Unsupervised Domain Adaptation for Object Counting

A Thesis Submitted to the
College of Graduate and Postdoctoral Studies
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Tewodros Wondifraw Ayalew

Permission to Use

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
176 Thorvaldson Building
110 Science Place
University of Saskatchewan
Saskatoon, Saskatchewan
Canada
S7N 5C9

Or

Dean
College of Graduate and Postdoctoral Studies
University of Saskatchewan
116 Thorvaldson Building, 110 Science Place
Saskatoon, Saskatchewan S7N 5C9
Canada

Abstract

Supervised learning is a common approach for counting objects in images, but for counting small, densely located objects, the required image annotations are burdensome to collect. Counting plant organs for image-based plant phenotyping and crowd counting fall within this category. Object counting in plant images is further challenged by having plant image datasets with significant domain shift due to different experimental conditions, e.g. applying an annotated dataset of indoor plant images for use on outdoor images, or on a different plant species. Learning to count from synthetic data to apply the knowledge in real-world data is also another important domain shift addressed in crowd counting tasks where getting annotations for real-world images, especially for highly crowded images, is tedious and potentially inaccurate.

In this thesis, we propose a domain-adversarial learning approach for domain adaptation of density map estimation for the purposes of object counting. We took a fully convolutional network — initially designed for image segmentation — and trained it to count objects via density estimation from images sampled from a distribution and in parallel adapted the knowledge to a related counting task without the need of annotations. The proposed approach does not assume perfectly aligned distributions between the source and target datasets, which makes it more broadly applicable within general object counting and plant organ counting tasks. Evaluation on three diverse object counting tasks (wheat spikelets, leaves, crowd) demonstrate consistent performance on the target datasets across different classes of domain shift: from indoor-to-outdoor images, from species-to-species, and from synthetic-to-real.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Dr. Ian Stavness. Without his supervision, this work would not have been possible. His insights and advice have been invaluable for this work and for all the other research projects I was involved in throughout the program. I am grateful for his constant support and guidance. Accept my heartfelt gratitude for your time, support, and patience.

My sincere thanks also go to Dr. Kevin Stanley. I have been fortunate to have had the pleasure of taking his course. And I have learned a lot from his lectures as well as his project supervision. His challenging questions allowed me to see things from different perspectives and encouraged me to explore more. I am grateful for his valuable lessons, which I will use for the rest of my career.

I would also like to thank my friends and my labmates who have been my support system since the first day I joined the graduate program: from winter jacket to course project advice. Thank you for making this journey fun. I would like to give very special thanks to Dr. Jordan Ubbens for tirelessly answering my questions on a daily basis. I am thankful for his constant help.

Finally, I would like to thank my mom, dad, and uncles for their unwavering support. I couldn't have made it this far without their unconditional love and continuous encouragement.

To my loving parents, Tiru Tegegne Feleke and Wondifraw Ayalew Misikir

To my uncles, Worku Tegegne Feleke and Wondimagegn Chekol, and their families

Contents

List of Tables

List of Figures

List of Abbreviations

CVPPP	Computer Vision Problems in Plant Phenotyping
CNN	Convolutional Neural Networks
FCN	Fully Convolutional Network
GAN	Generative Adversarial Network
TL	Transfer Learning
DA	Domain Adaptation
FPN	Feature Pyramid Network
MSE	Mean Squared Error
MAE	Mean Absolute Error
RMSE	Root Mean Squared Error
LSTN	Locality-constrained Spatial Transformer Network
LST	Locality-constrained Spatial Transformer
FDST	Fudan-ShanghaiTech
CoGAN	Coupled Generative Adversarial Network
ADDA	Adversarial Discriminative Domain Adaptation
NLT	Neuron Linear Transformation
SFCN	Spatial Fully Convolutional Network
DANN	Domain-Adversarial Neural Network
GRL	Gradient Reversal Layer
GWD	Global Wheat Head Dataset
GCC	GTA5 Crowd Counting
XE	Cross-entropy
LS	Least Square

Publications

Parts of this thesis have been published in:

Tewodros Ayalew, Jordan Ubbens, and Ian Stavness, "Unsupervised Domain Adaptation For Plant Organ Counting", Computer Vision Problems in Plant Phenotyping, European Conference on Computer Vision (ECCV), August 2020

Author Contributions

TA designed and implemented the architecture and the method, designed the experiments, implemented the hyperparameter search and tuned the hyperparameters, generated ground truth for the experiments, annotated ground truth for the ACID to GWD spikelet counting task, trained the model for all the experiments, evaluated the method and the baseline models, and wrote the paper.

JU edited the paper and advised the direction of some of the experiments.

IS supervised and guided in designing the experiments and the method, wrote parts of the paper, and edited the paper.

1 Introduction

1.1 Motivation

Object counting is an important task in computer vision with a wide range of applications, including counting the number of people in a crowd [49, 98, 48, 76], the number of cars on a street [87, 115], and the number of cells in a microscopy image [105, 120, 72]. Object counting is a highly relevant task in image-based plant phenotyping, notably for counting plants in the field to estimate the rate of seedling emergence, and counting plant organs to estimate traits relevant for selection in crop breeding programs. For example, counting spikes or heads in cereal crops is a relevant trait for estimating yield [74, 59, 2], counting flowers for estimating the start and duration of flowering is relevant for demarcating plant growth stages [52, 23], and counting leafs and tillers is a relevant trait to assess plant health [1, 31]. Leaf counting, in particular, has been a seminal plant phenotyping task, thanks to the widely utilized CVPPP leaf counting challenge dataset [65, 64].

Convolutional neural networks (CNN) provide state-of-the-art performance for object counting tasks. Supervised learning is most common in previous work, but object detection [50, 48] and density estimation [49, 68, 59] approaches both require fairly tedious annotation of training images with either bounding boxes or dots centered on each object instance. In the context of plant phenotyping, plant organ objects are often small and densely packed making the annotation process even more laborious. In addition, unlike general objects which can be annotated reliably by any individual, identifying and annotating plant organs in images often requires specialized training and experience in plant science [29, 118]. This makes it difficult to obtain large annotated plant image datasets.

Another challenge for computer vision tasks in plant phenotyping is that, unlike large image datasets of general objects, plant image datasets usually include highly self-similar images with a small amount of variation among images within the dataset. An individual plant image dataset is often acquired under similar conditions (single crop type, same field) and therefore trying to directly use a CNN trained on a single dataset to new images from a different crop, field, or growing season will likely fail. A model trained to count objects on one dataset (source dataset) will not perform well on a different dataset (target dataset) when these datasets have different prior distributions. This challenge is generally called *domain-shift*. An extreme case of domain shift in plant phenotyping is using a source dataset of plant images collected in an indoor controlled environment, with individual plants with controlled lighting on a blank background, which can be more easily annotated, and attempting to apply the model to a target dataset of outdoor field images with multiple, overlapping plants with variable lighting and backgrounds, and blur due to wind motion.

1.2 Proposed solution

Domain-shift is often handled by fine-tuning a model, initially trained on a source dataset, with samples taken from a target dataset. Fine-tuning, however, requires the existence of some annotated images in the target dataset. Annotating sufficient data to fine-tune models is an expensive and time-consuming task. A second approach commonly used to solve this problem is domain adaptation. Domain adaptation techniques typically try to align the source and target data distributions [94]. In most cases, domain adaptation techniques do not require the existence of annotated data in the target domain or require much less annotation than needed for fine-tuning techniques.

