loss function (i.e. $\beta = 0$) in training. Orange line: best results obtained by using AMSE loss function in training, with β value above each point showing when the corresponding result is achieved. Green bar: what percentage of annotations per image is used in training (100% stands for fully-annotated data).

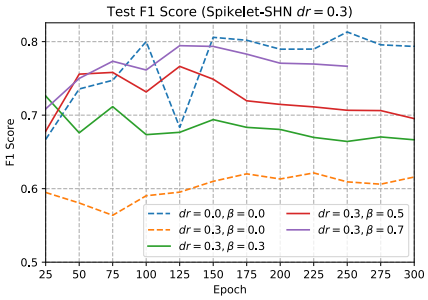
tends to increase as dr grows, in a similar way to how it behaves in the crowd counting experiments. The nature of the increase is not linear, but still shows a monotonic relationship.

6.4.4.2 The Impacts of Different β values in AMSE

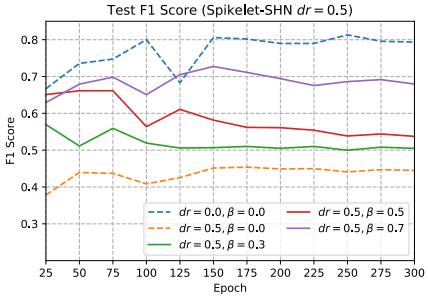
Moreover, the validation curves throughout the training process with various dr and β are shown in Figure 6.14. The models are validated after every 5 training epochs on unseen fully-annotated images, and the presented curves are smoothed by averaging every 5 results. Figure 6.14 again confirms that AMSE is able to delay overfitting, and helps the models learn better on dropped annotations, with performance nearly always falling between the baseline best case (fully-annotated data, MSE) and baseline worse case (dropped data, MSE).



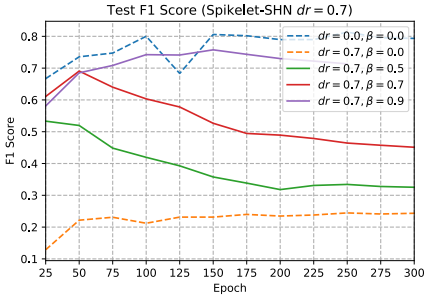
(a) $dr = 0.1$



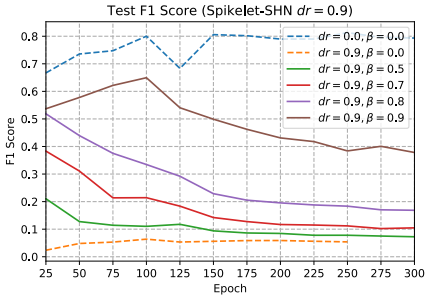
(b) $dr = 0.3$



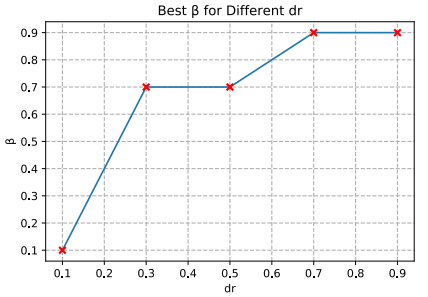
(c) $dr = 0.5$



(d) $dr = 0.7$



(e) $dr = 0.9$



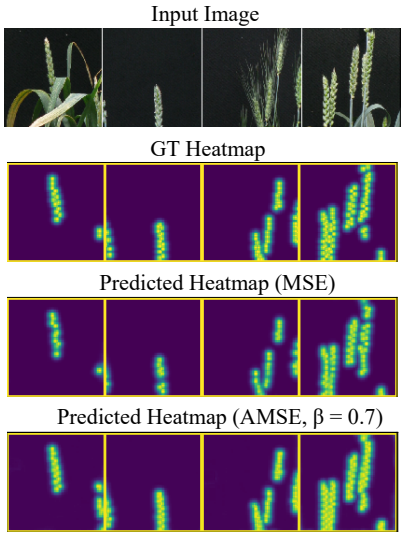
(f) best β under different dr

Figure 6.14: (a) – (e): validation F1 scores of spikelet-SHN under $dr = 0.1, 0.3, 0.5, 0.7, 0.9$ respectively through training. Each curve is smoothed by averaging every 5 results. In each sub figure, the blue dashed line (trained on fully-annotated data with MSE loss function) and the orange dashed line (trained on dropped data with MSE loss function) delineate the baseline boundaries of the experiments. The solid lines are trained on dropped data with AMSE of different values of β. (f): The best value of β for each drop rate.

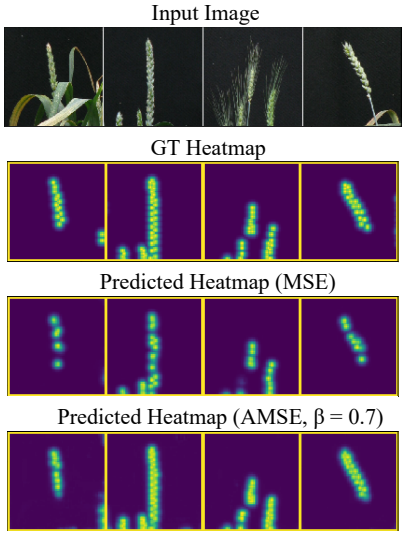
6.4.4.3 Visual Results of Wheat Spikelet Localisation

As mentioned before, spikelet localisation aims to obtain the accurate *position* of each spike (rather than simply a count), so here some graphical results are presented in addition to the F1 scores. Figure 6.15 compares the predicted heatmaps on validation data from the models trained with MSE and AMSE respectively, when $dr = 0.3, 0.5, 0.7, 0.9$. This figure shows that the models trained with MSE predict much fewer spikelets compared to the ground truth (2nd row versus 3rd row in sub figures) especially under extreme cases where $dr \geq 0.5$. However, the predictions from the models trained with loss are very close to the ground truth (2nd row versus 4th row in sub figures), even when the drop rate is extreme at $dr = 0.9$. The performance gap between using MSE and AMSE becomes increasingly wider as dr grows. In Figure 6.15 (d), the predicted heatmaps from the AMSE model (4th row) contain some noticeably brighter background. This is a side effect when a large β is applied (e.g. $\beta = 0.9$) because the model tends to predict higher pixel values globally. However, it does not affect the accuracy of the predicted wheat spikelet positions. Applying a simple threshold method, or looking for local maximum, would be able to counteract the background, if necessary.