In this thesis, we propose a method that applies an unsupervised domain adaptation technique, first proposed for image classification [24], to jointly train a CNN to count objects from images with dot annotations (i.e., source domain) and adapt this knowledge to related sets of images (i.e., target domain) where labels are absent. We modeled the object counting problem as a density map estimation problem. We evaluated our proposed domain adaptation method on three object counting tasks (spikelet counting, leaf counting and crowd counting) each with a different challenge. The wheat spikelet counting task adapts from indoor images to outdoor images and presents challenges due to self-similarity, self-occlusion, and appearance variation [74, 2]. The leaf counting task adapts from one plant species to a different plant species and presents variability in shape and size, overlapping and occlusion [44]. Crowd counting tasks present challenges such as variation in background, occlusion, and variation in scale [84]. Results show consistent improvements over the baseline models and comparable results to previous domain adaptation work in leaf counting.

1.3 Contributions

The contributions of this thesis include:

1. The extension of an unsupervised domain adaptation method, initially designed for classification tasks, for density map estimation that learns in the source domain and adapts the knowledge to the target domain in each step of a training.

2. The evaluation of our method on three diverse counting tasks with different domain-shift. From indoor-to-outdoor, from species-to-species, and from synthetic-to-real-world domain adaptation on object counting tasks

3. A new public dataset of annotated wheat spikelets imaged in outdoor field conditions. We made dot annotations representing spikelet positions. This dataset can be used for training and testing machine learning models for wheat spikelet localization and counting tasks.

4. A baseline model for a new domain adaptation task: Domain adaptation in wheat spikelet counting

5. Detailed discussion on our proposed architecture, how hyperparameters were selected, and how we trained the model.

6. Detailed analysis of the results and discussions on the drawbacks of our proposed method

1.4 Organization of Thesis

This thesis is organized as follows:

- **Chapter 2**: provides essential background information on the topics that help to understand the rest of the thesis: Convolutional Neural Networks, Fully Convolutional Networks, object counting, Adversarial learning, and Deep transfer learning.

- **Chapter 3**: presents related works where we review important papers in spikelet, leaf, and crowd counting. Additionally, we also reviewed publications on domain adaptation tasks in plant phenotyping and crowd counting tasks.

- **Chapter 4**: provides a detailed explanation of our method, including our proposed architecture, the cost functions used to train the model, and the training process. We also describe the designed experiments and the dataset we used for these experiments.

- **Chapter 5**: outlines the results of our experiments from evaluating our proposed method and the baseline models. Furthermore, we compared our method with previously proposed methods for those experiments with existing related work.

- **Chapter 6**: provides a detailed discussion of the results. After that, we present the limitations and potential future works following our work.

- **Chapter 7**: summarizes this thesis and presents its contributions.

2 Background

This chapter presents the background concepts used to develop our proposed method. Section 2.1 gives a general overview of Convolutional Neural Networks and their building blocks. The following section, Section 2.2, discusses the need and the design principles of Fully Convolutional Networks. Section 2.3 describes the object counting problem and prominent deep-learning-based solutions. Section 2.4 defines the adversarial learning framework by discussing the pioneering work in deep generative learning. Finally, section 2.5 presents the problems associated with training deep learning models in the absence of an extensive training dataset and how they are approached.

2.1 Convolutional Neural Networks

LeCun et al. [47] defined Deep learning as a representation learning technique used to train multi-layered computational models. Typically, these computational models are composed of layers of non-linear functions. Deep learning has shown state-of-the-art performance in a broad spectrum of learning tasks particularly in domains such as computer vision [83, 43, 35, 40] and natural language understanding [73, 63, 96].

Convolutional Neural Networks (CNNs) [45] are a particular kind of deep learning models that are typically used for image or video analysis. CNNs are built from layers designed to learn hierarchical visual features in various levels of abstraction. Basic CNN architectures are usually composed of Convolutional layers, Pooling layers, Fully-Connected layers, and Non-Linear activations.

2.1.1 Convolutional Layer

Convolutional layers are the main components of a CNN. These layers are designed to perform a linear operation called *convolution* in order to extract features. The convolution of a function f with another function g is defined as:

$$(f * g)(t) = \int f(\tau)g(t-\tau)d\tau \tag{2.1}$$

where $*$ represents the convolution operation. For discrete functions

$$(f * g)(t) = \sum_{\tau=-\infty}^{\infty} f(\tau)g(t-\tau) \tag{2.2}$$

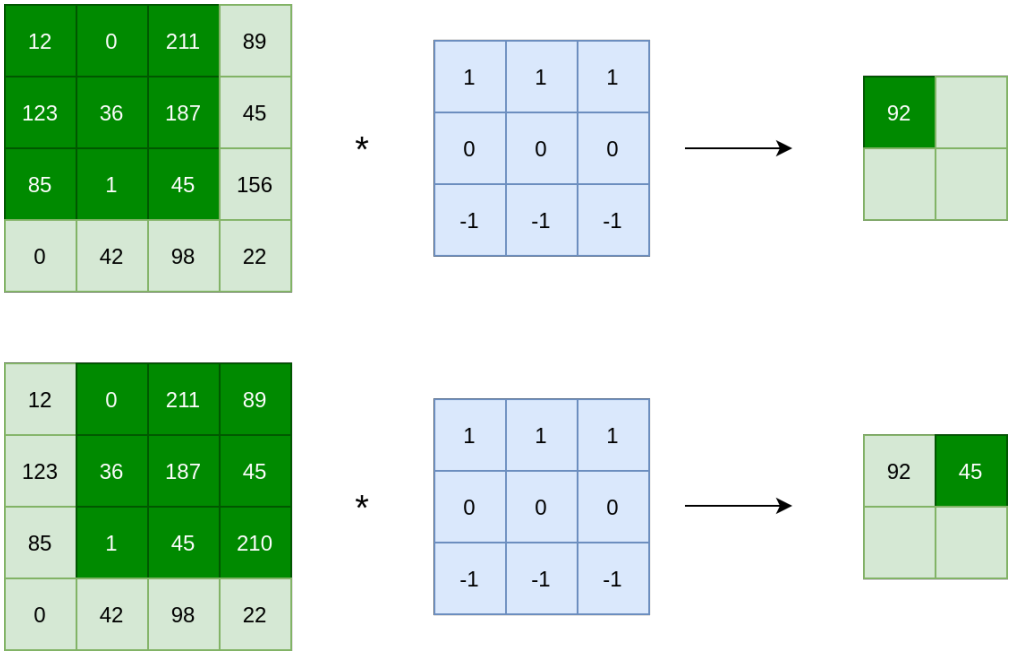

Figure 2.1: Convolution operation between a 4×4 image with a 3×3 kernel with a stride of 1

which is extended to the two dimensional case as

$$(f * g)(x, y) = \sum_i \sum_j g(i, j) \cdot f(x - i, y - j) \tag{2.3}$$

In image-based tasks, f represents a pixel intensity of an input image, and g represents a 2D matrix called a *kernel* or *filter*. Convolution of an image with a filter is performed by sliding the filter over the image and applying elementwise multiplication between the image's submatrix aligned with the kernel and the kernel itself followed by summing over the results. The step size defining the number of pixel a kernel slides while convolving an image is defined by the *stride*. Figure 2.1 provides an example for the convolution operation between a 4×4 image with a 3×3 kernel with a stride of 1. As can be seen in the example, the size of the convolution operation is smaller than the input. Generally, given an input of shape $I_h \times I_w$, a kernel size $k_h \times k_w$ and stride of s the output shape is computed as:

$$\left\lfloor \frac{I_h - k_h}{s} \right\rfloor + 1 \times \left\lfloor \frac{I_w - k_w}{s} \right\rfloor + 1. \tag{2.4}$$

The edges of the input image are padded — commonly with zeroes (i.e., *zero-padding*) — to get an output with a similar size as input.

Usually, kernels are small $k \times k$ square matrices employed to extract features (e.g., edges) from images while accounting for the spatial (neighborhood) dependencies. These spatial dependencies are encoded via

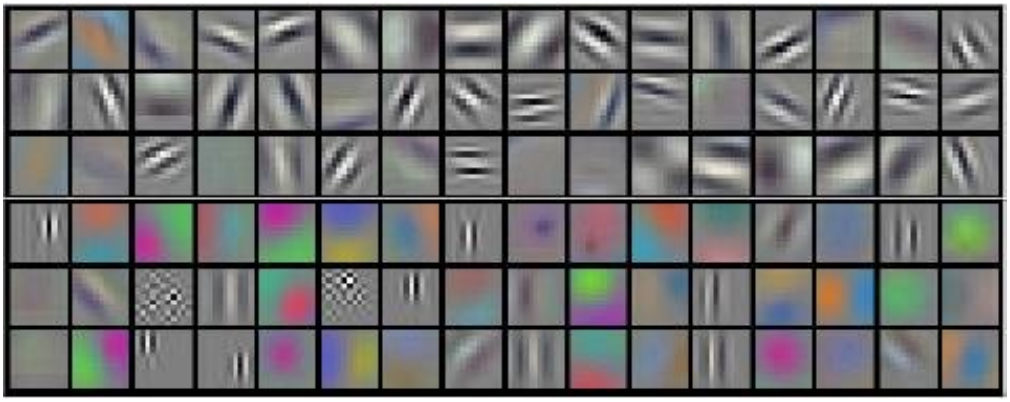

Figure 2.2: Visualization of kernel weights from AlexNet's first convolutional layer trained on the ImageNet [79] dataset. Figure taken from Krizhevsky et al. [43]

the elements of the matrix forming the kernel, which are used as weights in the convolution operation. In traditional image-processing, the elements of a kernel are predetermined to perform specific tasks such as image denoising and edge detection. For instance, the Sobel filter is designed to detect edges from images. The horizontal and vertical Sobel filters are represented as:

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

where the matrix on the left is designed to detect horizontal edges from images whereas the one on the right is for vertical edges.

Even though there are several preexisting kernels designed to extract various features from images, complex computer vision tasks — such as object detection and classification — require a combination of several kernels. Handcrafting these kernels is a cumbersome task. Additionally, selecting an optimum combination of kernels to extract important features for each image is difficult [71]. To tackle these problems, CNNs use layers of trainable kernels called *Convolutional layers*. Each convolutional layer is composed of a set of n kernels. The architecture of a CNN defines the number and size of these kernels. However, the weights are discovered from data via a training process.

In the context of deep learning, training a model is tasked to find parameters θ such that the model, $\mathcal{G}$, can approximate the mapping between a set of input-output pairs $\{(x_i, y_i)\}_{i=1}^{n}$, $\mathcal{G}(x;\theta) = y$. These optimizations are performed by trying to minimize errors computed via a prespecified function called a *Cost function (a.k.a Objective function or Loss function)*. An objective function evaluates a model's performance by computing the error between the model's predicted values and ground-truth values. In each step of a training loop, the parameters are updated by the gradient of the cost function with respect to the weights computed via the back-propagation algorithm [78]. Figure 2.2 shows a visualization of learned kernel weights taken from AlexNet's [43] first convolutional layer — an early important CNN models in the deep learning literature —

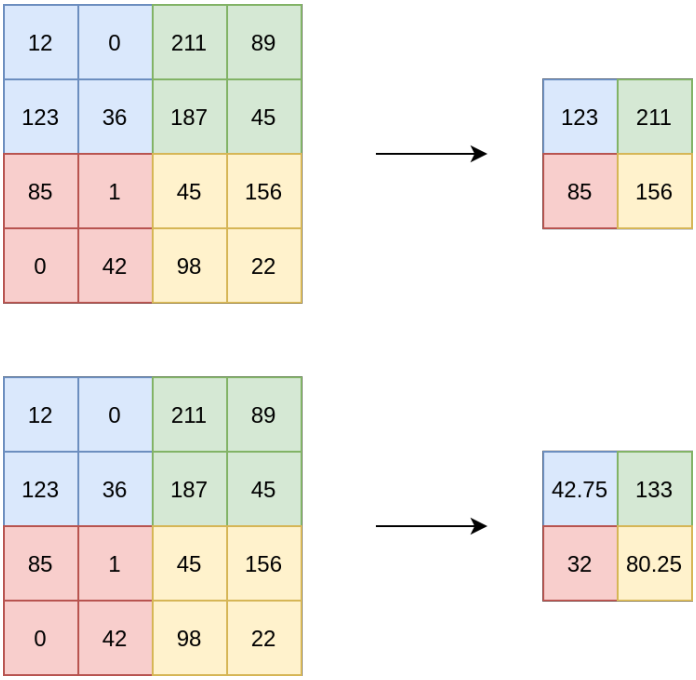

Figure 2.3: Example (Top) max-pooling and (Bottom) average-pooling operations using a 4×4 image and a 2×2 kernel with a stride of 2

after training it using the ImageNet [79] dataset.

1×1 convolutions are a special kind of convolutional layers that use 1×1 kernels. This layers are used for dimensinality reduction instead of feature extraction. Applying a 1×1 kernel results in the same spatial resolution output but with a reduced depth, i.e number of channels. Szegedy et al. [85] showed that 1×1 kernels can be used to compress an input volume before expensive convolutional operations in order to learn efficient data representations.

2.1.2 Pooling Layer

Typically, outputs from convolutional layers — called *Feature Maps or Activation Maps* — are passed to pooling layers. The *Pooling layer* is designed to reduce the spatial resolution of activation maps using a pre-defined downsampling operation. The downsampling operations, in turn, reduces the number of parameters in CNNs. Commonly used downsampling operations are max- and average- pooling. Max-pooling is more commonly used because it preserves details that are important for the discriminative ability of a CNN [81]. The max-pooling layer selects the maximum pixel value with a $k \times k$ kernel. On the other hand, the average pooling takes the mean of the pixel values. Figure 2.3 shows an example of how the two pooling layers estimate an output given the same input.