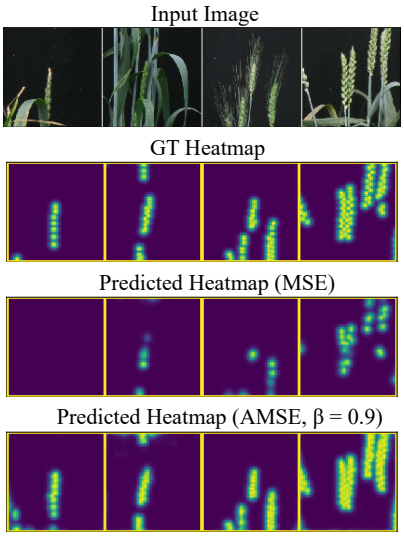
The F1 scores (Table 6.6) and the graphical results (Figure 6.15) demonstrate that AMSE is able to effectively counteract the absence of some point-annotations in wheat spikelet localisation, even when the missing portion is large (e.g. $dr = 0.9$). On the other hand, images from the ACID dataset have consistent background and the average number of targets (i.e. spikelets) is relatively small compared to crowd counting, which may be easier to localise more true positives even when the training labels are inaccurate and incomplete.



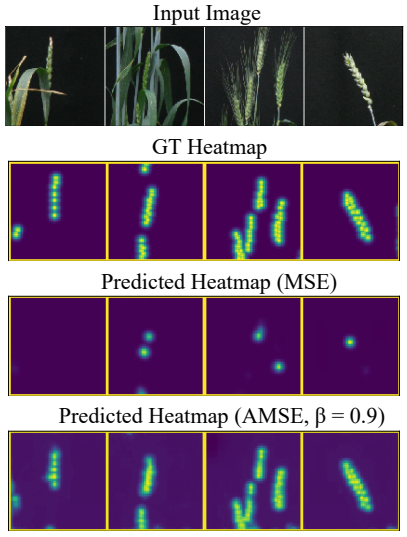
(a) $dr = 0.3$, best $\beta = 0.7$



(b) $dr = 0.5$, best $\beta = 0.7$



(c) $dr = 0.7$, best $\beta = 0.9$



(d) $dr = 0.9$, best $\beta = 0.9$

Figure 6.15: Inputs, ground truth and predicted heatmaps of spikelet localisation under different dr. The first row in each sub figure shows the validation images. The second row in each sub figure shows the ground truth (fully annotated). The third row in each sub figure shows the predictions by the model trained using MSE loss on the dropped data under the given dr. The fourth row in each sub figure shows the predictions by the model trained using AMSE loss with the optimal β under the given dr.

6.5 Discussion

6.5.1 Summaries of AMSE

The experiments of crowd counting and wheat spikelet localisation have shown that the proposed AMSE loss function can achieve performance close to (or even better than) the no-dropout baseline in the presence of substantial dropped annotations when the only tuning parameter β is set properly. Even without tuning, a small value of β still leads to better performance than using MSE in some cases. This result provides tolerance to unintentionally missing annotations, and potentially more practical annotation guidelines, where 100% annotation coverage is not required.

The proposed method is not restricted to object counting and localisation; it could be expanded to other learning tasks. AMSE may also provide some insights of how to deal with other non-intentional errors in annotations. For example, human annotators may *over-count* the instances in the images when the gaps between the objects in the images are blurry; in such case, using AMSE with $\beta < 0$ may help ease the situation, which can be explored in the future.

Despite the effectiveness of the AMSE method, the downside of using the AMSE approach would seem to be a relatively longer training time when β is large; however, too long also causes overfitting, so a balance is necessary, as with most training regimes. The estimation of dr and β is also worth discussing:

- On an intentionally-dropped dataset, where the annotators are instructed to label fewer targets, dr is known, which simplifies the case. Annotators can be explicitly instructed to only label about half the

instances, for example. However, in the case where the annotations are missed due to fatigue or carelessness of the annotators, dr needs to be determined. This dr could be estimated empirically by careful annotation of a small subset of the training data, and comparing this with the existing annotations.

- The best β is positively related to dr, but the value of the optimal β varies in different data and models. One could empirically determine the best β for the AMSE loss by trial and error on a subset of a dataset. This may require further mathematical exploration on the patterns of AMSE under different drop rates.

Nevertheless, if the dr or β is still unclear, gradually increasing β from a relatively small value (e.g. $\beta = 0.1$) can potentially improve the neural network. As shown in Section 6.4.3, a small β does not worsen the performance in our experiments; it may even benefit the model trained on the "perfectly-annotated" data. This indicates that the labels of the public datasets may be imperfect, although the true drop rates of them are difficult to estimate.

6.5.2 Limitations of Experiment Settings

Instructing the annotators to ignore some targets to speed up annotation may result in only clear and easy-to-see instances being labeled, whilst the challenging objects are omitted, which would skew the training set and reduce the learning models' capabilities of generalization and coping with complex senarios. This is not considered in this chapter as the annotations are *randomly* dropped from the original datasets. Future work should consider the effect of this potential skewing.

In addition, in this chapter, drop rates are set to be consistent across all training images. However, in real-life applications, drop rates fluctuate among images because labeling exact portions of visible objects of an image is a difficult task for humans; it is also unlikely that annotators unintentionally miss a same percentage of labels in every image. The effectiveness of AMSE upon this more realistic situation should be investigated in the future.

To note, in this chapter, pixel-level labels are dropped subject to a preset drop rate at every training image of original resolution (1956×1530), while in the wheat spikelet localisation experiments, the training images together with their ground-truth heatmaps (generated from the dropped point annotations, as described in Section 6.4.1.1) are randomly cropped to a smaller resolution (256×256 as described in Section 6.4.2) before sending to the model, which may create crops varying drop rates. As AMSE works quite well in wheat spikelet localisation, it may indicate that AMSE is capable to deal with inconsistent drop rates among training images.

6.6 Conclusions

This chapter discusses a novel scenario – training with fewer annotations per image – which is a combination of weakly-supervised learning and semi-supervised learning. This scenario can potentially create a more efficient annotation process to save time and efforts, particularly in the tasks with a large number of similar objects; for example, annotators could be advised to annotate half of the visible objects per image. A simple yet effective method – AMSE is proposed and explored, to not only adapt to this scenario, but also improve the neural networks without modifying the fundamental

structure of them when the ground truth is, or is likely to be, incomplete.

The results of the experiments on wheat spikelet localisation and crowd counting demonstrate the effectiveness and robustness of AMSE, even when removing 90% of annotations. When $dr \leq 0.5$, using AMSE in training can achieve comparable performance as using a fully-annotated dataset with MSE. The approach requires only one parameter to be chosen – β - which is positively related to the drop rate of the data, though the nature of this relationship does vary depending on dataset. It is worth deploying to save the efforts of labeling or counteract the unintentionally missing labels.

Chapter 7

Conclusions and Future Work

7.1 Summaries of the Work

This thesis aims to address the problem of training deep learning models with *fewer labels* in image-based plant phenotyping. Active learning, semi-supervised learning, and weakly-supervised learning approaches are applied, improved or proposed to address various plant phenotyping tasks, with extended verification on more widely relevant, more generic computer vision tasks. The findings of the thesis provide the plant phenotyping community with how to effectively use limited data and lower annotation requirements to develop deep learning models which still perform well. The main contributions of the thesis are concluded below:

- Chapter 3 evaluates different active learning techniques in flower classification, showing that the number of labels can be reduced by incrementally selecting and labeling the most informative data.

- Chapter 4 explores semi-supervised heatmap-regression-based crowd counting and aphid counting with a consistency-regularization-based approach, Mean Teacher. An adaptive method of updating the weight of the consistency loss is proposed to replace the common ramp-up functions. Rather than simply assuming the increasing reliability of the teacher model during training, the proposed method considers the fluctuation of the performance and calculates weights dynamically by evaluating the teacher model on the labeled data in every training iteration. The proposed adaptive weight not only improves the performance, but also reduces the complexity of optimizing Mean Teacher on aphid counting.

  In addition, we propose an aphid dataset of 199 high-resolution images with both pixel-level annotations (i.e. point-annotations on four

types of insects) and image-level labels (i.e. confidence label), which will boost the research of image-based plant phenotyping.

- As Mean Teacher is less effective on the more general task of crowd counting, Chapter 5 explores FixMatch, which is based on pseudo-labeling and consistency regularization, on semi-supervised heatmap-regression-based crowd counting and aphid counting.

  FixMatch is initially proposed for image classification, using Cutout to enhance the data transformation, and thresholding the predicted probabilities to select the confident pseudo-labels. However, in heatmap regression, Cutout is inappropriate – causing inconsistency between the pair of input and output, and estimating the confidence of predictions is difficult. Therefore, we propose to use FixMatch with *Monte Carlo Dropout* and *Shadowout* in semi-supervised heatmap regression. Monte Carlo Dropout provides a practical solution to estimate the uncertainty on regression tasks, by simply calculating the mean and variance from multiple predictions on the same image. Shadowout, which randomly alters the brightness of multiple random areas, instead of brutally blocking a region of an image, which can distort counts, can also be applied to other tasks where local information of the inputs affects the outputs. The experiments on both crowd counting and aphid counting demonstrate the effectiveness and robustness of the proposed methods.

- Chapter 6 explores a task relevant to both semi-supervised and weakly-supervised learning – learning with fewer pixel-level annotations per image. The incomplete annotations can be caused unintentionally, for instance, due to fatigue and carelessness, or intentionally, for example, the human annotators are instructed to only label a few objects on each image. The latter leads to a potentially more efficient and

more accurate annotating scheme, where the workload and difficulty of labeling a dataset and each single image are both reduced.

Learning with fewer pixel-level labels is less explored; the existing methods are complex in terms of model architecture or hyper-parameters and mainly address segmentation or detection tasks, which often contain fewer objects. We propose to use an asymmetric MSE (AMSE) loss function to train deep learning models on heatmap-regression-based crowd counting and wheat spikelet localisation with fewer point-annotations. The counting and localisation datasets generally have more targets in the images than the previous research of training with fewer pixel-level labels. The proposed AMSE is effective and robust across various models and datasets. In addition, AMSE only has one hyper-parameter, which is related to how few point-annotations there are, simplifying the optimization of the algorithm.

7.2 Limitations of the Work

Although the proposed methods show effectiveness and robustness in reducing the labeling expenses and efforts, they can still be improved. The limitations of the proposed approaches are summarized below:

- In Chapter 3 (active learning on flower classification), first, a model of active learning may lack the ability of self-correction when it is misled by some training samples. Second, in real application, a real-time system to enable a rapid interaction between an active learning model and human annotators may be needed. Third, the annotating process is often an open-set problem, meaning that not all the target classes are known in advance, but only closed-set problems are discussed in

the chapter.

- In Chapter 4 (adaptive Mean Teacher on aphid and crowd counting), first, the proposed adaptive consistency weight is calculated based on evaluating the teacher model on the labeled data of each batch, which may be inaccurate because the number of labeled images per batch is relatively small. Second, the proposed adaptive consistency weight favors the images with fewer targets, which may skew the learning process.

- In Chapter 5 (FixMatch with Monte-Carlo Dropout and Shadowout on aphid and crowd counting), the main limitation is that Monte-Carlo Dropout only estimates the uncertainty from a model, but the uncertainty from data is ignored. One way to fix this problem is to integrate aleatoric uncertainty (introduced in Section 5.5) into the model, which often requires an additional predicting head that may be harmful to performance of the model. Another way to incorporate the uncertainty of data is to apply different data augmentation parameters (e.g. using Shadowout with a smaller intensity range) before each time performing Monte-Carlo Dropout, which does not require modification of the structure of a model.

- In Chapter 6 (incomplete pixel-level supervision using AMSE on wheat spikelet localisation and crowd counting), first, in the setting of experiments, the annotations are randomly dropped. However, in reality, annotators may only label the big or clear targets, which may skew the training set and reduce the accuracy of counting or localisation. Second, the drop rate is set to be consistent on each image in our experiments. However, in reality, it is unlikely that an annotator labels or misses a same percentage of targets on every image. Third, the only parameter of an asymmetric mean squared error (AMSE)

needs to be determined in advance. It is related to the drop rate, but the drop rate itself is also difficult to estimate.

7.3 Future Work

This thesis has explored active learning, semi-supervised learning and weakly-supervised learning as a means to reduce the labeling labour and boost machine intelligence. There are other promising domains that can either improve the proposed methods in this thesis or further reduce the dependence on labeled data, in addition to those that have already been mentioned in the discussion sections of the previous chapters.

7.3.1 Open-set Active Learning

In Chapter 3, active learning is applied to a 17-category flower classification task, which is a closed set problem, i.e. the target classes are known in advance. However, in real applications, one may find that the unlabeled set, which contains much more images than the labeled set, consists of both known classes and unknown classes, making the learning process an open-set problem.

Liu and Huang [95] attempt to develop a model for the scenario where the training set contains only unlabeled data with known classes and the test set includes data from both known and unknown classes. They propose to use a diversity-based querying strategy to find the most representative samples to label, and a low-rank representation method to learn the discriminative features that are able to differ novel classes from known classes. Ning et al. [107] also tries to solve the open-set problem but in a different

scenario – the training set is an open set containing both known and unknown classes, while the test set is a closed set including only the known classes. The key to this problem is filtering the samples from unknown classes when querying labels on unlabeled data, as they are often quite uncertain to the model (i.e. easy to be selected by query functions) but can mislead the learning model (because their labels can only be drawn from the known classes). The authors first train a detector with both known unknown classes to model the per-sample max activation value distribution with a mixture of Gaussian models to differ images from known and unknown categories; and then the most uncertain samples of known classes are queried for labels to train a separate classification model.