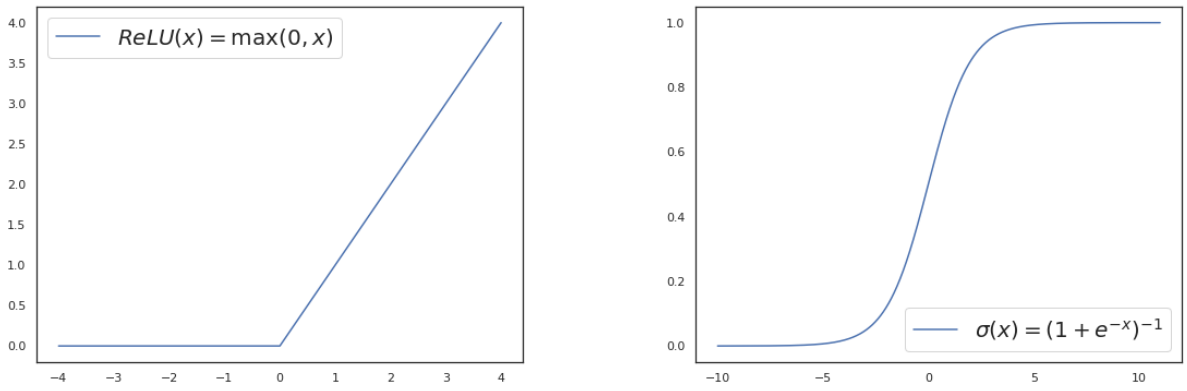

Figure 2.4: Visualization of the (left) ReLU and (right) Sigmoid activation functions

2.1.3 Fully Connected Layer

The outputs of a sequence of convolutional and pooling layers are usually unrolled to form a vector, in a process called *flattening*. The flattened vectors are passed to Fully-Connected layers. *Fully-Connected layers* are functions that map vector representations of an input $\mathbf{x}$ from $\mathbb{R}^n$ to $\mathbb{R}^m$. These functions are defined as an affine transformation of the input vector $\mathbf{x}$:

$$f(\mathbf{x}) = \mathcal{W}\mathbf{x} + \mathbf{b} \tag{2.5}$$

where $\mathcal{W}$ is a matrix of size $\mathbb{R}^{m \times n}$, and $\mathbf{b}$ is the bias term. The values for the elements of $\mathcal{W}$ and $\mathbf{b}$ are learned from data. Unlike convolutional layers that share kernel weights across inputs, fully-connected layers form weighted mappings — via the matrix $\mathcal{W}$ — between all inputs and outputs, which drastically increases the number of parameters. The resulting pre-activations after the affine transform are then typically passed through a non-linear "activation" function.

2.1.4 Non-linear Activation Functions

As the name suggests, *Non-linear Activation Functions* are designed to help deep learning models learn complex non-linear relations from datasets. In the absence of these functions, deep learning models can only learn some form of an affine transform. In CNNs, non-linear activation functions generally follow convolutional-layers and fully-connected layers. Each components of the outputs from these layers (Equations 2.3 and 2.5) are passed to a pre-defined activation function.

Among the several non-linear activation functions, the *Rectified linear unit (ReLU)* is one of the most commonly used in CNN models. It is defined as $ReLU(x) = \max(0, x)$. Figure 2.4 (left) shows the visualisation of the ReLU activation function. The ReLU function and its variants have practical benefits over

other non-linear activation functions. The most important advantages are their ability to accelerate model convergence and solve the vanishing/exploding gradient problems [107, 43].

The *Sigmoid function* is another important non-linear activation function, which is defined as $\sigma(x) = (1 + e^{-x})^{-1}$. The sigmoid function is a squashing function that takes inputs and maps them to real numbers between 0 and 1, as shown in Figure 2.4 (right). This activation function is commonly used on a CNN's output layers to predict probability in binary classification tasks.

2.1.5 Transposed Convolutional Layer

The transposed convolutional layer is a special kind of layer designed to upsample and increase the spatial dimensions of a given input. This layer is typically present in CNNs designed to perform tasks that require per-pixel predictions such as image segmentation and density estimation. It employs trainable upsampling operations parametrized by weights of kernels. Moreover, by using an activation layer, the transposed convolution layer learns a nonlinear upsampling function.

2.2 Fully Convolutional Network

Fully convolutional networks (FCN) are CNN architectures whose feature extracting layers are exclusively built with convolutional layers. This design permits a network to take an arbitrary sized input, and it drastically reduces the number of trainable parameters without losing the representation learning capacity of a CNN. Fully connected layers are commonly replaced by 1×1 convolutions. Because of the absence of fully-connected layers, inputs are not required to be flattened, which gives the advantage of preserving the input image's spatial information. In general, preserving spatial information is important for tasks that require per-pixel prediction such as semantic segmentation and density estimation.

Current state-of-the art techniques for density estimation regression [116, 105, 39, 2] employ FCNs . Long et al. [56] presented the first end-to-end trainable FCN for pixel-wise prediction. Following this work, Ronneberger et al. [77] proposed the U-Net architecture. The U-Net architecture is designed as a symmetric down-sampling network followed by an up-sampling network. This network was shown to have state-of-the art performance for biomedical image segmentation applications. Different extensions of the U-Net architecture have been successfully applied to counting tasks [95, 82].

2.3 Object Counting

Object counting is a fundamental task in computer vision that tries to estimate the number of instances of an object in a given image or video [49]. Formally, the learning problem of object counting can be stated as given an image or video-frame, I_j, finding parameters θ for a model $\mathcal{G}$ that could map from the image to a real number — $\mathcal{G}(I_j; \theta) \mapsto \mathbb{R}$ — where the real number represents the number of instances of the object of

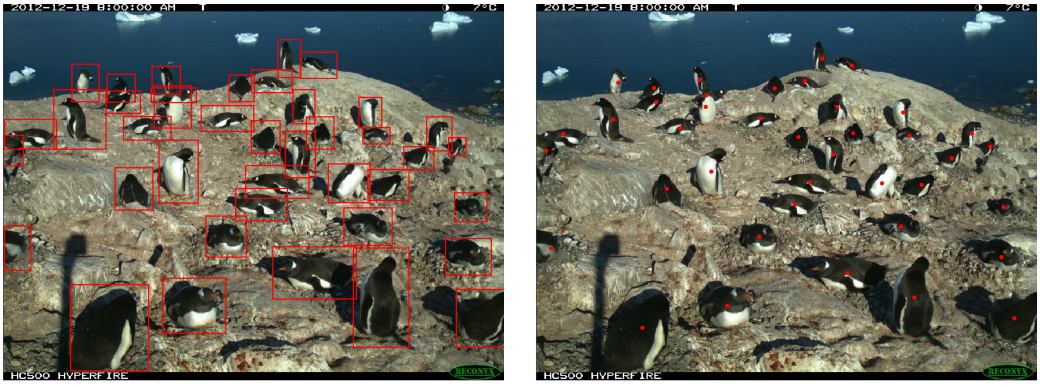

Figure 2.5: Example ground truth annotations overlaid on top of a sample image taken from the Penguin dataset [5]: (left) counting by detection method and (right) counting by density estimation

interest present in I_j. A number of approaches have been proposed to train CNNs for the object counting task. Among these, the most prominent methods are counting by object detection [61, 28, 27], counting by regression [31, 18, 91, 1], and counting by density estimation [49, 26, 84, 74].