7.3.2 Improving AMSE with Uncertainty Estimation

In Chapter 6, we propose to use an asymmetric mean squared error (AMSE) to adapt to scenarios of incomplete pixel-level labels. However, the only hyper-parameter of an AMSE, β, needs to be determined by estimating the level of label incompleteness (i.e. drop rates) in advance, which is often difficult. If we can find a way to automatically represent or approximate to the drop rates, it may make the proposed AMSE more effective and robust.

In Chapter 5, we have shown that estimating the uncertainty of the predictions using Monte Carlo Dropout (or aleatoric uncertainty introduced in the discussion section) is useful when adapting FixMatch to regression. Uncertainty estimation can also be incorporated with AMSE by assuming that greater uncertainty corresponds to greater drop rates, and hence to adjust the parameter β accordingly.

7.3.3 Learning the Concept of Repetition

In Chapter 4, 5 and 6, we explore counting tasks, which are important in plant phenotyping. These tasks often require pixel-level labels (e.g. point-annotations) on a large number of, but similar-in-appearance (e.g. from the same category) targets. This makes us wonder if we could train a model to learn the concept of *repetition*. This may boost the development of generic counting models that can automatically count all repeated or similar objects with much fewer labels.

In fact, some inspiring works in an area called class-agnostic counting have been done recently. Class-agnostic counting aims to train a counting model that can potentially count all the targets from any given class, which is often treated as a one-shot or few-shot learning problem, where one or few exemplars of the target class are presented and the model needs to identify the other similar items in an image. The similarity between targets from the same category is explored and matching or clustering methods are often applied to class-agnostic counting. For example, Lu et al. [97] train a generic matching network based on the self-similarity of an image. They consider matching as tracking within an image and use some video tracking datasets to train the model. Yang et al. [161] develop a two-stream ResNet to match the reference images (i.e. exemplars to count) to the targets in an query image in different scales. Godi et al. consider many objects (e.g. cells and LEGO toys) are composed of one or multiple basic shapes (e.g. circles and triangles) and hence pre-train a detection model (e.g. Mask-RCNN [51]) on a dataset consisting of various basic shapes. The pre-trained model is then used to detect objects of arbitrary classes, before grouping (counting) the objects composed of similar shapes using a clustering method. The scenario of incomplete pixel-level labels (discussed



(a) snack of majority: corn

(b) snack of majority: marshmallow

(c) shape of majority: circle

(d) shape of majority: square

Figure 7.1: Example images of the data with mixed snacks (top) and synthetic shapes (bottom). The synthetic shapes are generated in random color and size to force the learning model focus on the contours of the shapes.

in Chapter 6) can also be transferred to a class-agnostic counting problem by treating the few labeled targets as exemplars from the interested class.

During my PhD, we have an unfinished idea to count the *major targets* regardless of their categories in an image. This idea is inspired by the observation that in most counting tasks, the class we want to count dominates in quantity. For example, in aphid counting, aphids appear far more than other insects, and in crowd counting, the number of humans is much greater than the number of cars. In practice, to being exploring this idea, we created a synthetic shape dataset consisting of different num-

bers of circles, squares and triangles (Figure 7.1 bottom) and attempted to train a UNet to segment and count the *dominant shape* in each image; we expected to learn the representation of "majority" and promote it to counting other objects. The early results of this work were encouraging but further exploration is needed to warrant future development. In addition, we also started collecting some real-life images for learning the major class, for example, images of mixed snacks (Figure 7.1 top).

7.3.4 Coarse-grained Supervision

As introduced in the background Section 2.6.2, coarse-grained supervision is an important category of weakly-supervised learning, which attempts to learn a finer-grained task using coarser-grained labels. For example, learn object counting using image-level labels (e.g. the number of counts), instead of pixel-level point-annotations.

For the scenarios having few targets (e.g. wheat spikelet counting, compared to crowd counting), human annotators may quickly estimate the number of targets without clicking on them individually, which is a natural application of coarse-grained supervision. Moreover, it is feasible to only use the binary classification label, which indicates the existence of the desired class, to supervise counting [11].

During my PhD, we ran some exploratory experiments applying class activation mapping (introduced in Section 2.6.2), which is a popular approach in coarse-grained supervision, to count pollen cells. Figure 7.2 shows some images from the pollen cell dataset used. In particular, we first trained a classification model to detect the existence of pollen cells in the images with binary labels, before generating the class activation maps

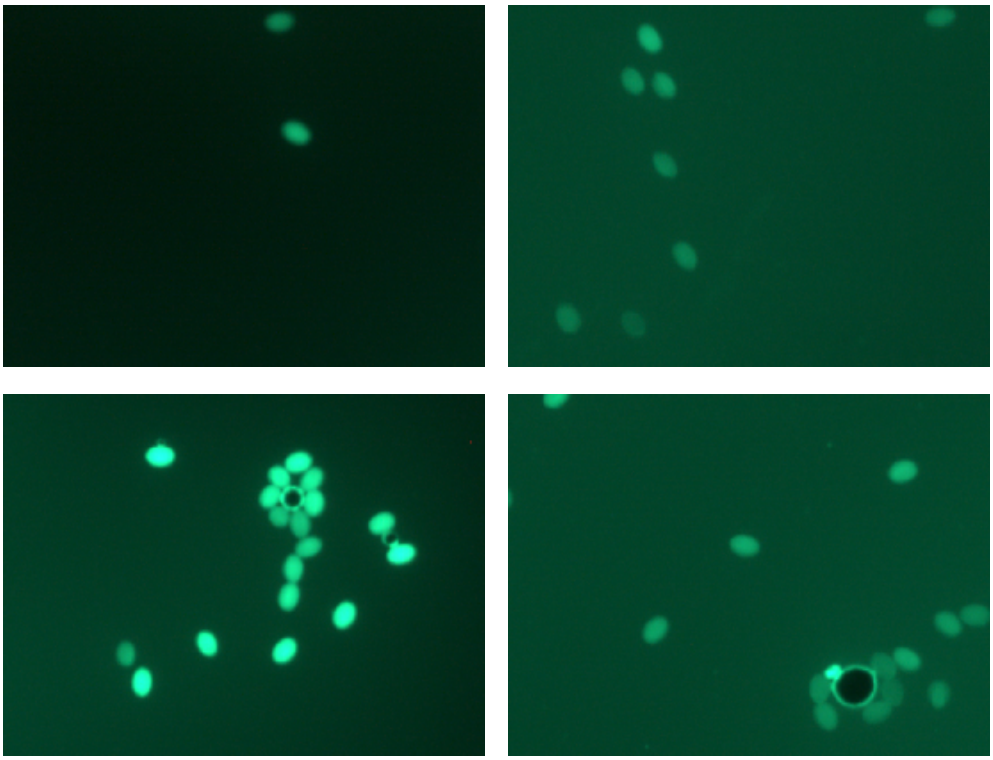


Figure 7.2: Example images of pollen cells (the bright blobs).

of the pollen cells using Grad-CAM++ [21], which is good at highlighting multiple targets from the same category. We then applied some conventional image processing techniques, i.e. thresholding and edge detection, to the class activation maps to obtain bounding boxes on small groups of pollen cells. Figure 7.3 illustrates how we utilize class activation mapping towards pollen cell counting in the coarse-grained supervision manner, where the final counting approach is still under exploration. We expected to perform *unsupervised* counting from the generated bounding boxes; a potential solution is to use a clustering method based on the size of the bounding boxes and the density (e.g. distance between objects) of pollen cells within the individual bounding boxes because they are highly relevant to the number of targets.