Counting by detection methods are two step processes that first detect objects in an image followed by counting the number of detected objects. These methods use object detection models that could localize and classify instances of an object in an image. Commonly, object detection models localize objects with axis-aligned bounding boxes. Figure 2.5 (left) shows sample ground truth that can be used to train an object detection models where each Penguin in enclosed by a rectangular bounding box. Counting by detection method uses these bounding boxes to estimate the number of object instances. The accuracy of counting by detection methods is highly dependent on the performance of the detection model used. Additionally, these methods' performance is hindered by occlusion and high density of objects.

Counting by regression methods estimate count by directly predicting an integer representing the number of objects in an image. Models designed to estimate count by regression extract features from images and output a number. Counting by regression methods are less prone to inaccuracies caused by occlusion and high density. However, these methods disregard all location information.

Alternatively, density estimation methods estimate a count by generating continuous density maps and then integrating over the values of the density maps. As opposed to counting by regression methods, density estimation methods employ localized regression to estimate density maps, which preserves the location information. Density estimation based counting was first proposed by Lempitsky and Zisserman [49], who demonstrated higher counting accuracy with smaller amounts of data. Density estimation based methods use weak labels (dot annotations on object centers) which are less tedious to annotate than bounding boxes or instance segmentation masks. Dot annotations are a set of 2D points, P_j, representing the pixel positions. Figure 2.5 (right) shows example ground truth with dot annotation where each Penguin's position is represented using a dot. Given dot annotated ground-truth, density maps are generated by filtering the

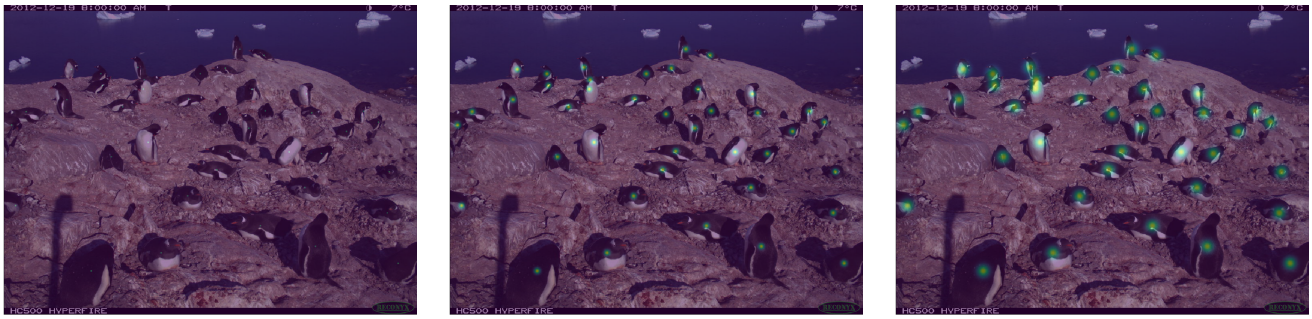

Figure 2.6: Effects of increasing the standard deviation (σ) for the Gaussian kernel on a sample dot annotation overlaid on the corresponding image [5]. (Left) $\sigma = 1$, (Middle) $\sigma = 5$ and (Right) $\sigma = 10$

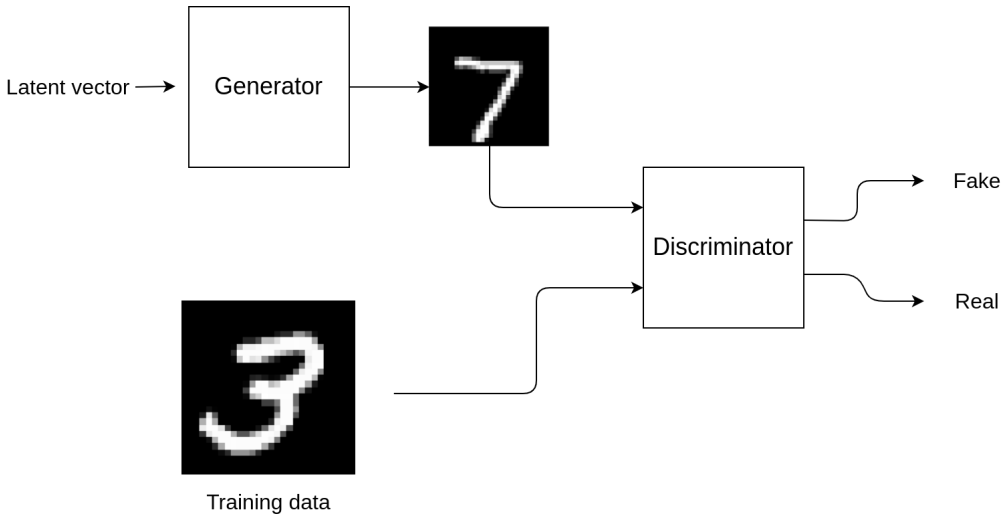

Figure 2.7: Simplified model for Generative Adversarial Networks [32]. The generator and discriminator are deep learning models where the generator is tasked to generate handwritten digits using MNIST [46] dataset as the training set

annotation using a normalized Gaussian kernel with mean $\mu = 0$ and standard deviation σ controlling the area of the kernel. As can be seen from Figure 2.6, increasing the standard deviation increases the area of the kernel. At prediction time, the total number of objects is computed by summing over the pixel values of the density map.

2.4 Adversarial Learning Framework

Adversarial learning is a learning framework used to train Neural Networks that compete to optimize for different but interdependent tasks. In the learning framework, these networks compete to help advance each other's performance. Learning in this manner creates robust representation learning and increases

generalizability [33]. This training scheme has been used in tasks such as image generation [41, 62, 40], image-to-image translation [15, 119], and domain adaptation [37, 14].

Goodfellow et al. [32] presented one of the most influential adversarial learning techniques: the Generative Adversarial Network (GAN). GANs are generative models that try to learn the distribution of data via a minimax game between two networks, namely Generator (G) and Discriminator (D). Figure 2.7 shows a simplified model for GANs. The generator, G, takes a latent vector $z \sim p_z(z)$ as input and tries to map it to an output: data point similar to the training data. The latent vector is a noise sampled from multivariate Gaussian distribution. The intention of using a the noisy latent vector is to help the generator produce diverse output. In the absence of that, the network might learn to produce a single data point from the data distribution. The discriminator is a classifier tasked to decide whether a data point is sampled from the training data or generated by the generator. In the training loop, the discriminator is provided with real data points sampled from the training data and the generator's outputs with labels real and fake, respectively. Therefore, the discriminator learns to distinguish between real and fake data. In parallel, the generator is tasked to trick the discriminator into classifying a generated data point as sampled from the training data.

Goodfellow et al. [32] formulated this minimax game (i.e., the objective function) as

$$\min_G \max_D V(D,G) = \mathop{\mathbb{E}}_{x \sim p_{data}(x)} [\log(D(x)] + \mathop{\mathbb{E}}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \tag{2.6}$$

where V represents the value function for the minimax game and x is a data point sampled from the training data distribution. At the end of a successful training, the generator learns to generate data indistinguishable from the training data. However, training such models often encounters convergence problems [4].

Theoretically, the competition between the generator network and the discriminator network is defined as a zero-sum game. Assuming that the two networks have sufficient capacity, it has been shown that Nash equilibrium is achieved when the generated data distribution matches the data distribution, and the discriminator always predicts 0.5 for all inputs. A prediction of 0.5 from the discriminator for inputs sampled from the training data as well as for the generated data means the discriminator cannot distinguish between real and fake data points. This is the theoretical convergence point when training GANs.

2.5 Domain-Shift

Training deep learning models from scratch requires large annotated datasets. Training deep learning models in the absence of such data would create generalization issues. Given sizeable deep learning models with the number of parameters exceeding the number of training data points, Zhang et al. [113] empirically showed the models could easily memorize training datasets under different scenarios. Recent deep learning models with state-of-the-art performance in different tasks have millions of parameters. However, collecting and annotating datasets in the order of magnitude of these models' parameters is expensive. This problem, in turn, hurts the generalizability of deep learning models.

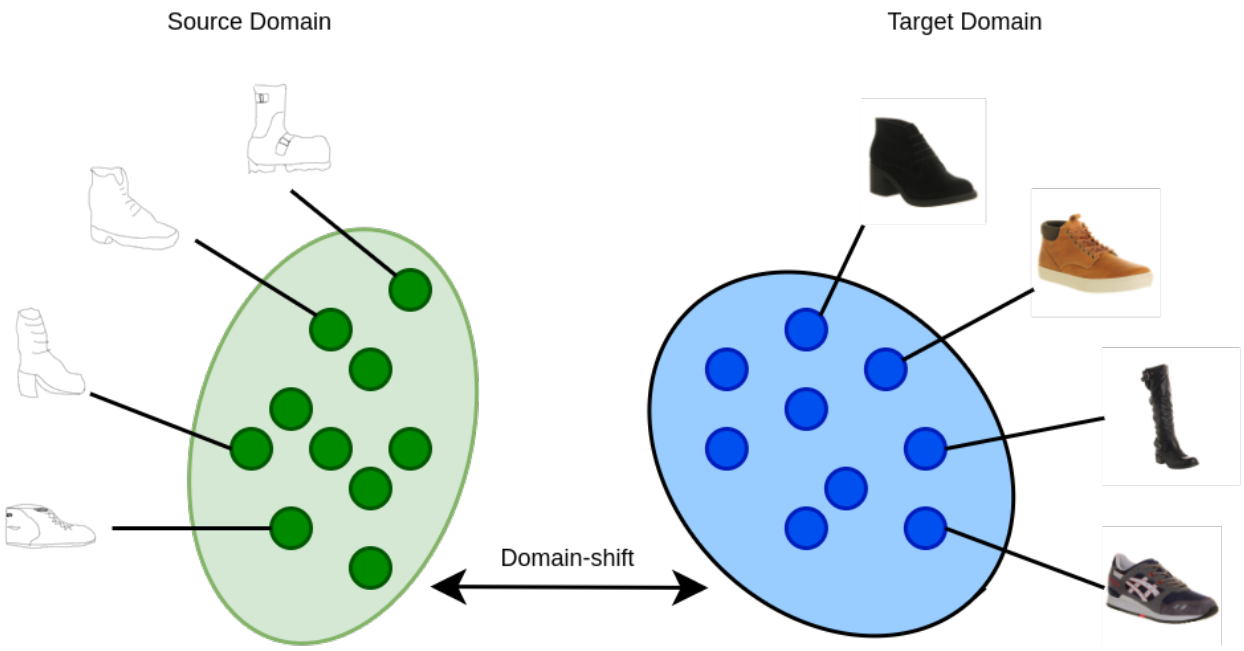

Figure 2.8: Domain-shift between the representation of a source domain (free-hand sketches [111]) and a target domain (RGB images [111])

Machine learning applications generally assume the training and testing dataset are sampled from the same underlying distribution. However, in real-world deep learning applications, it is impractical to rely on such assumptions given the challenges in collecting sufficient data that could align with the various test data distributions when the models are deployed in the real world. Such shifts in distributions across datasets have been shown to hinder the performance of deep learning models [20, 60, 19]. This shift in the joint distributions of input and output between training and testing data is called *Domain Shift*.

In image-based applications, domain-shift can be a consequence of change in background, lighting conditions, imaging sensor, color, and many other factors. In practice, these changes affect the performance of a trained deep learning model. Commonly, in domain adaptation setting, the source task $\mathcal{T}^s$ is similar to the target task $\mathcal{T}^t$. Here, we focus on one of the most practical kinds of domain-shift: covariate shift. The covariate shift is a domain-shift setting in which the distribution of the inputs changes across datasets. Figure 2.8 shows a graphical example showing domain-shift between the representations for the free-hand sketches and RGB images with the same class label (i.e., shoe).

A number of strategies have been proposed to reuse learned weights to alleviate these generalization problems caused by domain-shift. The two most prominent schemes are transfer learning and domain adaptation.

2.5.1 Transfer learning

Transfer learning (TL) is one of the most prominent techniques developed to overcome the problem of insufficient data. It is a technique that tries to transfer the knowledge a model learns from a sufficiently annotated dataset $\mathcal{D}^s$ (source- domain or dataset) for a certain task $\mathcal{T}^s$ (source task) to improve performance on a different task $\mathcal{T}^t$ (target task) or on a similar task with changes in the distribution of the input data $\mathcal{D}^t$ (target- domain or datataset).

Fine-tuning is a TL technique in which a deep learning model is pre-trained on an existing dataset (i.e., source data $\mathcal{D}^s$) with sufficient annotations for a particular task (source task $\mathcal{T}^s$) followed by training on a target task $\mathcal{T}^t$. Even though the source tasks and the target tasks are different, Yosinski et al. [110] showed that transferring learned weights from any number of layers could improve generalization. Figure 2.9 shows a typical fine-tuning process. The process starts with pre-training all the layers of a deep-learning model. The weights optimized for each layer (with the exception of the output layer if $\mathcal{T}^s \neq \mathcal{T}^t$) in the pre-training stage are copied to a similar model designed for the target task. These weights are used as initialization to train for the target task. In the fine-tuning process, the model gets trained on the target dataset, typically with a smaller learning rate than used for the pre-training. A practical application of fine-tuning uses object classification networks — for instance, Inception[85] or EfficientNet[86] — pretrained on the ImageNet[79] dataset to a different image classification task such as plant disease identification [7, 66].