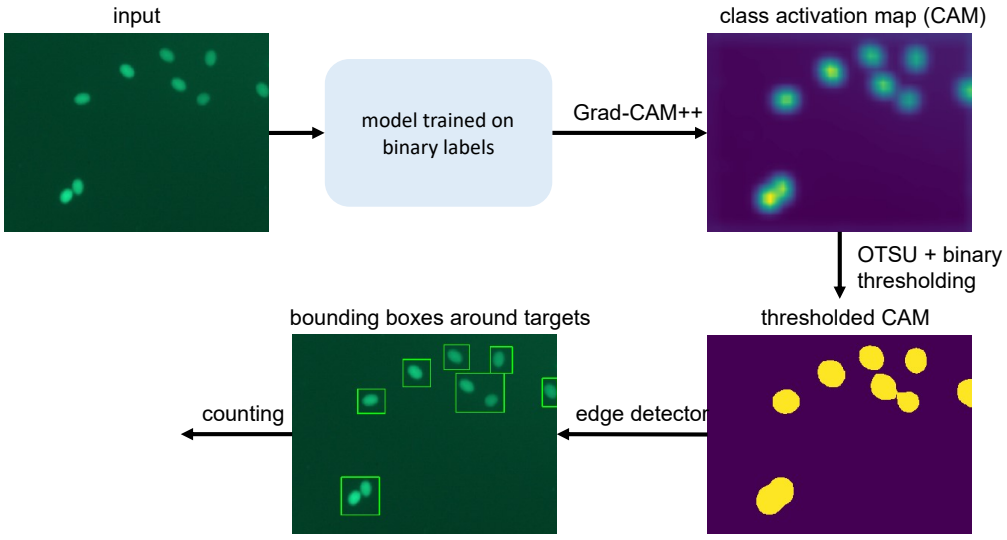


Figure 7.3: The pipeline of weakly-supervised pollen cell counting.

7.4 Conclusions

As the rapid development of deep-learning and computer-vision based plant phenotyping advances, the increasing demands and costs of obtaining high-quality labeled data cannot be neglected.

In this thesis, we examine and develop various approaches, including active learning (Chapter 3), the adaptive Mean Teacher (Chapter 4), MC-FixMatch and Shadowout (Chapter 5), and Asymmetric MSE (AMSE, Chapter 6), to train the learning models with fewer or easier labels to reduce the time and expense of annotating data. Besides, these methods are designed in an adaptive or less-parametric way to reduce the effort of optimizing the learning models. We also discuss some potential improvement (Section 7.3) which may further reduce the dependence on labeled data when developing deep learning models in plant phenotyping.

We hope the findings and the proposed approaches in this thesis can promote the research of more effective and more efficient methods of utilizing data in deep learning for plant phenotyping, as well as the more general

research of machine intelligence.

Bibliography

[1] A. Abuduweili, X. Li, H. Shi, C.-Z. Xu, and D. Dou. Adaptive consistency regularization for semi-supervised transfer learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6923–6932, 2021.

[2] M. Afonso, H. Fonteijn, F. S. Fiorentin, D. Lensink, M. Mooij, N. Faber, G. Polder, and R. Wehrens. Tomato fruit detection and counting in greenhouses using deep learning. *Frontiers in Plant Science*, 11:571299, 2020.

[3] V. B. Alex Kendall and R. Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. In *Proceedings of the British Machine Vision Conference (BMVC)*, page 57. BMVA Press, 2017.

[4] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

[5] D. Angluin. Queries revisited. In *Proceedings of the International Conference on Algorithmic Learning Theory*, pages 12–31. Springer, 2001.

[6] A. Araujo, W. Norris, and J. Sim. Computing receptive fields of convolutional neural networks. *Distill*, 2019. https://distill.pub/2019/computing-receptive-fields.

[7] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, and S. Lacoste-Julien. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

[8] V. N. Balasubramanian, W. Guo, A. L. Chandra, and S. V. Desai. Computer vision with deep learning for plant phenotyping in agriculture: A survey. *Advanced Computing and Communications*, Mar 2020.

[9] S. Bargoti and J. Underwood. Deep fruit detection in orchards. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3626–3633. IEEE, 2017.

[10] E. Bellocchio, T. A. Ciarfuglia, G. Costante, and P. Valigi. Weakly supervised fruit counting for yield estimation using spatial consistency. *IEEE Robotics and Automation Letters*, 4(3):2348–2355, 2019.

[11] E. Bellocchio, T. A. Ciarfuglia, G. Costante, and P. Valigi. Weakly supervised fruit counting for yield estimation using spatial consistency. *IEEE Robotics and Automation Letters*, 4(3):2348–2355, 2019.

[12] K. Bennett, A. Demiriz, et al. Semi-supervised support vector machines. *Advances in Neural Information Processing Systems*, pages 368–374, 1999.

[13] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2001.

[14] A. Blum and T. Mitchell. Combining labeled and unlabeled data

with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT)*, page 92–100, 1998.

[15] K. Brinker. Incorporating diversity in active learning with support vector machines. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, page 59–66. AAAI Press, 2003.

[16] A. B. Chan, Z.-S. J. Liang, and N. Vasconcelos. Privacy preserving crowd monitoring: Counting people without people models or tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–7. IEEE, 2008.

[17] A. B. Chan and N. Vasconcelos. Bayesian poisson regression for crowd counting. In *Proceedings of the IEEE 12th International Conference on Computer Vision (ICCV)*, pages 545–551. IEEE, 2009.

[18] O. Chapelle, B. Scholkopf, and A. Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[19] O. Chapelle, V. Sindhwani, and S. S. Keerthi. Optimization techniques for semi-supervised support vector machines. *Journal of Machine Learning Research*, 9(2), 2008.

[20] O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *International Workshop on Artificial Intelligence and Statistics*, pages 57–64. PMLR, 2005.

[21] A. Chattopadhay, A. Sarkar, P. Howlader, and V. N. Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 839–847. IEEE, 2018.

[22] K. Chen, C. C. Loy, S. Gong, and T. Xiang. Feature mining for localised crowd counting. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2012.

[23] A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pages 215–223, 2011.

[24] D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

[25] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[26] K. Crowston. Amazon mechanical turk: A research tool for organizations and information systems scholars. In A. Bhattacherjee and B. Fitzgerald, editors, *Shaping the Future of ICT Research. Methods and Approaches*, pages 210–221. Springer Berlin Heidelberg, 2012.

[27] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 702–703, 2020.

[28] I. Dagan and S. P. Engelson. Committee-based sampling for training probabilistic classifiers. In *Proceedings of the 12th International Conference on Machine Learning (ICML)*, pages 150–157, 1995.

[29] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[31] J. Deng, O. Russakovsky, J. Krause, M. Bernstein, A. C. Berg, and L. Fei-Fei. Scalable multi-label annotation. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 2014.

[32] T. DeVries and G. W. Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[33] Y.-X. Ding and Z.-H. Zhou. Crowdsourcing with unsure option. *Machine Learning*, 107(4):749–766, 2018.

[34] A. Dobrescu, M. Valerio Giuffrida, and S. A. Tsaftaris. Leveraging multiple datasets for deep leaf counting. In *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 2072–2079, 2017.

[35] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 34(4):743–761, 2011.

[36] Y. Fan, A. Kukleva, and B. Schiele. Revisiting consistency regularization for semi-supervised learning. *arXiv preprint arXiv:2112.05825*, 2021.

[37] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 32(9):1627–1645, 2010.

[38] L. G. Freeman. *Elementary applied statistics*. Wiley, 1965.

[39] Y. Fu, X. Zhu, and B. Li. A survey on instance selection for active learning. *Knowledge and Information Systems*, 35(2):249–283, 2013.

[40] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of The 33rd International Conference on Machine Learning (ICML)*, pages 1050–1059. PMLR, 2016.

[41] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11):2188–2202, 2011.

[42] J. GenÃ©-Mola, V. Vilaplana, J. R. Rosell-Polo, J.-R. Morros, J. Ruiz-Hidalgo, and E. Gregorio. Multi-modal deep learning for fuji apple detection using rgb-d cameras and their radiometric capabilities. *Computers and Electronics in Agriculture*, 162:689–698, 2019.

[43] M. V. Giuffrida, F. Chen, H. Scharr, and S. A. Tsaftaris. Citizen crowds and experts: observer variability in image-based plant phenotyping. *Plant Methods*, 14(1):1–14, 2018.

[44] M. V. Giuffrida, M. Minervini, and S. Tsaftaris. Learning to count leaves in rosette plants. In *Proceedings of the British Machine Vision Conference Workshops (BMVCW)*. BMVA Press, September 2015.

[45] J. Goldberger and E. Ben-Reuven. Training deep neural-networks using a noise adaptation layer. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[46] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27, 2014.

[47] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[48] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020.

[49] T. Guo and G. Li. Improved tri-training with unlabeled data. In *Software Engineering and Knowledge Engineering: Theory and Practice*, pages 139–147. Springer, 2012.

[50] N. Häni, P. Roy, and V. Isler. A comparative study of fruit detection and counting methods for yield mapping in apple orchards. *Journal of Field Robotics*, 37(2):263–282, 2020.

[51] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2961–2969, 2017.

[52] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[53] M. Hein and M. Maier. Manifold denoising. *Advances in Neural Information Processing Systems*, 19, 2006.

[54] S. C. Hoi, R. Jin, J. Zhu, and M. R. Lyu. Batch mode active learning and its application to medical image classification. In *Proceedings*

of the 23rd International Conference on Machine Learning (ICML), pages 417–424, 2006.

[55] N. Häni, P. Roy, and V. Isler. Minneapple: A benchmark dataset for apple detection and segmentation. *IEEE Robotics and Automation Letters*, 5(2):852–858, 2020.

[56] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 448–456. PMLR, 2015.

[57] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 2304–2313. PMLR, 2018.

[58] Y. Jiang and C. Li. Convolutional neural networks for image-based high-throughput plant phenotyping: a review. *Plant Phenomics*, 2020, 2020.

[59] T. Joachims et al. Transductive inference for text classification using support vector machines. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 99, pages 200–209, 1999.

[60] A. Kamilaris and F. X. Prenafeta-Boldu. Deep learning in agriculture: A survey. *Computers and Electronics in Agriculture*, 147:70–90, 2018.

[61] M. Kampffmeyer, A.-B. Salberg, and R. Jenssen. Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1–9, 2016.

[62] Z. Ke, D. Wang, Q. Yan, J. Ren, and R. W. Lau. Dual student: Breaking the limits of the teacher in semi-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6728–6736, 2019.

[63] A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in Neural Information Processing Systems*, 30, 2017.

[64] R. Kestur, A. Meduri, and O. Narasipura. Mangonet: A deep semantic segmentation architecture for a method to detect and count mangoes in an open orchard. *Engineering Applications of Artificial Intelligence*, 77:59–69, 2019.

[65] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[66] D. P. Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling. Semi-supervised learning with deep generative models. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[67] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[68] J. Kremer, K. Steenstrup Pedersen, and C. Igel. Active learning with support vector machines. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(4):313–326, 2014.

[69] V. Krishnamurthy. Algorithms for optimal scheduling and management of hidden markov model sensors. *IEEE Transactions on Signal Processing*, 50(6):1382–1397, 2002.

[70] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, Toronto, Ontario, 2009.

[71] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[72] D. Krueger, N. Ballas, S. Jastrzebski, D. Arpit, T. Maharaj, E. Bengio, A. Fischer, and A. Courville. Deep nets don't learn via memorization. In *Proceedings of the International Conference on Learning Representations Workshops (ICLRW)*, 2017.

[73] P. Kumar and A. Gupta. Active learning query strategies for classification, regression, and clustering: a survey. *Journal of Computer Science and Technology*, 35(4):913–945, 2020.

[74] S. Laine and T. Aila. Temporal ensembling for semi-supervised learning. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

[75] K. Lang and E. Baum. Query learning can work poorly when a human oracle is used. In *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*. IEEE Press, 1992.

[76] F. Lateef and Y. Ruichek. Survey on semantic segmentation using deep learning techniques. *Neurocomputing*, 338:321–348, 2019.

[77] V. Lempitsky and A. Zisserman. Learning to count objects in images. *Advances in Neural Information Processing Systems*, 23, 2010.

[78] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the ACM International Conference*

on Research and Development in Information Retrieval SIGIR, pages 3–12. Springer, 1994.