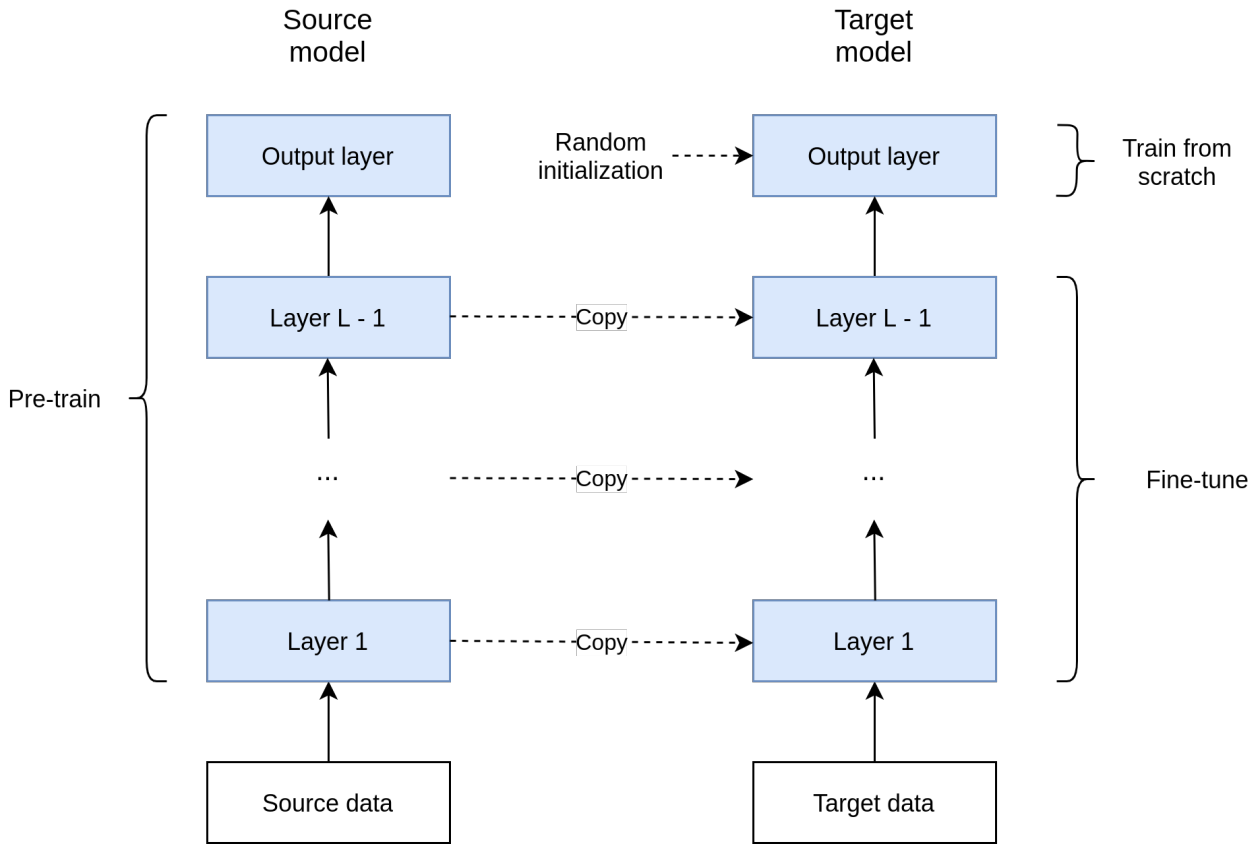

Figure 2.9: Fine-tuning (reproduced from Zhang et al. [112])

2.5.2 Domain Adaptation

Domain Adaptation (DA) approaches the generalizability issues by trying to learn labels from the source domain and adapting the learned knowledge to the target domain via reducing the gap between the source and the target domain distributions. Following the taxonomy from Wilson and Cook [103], domain adaptation can be categorized as Supervised DA, Semi-supervised DA, and Unsupervised DA. Supervised DA tasks assume the presence of annotated data in the source dataset ($\mathcal{D}^s$) as well as in the target dataset ($\mathcal{D}^t$) at training time. Semi-supervised DA makes the same assumption about the source dataset but makes use of a mix of labeled and unlabeled target datasets. On the other hand, Unsupervised DA represents the cases where labels are only available for the source domain data.

Deep domain adaptation techniques are methods that leverage deep neural networks to address the domain-shift problem. Wang and Deng [99] classified these approaches into three groups: Discrepancy-based, Reconstruction based, and Adversarial-based. Discrepancy-based techniques try to minimize the divergence between the source and target domain features measured by certain criteria. Commonly, these techniques use statistic-based discrepancy measures such as Maximum Mean Discrepancy [57]. On the other hand, Reconstruction-based methods try to achieve data invariance via data reconstruction. These methods try to reconstruct the target data from the source data and vice versa. The reconstruction aims to get indistinguishable intra-domain representation. Finally, Adversarial-based techniques address the domain-shift problem by leveraging the adversarial learning framework. Generally, these methods try to align the source and target distributions via an adversarial objective that tries to confuse a discriminator, which attempts to distinguish between source and target domain data.

Several works have been done towards understanding the learning bound for unsupervised domain adaptation techniques [114, 9, 10]. Ben-David et al. [9] gave an approximation for learning bounds in the target domain showing the trade-off between the source data size and the target data accuracy. Their work suggests different approaches to learn domain-invariant features via reducing the domain gap. Zhang and Harada [114] proposed a general upper bound by accounting for the joint error created by matching marginal distributions. Quantifying the learning bound for domain adaptation is still an open question in the literature.

3 Literature review

In this chapter, we review research-works that are relevant to our project. We start by discussing works in object counting tasks, particularly in wheat spikelet counting, leaf counting, and crowd counting. Following the discussion, we present various approaches in the domain adaptation literature and their plant phenotyping and crowd counting applications.

3.1 Counting

Object counting is a well-studied computer vision task. It has been examined for tasks such as crowd counting [22, 97, 108, 70], vehicle counting [87, 115], plant-organ counting [18, 1], and cell counting [105, 36]. This thesis focuses on plant-organ and crowd counting because these tasks are prevalent in the object counting literature, and in addition to that, they present general counting challenges, including occlusion, self-similarity, and size variation.

3.1.1 Wheat Spikelet counting

Convolutional Neural Networks have been employed to estimate the number of spikes and spikelets from images taken in the field (Figure 3.1). The in-field spike detection task has been investigated by leveraging segmentation based [80], bounding-box regression-based [34], and density estimation based [106] methods. These approaches have shown to be successful. However, it is more challenging to count individual wheat grains (spikelets) on spikes mainly because of their size and high-density concentration among others.

Pound et al. [74] presented a method for counting and localizing spikes and spikelets using stacked hourglass networks. They trained and evaluated their model on spring wheat plant images taken in a controlled environment. Their approach achieved a 95.91% accuracy for spike counting and a 99.66% accuracy for spikelet counting. Even though their results show high accuracy, counting spikelets from field images requires further fine-tuning of their model.

Alkhudaydi et al. [2] proposed a method to count spikelets from infield images using a fully convolutional network named SpikeletFCN. They compared the performance of training SpikeletFCN from scratch on infield images and fine-tuning the model, which is initially trained using images taken in a controlled setting. Additionally, they showed manually segmenting the background increases the performance of their proposed model. However, manually annotating spikelets for infield datasets and manually removing background is a time-consuming process.