[79] C. Li, K. Xu, J. Zhu, J. Liu, and B. Zhang. Triple generative adversarial networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[80] J. Li, Y. Wong, Q. Zhao, and M. S. Kankanhalli. Learning to learn from noisy labeled data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[81] M. Li, Z. Zhang, K. Huang, and T. Tan. Estimating the number of people in crowded scenes by mid based foreground segmentation and head-shoulder detection. In *Proceedings of the 19th International Conference on Pattern Recognition (ICPR)*, pages 1–4. IEEE, 2008.

[82] S. Li, W. Deng, and J. Du. Reliable crowdsourcing and deep locality-preserving learning for expression recognition in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2852–2861, 2017.

[83] S. Li, W. Deng, and J. Du. Reliable crowdsourcing and deep locality-preserving learning for expression recognition in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[84] S. Li, W.-T. Li, and W. Wang. Co-gcn for multi-view semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4691–4698, 2020.

[85] Y. Li, X. Zhang, and D. Chen. Csrnet: dilated convolutional neural networks for understanding the highly congested scenes. In *Pro-*

ceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1091–1100, 2018.

[86] Y.-F. Li, I. W. Tsang, J. T. Kwok, and Z.-H. Zhou. Convex and scalable weakly labeled svms. *Journal of Machine Learning Research*, 14(7), 2013.

[87] Z. Li, L. Zhang, Y. Fang, J. Wang, H. Xu, B. Yin, and H. Lu. Deep people counting with faster r-cnn and correlation tracking. In *Proceedings of the International Conference on Internet Multimedia Computing and Service*, pages 57–60, 2016.

[88] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[89] S.-F. Lin, J.-Y. Chen, and H.-X. Chao. Estimation of number of people in crowded scenes using perspective transformation. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 31(6):645–654, 2001.

[90] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken, and C. I. Sanchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, 2017.

[91] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen. Deep learning for generic object detection: A survey. *International Journal of Computer Vision (IJCV)*, 128(2):261–318, 2020.

[92] W. Liu, M. Salzmann, and P. Fua. Context-aware crowd counting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[93] X. Liu, J. Van De Weijer, and A. D. Bagdanov. Leveraging unlabeled data for crowd counting by learning to rank. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7661–7669, 2018.

[94] Y. Liu, L. Liu, P. Wang, P. Zhang, and Y. Lei. Semi-supervised crowd counting via self-training on surrogate tasks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 242–259. Springer, 2020.

[95] Z.-Y. Liu and S.-J. Huang. Active sampling for open-set classification without initial annotation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4416–4423, 2019.

[96] C. C. Loy, K. Chen, S. Gong, and T. Xiang. Crowd counting and profiling: Methodology and evaluation. In *Modeling, Simulation and Visual Analysis of Crowds*, pages 347–382. Springer, 2013.

[97] E. Lu, W. Xie, and A. Zisserman. Class-agnostic counting. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, pages 669–684. Springer, 2018.

[98] Y. Lu and S. Young. A survey of public datasets for computer vision tasks in precision agriculture. *Computers and Electronics in Agriculture*, 178:105760, 2020.

[99] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *Proceedings of the Annual Conference on Neural Information Processing Systems Workshops (NeurIPSW)*, 2011.

[100] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for

human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2016.

[101] N.-V. Nguyen, C. Rigaud, A. Revel, and J.-C. Burie. A learning approach with incomplete pixel-level labels for deep neural networks. *Neural Networks*, 130:111–125, Oct. 2020.

[102] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the 9th International Conference on Information and Knowledge Management*, pages 86–93, 2000.

[103] M.-E. Nilsback and A. Zisserman. A visual vocabulary for flower classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1447–1454, 2006.

[104] M.-E. Nilsback and A. Zisserman. Delving into the whorl of flower segmentation. In *Proceedings of the British Machine Vision Conference (BMVC)*, volume 1, pages 570–579, 2007.

[105] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

[106] M.-E. Nilsback and A. Zisserman. Delving deeper into the whorl of flower segmentation. *Image and Vision Computing*, 2009.

[107] K.-P. Ning, X. Zhao, Y. Li, and S.-J. Huang. Active learning for open-set annotation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 41–49, 2022.

[108] C. G. Northcutt, L. Jiang, and I. L. Chuang. Confident learning:

Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research*, 2021.

[109] N. Paragios and V. Ramesh. A mrf-based approach for real-time subway monitoring. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–I. IEEE, 2001.

[110] V.-Q. Pham, T. Kozakaya, O. Yamaguchi, and R. Okada. Count forest: Co-voting uncertain number of targets using random forest for crowd density estimation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 3253–3261, 2015.

[111] M. P. Pound, J. A. Atkinson, D. M. Wells, T. P. Pridmore, and A. P. French. Deep learning for multi-task plant phenotyping. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2017.

[112] T. Qian, B. Liu, L. Chen, and Z. Peng. Tri-training for authorship attribution with limited training data. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 345–351, 2014.

[113] H. G. Ramaswamy et al. Ablation-cam: visual explanations for deep convolutional network via gradient-free localization. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 983–991, 2020.

[114] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *Journal of Machine Learning Research*, 11(4), 2010.

[115] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, B. B. Gupta, X. Chen,

and X. Wang. A survey of deep active learning. *ACM Computing Surveys (CSUR)*, 54(9):1–40, 2021.

[116] M. N. Rizve, K. Duarte, Y. S. Rawat, and M. Shah. In defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning. *arXiv preprint arXiv:2101.06329*, 2021.

[117] O. Russakovsky, J. Deng, Z. Huang, A. C. Berg, and L. Fei-Fei. Detecting avocados to zucchinis: what have we done, and where are we going? In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2013.

[118] 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 (IJCV)*, 115(3):211–252, 2015.

[119] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: a database and web-based tool for image annotation. *International Journal of Computer Vision (IJCV)*, 77(1-3):157–173, 2008.

[120] D. Ryan, S. Denman, C. Fookes, and S. Sridharan. Crowd counting using multiple local features. In *Proceedings of the International Conference on Digital Image Computing: Techniques and Applications*, pages 81–88. IEEE, 2009.

[121] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, 2017.

[122] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In

Proceedings of the 5th Annual Workshop on Computational Learning Theory (COLT), pages 287–294, 1992.

[123] N. B. Shah and D. Zhou. Double or nothing: Multiplicative incentive mechanisms for crowdsourcing. *Advances in Neural Information Processing Systems*, 28:1–9, 2015.

[124] C. E. Shannon. *A Mathematical Theory of Communication*, volume 27. Bell System Technical Journal, 1948.

[125] R. Sheikhpour, M. A. Sarram, S. Gharaghani, and M. A. Z. Chahooki. A survey on semi-supervised feature selection methods. *Pattern Recognition*, 64:141–158, 2017.

[126] V. S. Sheng, F. Provost, and P. G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 614–622, 2008.

[127] M. Shi, Z. Yang, C. Xu, and Q. Chen. Revisiting perspective information for efficient crowd counting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7279–7288, 2019.

[128] W. Shi, R. van de Zedde, H. Jiang, and G. Kootstra. Plant-part segmentation using deep learning and multi-view vision. *Biosystems Engineering*, 187:81–95, 2019.

[129] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[130] V. A. Sindagi and V. M. Patel. A survey of recent advances in cnn-

based single image crowd counting and density estimation. *Pattern Recognition Letters*, 107:3–16, 2018.

[131] R. Snow, B. O'connor, D. Jurafsky, and A. Y. Ng. Cheap and fast–but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 254–263, 2008.

[132] K. Sohn, D. Berthelot, N. Carlini, Z. Zhang, H. Zhang, C. A. Raffel, E. D. Cubuk, A. Kurakin, and C.-L. Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in Neural Information Processing Systems*, 33:596–608, 2020.

[133] J. T. Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015.

[134] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[135] H. Su, X. Shi, J. Cai, and L. Yang. Local and global consistency regularized mean teacher for semi-supervised nuclei classification. In *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 559–567. Springer, 2019.

[136] A. Subramanya and P. P. Talukdar. Graph-based semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(4):1–125, 2014.

[137] S. Sun. A survey of multi-view machine learning. *Neural Computing and Applications*, 23(7):2031–2038, 2013.

[138] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[139] H. Tao, C. Hou, F. Nie, J. Zhu, and D. Yi. Scalable multi-view semi-supervised classification via adaptive regression. *IEEE Transactions on Image Processing*, 26(9):4283–4296, 2017.

[140] A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*, page 1195–1204. Curran Associates Inc., 2017.

[141] A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *arXiv preprint arXiv:1703.01780*, 2017.

[142] N. Teimouri, M. Dyrmann, P. R. Nielsen, S. K. Mathiassen, G. J. Somerville, and R. N. Jørgensen. Weed growth stage estimator using deep convolutional neural networks. *Sensors*, 18(5), 2018.

[143] I. Triguero, S. García, and F. Herrera. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information Systems*, 42(2):245–284, 2015.

[144] J. E. Van Engelen and H. H. Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, 2020.

[145] V. N. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999.

[146] J. W. Vaughan. Making better use of the crowd: how crowdsourcing can advance machine learning research. *Journal of Machine Learning Research*, 18(1):7026–7071, 2017.

[147] F. Wang and C. Zhang. Label propagation through linear neighborhoods. *IEEE Transactions on Knowledge and Data Engineering*, 20(1):55–67, 2007.

[148] H. Wang, Z. Wang, M. Du, F. Yang, Z. Zhang, S. Ding, P. Mardziel, and X. Hu. Score-cam: Score-weighted visual explanations for convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 24–25, 2020.

[149] H. Wang and D.-Y. Yeung. A survey on bayesian deep learning. *ACM Computing Surveys (CSUR)*, 53(5):1–37, 2020.

[150] J. Wang, R. Xiao, Y. Guo, and L. Zhang. Learning to count objects with few exemplar annotations. *arXiv preprint arXiv:1905.07898*, 2019.

[151] W. Wang, Y. Yang, X. Wang, W. Wang, and J. Li. Development of convolutional neural network and its application in image classification: a survey. *Optical Engineering*, 58(4):040901, 2019.

[152] W. Wang and Z.-H. Zhou. Analyzing co-training style algorithms. In *Proceedings of the European Conference on Machine Learning*, pages 454–465. Springer, 2007.

[153] W. Wang and Z.-H. Zhou. A new analysis of co-training. In *Proceed-*

ings of the International Conference on Machine Learning (ICML), 2010.

[154] Y. Wang, J. Peng, and Z. Zhang. Uncertainty-aware pseudo label refinery for domain adaptive semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9092–9101, 2021.

[155] B. Wu and R. Nevatia. Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. *International Journal of Computer Vision (IJCV)*, 75(2):247–266, 2007.

[156] R. Xiang. Image segmentation for whole tomato plant recognition at night. *Computers and Electronics in Agriculture*, 154:434–442, 2018.

[157] C. Xie and C. Yang. A review on plant high-throughput phenotyping traits using uav-based sensors. *Computers and Electronics in Agriculture*, 178:105731, 2020.

[158] J. Xu, H. He, and H. Man. Dcpe co-training for classification. *Neurocomputing*, 86:75–85, 2012.

[159] Z. Xu, R. Akella, and Y. Zhang. Incorporating diversity and density in active learning for relevance feedback. In *Proceedings of the European Conference on Information Retrieval (ECIR)*, pages 246–257. Springer, 2007.

[160] J. Yang, X. Zhao, and Z. Li. Crowdsourcing indoor positioning by light-weight automatic fingerprint updating via ensemble learning. *IEEE Access*, 7:26255–26267, 2019.

[161] S.-D. Yang, H.-T. Su, W. H. Hsu, and W.-C. Chen. Class-agnostic few-shot object counting. In *Proceedings of the IEEE/CVF Winter*

Conference on Applications of Computer Vision (WACV), pages 870–878, 2021.

[162] Y. Yang, G. Li, Z. Wu, L. Su, Q. Huang, and N. Sebe. Weakly-supervised crowd counting learns from sorting rather than locations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 1–17. Springer, 2020.

[163] M.-C. Yuen, I. King, and K.-S. Leung. A survey of crowdsourcing systems. In *Proceedings of the IEEE 3rd International Conference on Privacy, Security, Risk and Trust and the IEEE 3rd International Conference on Social Computing*, pages 766–773. IEEE, 2011.

[164] R. Zemel and M. Carreira-Perpiñán. Proximity graphs for clustering and manifold learning. *Advances in Neural Information Processing Systems*, 17, 2004.

[165] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

[166] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[167] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma. Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 589–597, 2016.

[168] B. Zhao, X. Wu, Z.-Q. Cheng, H. Liu, Z. Jie, and J. Feng. Multi-view

image generation from a single-view. In *Proceedings of the 26th ACM International Conference on Multimedia*, pages 383–391, 2018.

[169] J. Zhong, K. Tang, and Z.-H. Zhou. Active learning from crowds with unsure option. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[170] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2921–2929, 2016.

[171] Z.-H. Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 2018.

[172] Z.-H. Zhou and M. Li. Tri-training: exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1529–1541, 2005.

[173] Z.-H. Zhou and M. Li. Semi-supervised learning by disagreement. *Knowledge and Information Systems*, 24(3):415–439, 2010.

[174] S. Zhu, X. Sun, and D. Jin. Multi-view semi-supervised learning for image classification. *Neurocomputing*, 208:136–142, 2016.

[175] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

[176] X. Zhu and A. B. Goldberg. Introduction to semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 3(1):1–130, 2009.

[177] Y. Zou, Z. Yu, X. Liu, B. Kumar, and J. Wang. Confidence regularized self-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5982–5991, 2019.