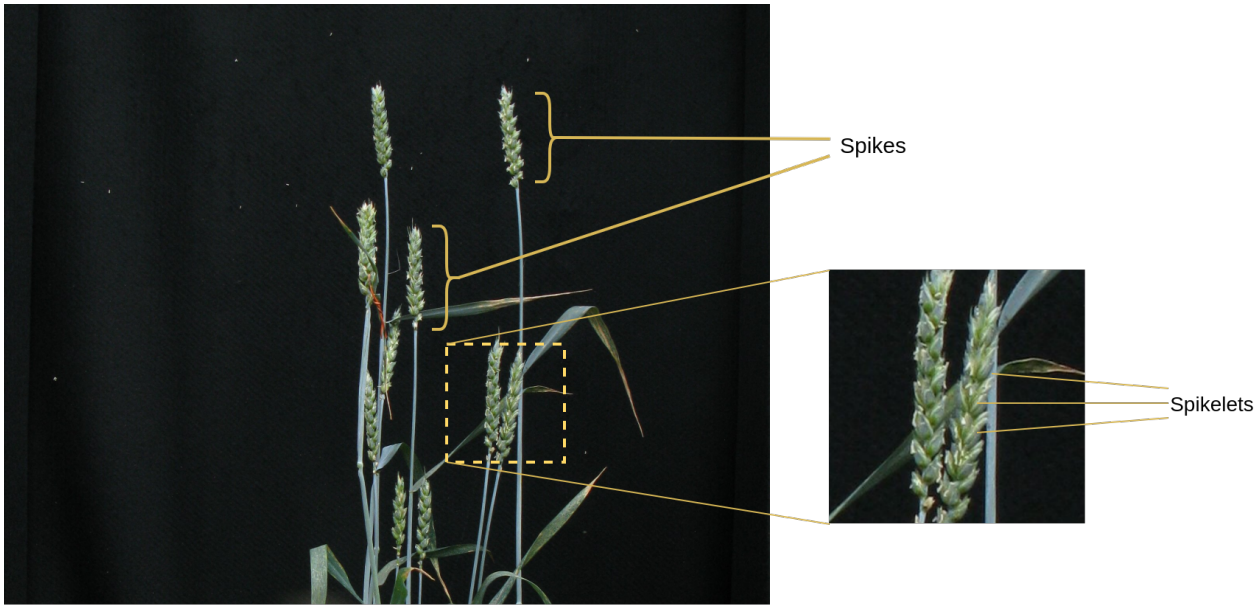

Figure 3.1: Wheat Spike and Spikelet. Image taken from the ACID dataset [74]

3.1.2 Leaf counting

Estimating the number of leaves in a plant is an important phenotypical trait related to its growth stages and yield potential [30]. However, manually counting leaves is a time-consuming and error-prone process. Towards automating this task, the CVPPP leaf segmentation and counting challenge was organized. The challenge featured a dataset composed of top view images of Arabidopsis and tobacco plants with their corresponding annotations, including segmentation masks and dot annotations. Following the challenge, the leaf counting task has become a popular counting task. Leaf counting in rosette plants has been widely studied, and CNN-based regression approaches have shown promising results.

Dobrescu et al. [18] proposed a direct regression based method to estimate the number of leaves from an image. They used a pre-trained ResNet architecture fine-tuned on rosette images to demonstrate state-of-the-art leaf counting performance in the CVPPP 2017 leaf counting competition. They showed combining datasets from sampled from different datasets and across different species improves the performance of their model. Their method achieved a 20% decrease in mean absolute difference in count compared to previous segmentation based algorithms.

Aich and Stavness [1] presented a two step leaf counting method, first using a encoder-decoder network to create a semantic segmentation separating the plant region from background, and then using a VGG-like CNN to estimate the count. To get a better performance of their method they used data augmentation techniques. Their method achieved an mean absolute count difference of 1.62 tested on the five datasets from the CVPPP 2017 leaf counting competition.

Giuffrida et al. [31] extended the CNN regression method by creating a multi-input network that leverages features from multiple modalities. They demonstrated that using multiple image modalities can improve counting accuracy. They also showed their architecture could easily be extended to accommodate different modalities. They demonstrated the generality of the approach by evaluating it with different plant species and image modalities.

Itzhaky et al. [39] proposed two deep learning based approaches that utilize a Feature Pyramid Network (FPN) for counting leaves, namely using direct regression and density map estimation. For their direct regression based model, they proposed a novel fusion technique to aggregate multi-scale predictions. This method achieved an MSE 1.49 tested across the five datasets from the CVPPP 2017 leaf counting competition. On the other hand, their density map based approach achieved an MSE of 1.17.

Ubbens et al. [90] presented a method that utilizes synthetic plants to improve the performance of a Convolutional Neural Network in the leaf counting task. The authors showed improved performance by training models with real plant images augmented with synthetic ones. Additionally, they showed that training CNN models solely on the computer-generated plant accounts for dataset shift better than training on a dataset with real plant images sampled from one distribution when tested on real plant images from another distribution.

3.1.3 Crowd counting

Crowd counting is one of the most popular counting tasks studied in the object counting literature. It is applied in real-world tasks such as public safety and surveillance. Due to the nature of the task, crowd counting commonly presents various challenges: occlusion, size and scale variability, and variation in background. Several CNN-based techniques have been developed for the crowd counting task [84, 26]. The techniques are mainly categorized as detection-based [104, 3], regression-based [12, 38], and density estimation [22, 116, 6, 97, 13]. Gao et al. [26] further classified the crowd estimation based on the availability of data at training time (form of supervision) as (1) Fully-supervised methods and (2) Un/semi/weakly/self-supervised methods. In this sections, we briefly review some of these methods.

Fu et al. [22] presented the first CNN based fully supervised crowd density estimation method. The authors proposed an architecture with multi-stage cascading CNNs that first starts by classifying if a sample has a complex background, and then performs a crowd density estimation. Zhang et al. [116] presented a Multi-Column Convolutional Neural Network architecture to estimate crowd density from an input image. Their proposed architecture is built from three columns, each defined by their filter size. This arrangement is targeted to account for differences in people's head size in images caused by a perspective effect and image resolution. Their results showed that they outperformed all state-of-the-art methods at the time. Babu Sam et al. [6] proposed a switching convolutional neural network that accounts for within image variations of crowd density. Their method switches between columns with different receptive fields instead of fusing them together. Their architecture passes patches extracted from images to a network designed to select a crowd

density estimator. This network, in turn, selects the estimator with appropriate receptive field size. Fang et al. [21] presented Locality-constrained Spatial Transformer Network (LSTN), which uses CNNs for estimating a density map from video frames. The method leveraged Locality-constrained Spatial Transformer (LST) to relate the density estimate from the current frame to the following. They also presented a dataset with 15,000 video frames called the Fudan-ShanghaiTech (FDST) dataset. Evaluating their proposed method on the FDST dataset achieved an MSE of 4.45.

Olmschenk et al. [68] proposed a semi-supervised crowd density estimation technique by leveraging the GAN objective. They showed their method outperforms a similar CNN trained with significantly more labeled data. In their following work, Olmschenk et al. [69] presented a semi-supervised method with a dual-goal GAN (DG-GAN) to learn from partially labeled data. They showed their method improves the performance of the discriminator network (i.e., the density estimation network) compared to a similar CNN trained with the same data but without the adversarial loss.

3.2 Domain Adaptation

In recent years, a number of deep domain adaptation approaches have been proposed to solve the domain-shift problem [99]. The general approaches are classified as supervised [67, 42], semi-supervised [88, 109], and unsupervised [58, 11, 89, 55]. Here, we mainly focus on adversarial-based unsupervised domain adaptation techniques.

Liu and Tuzel [55] presented a framework called coupled generative adversarial networks (CoGAN) that could learn the joint distribution of images sampled from multiple domains. CoGAN is designed with a pair of GANs with tied weights in the first and last few layers of the network. This design is targeted to force the network to learn joint distributions from samples drawn independently from each domain's marginals. Their experiment showed CoGAN learned correspondence of images in the two domains without any supervision.

Li et al. [51] presented a novel unsupervised density-based domain adaptation technique trained via an adversarial learning approach. Their method leverages the local structure similarity in density patterns. Their method, additionally, included a scale-aware adaptation by providing their network a multi-scale pyramid of patches. Their experiments showed that their model resulted in performance close to state-of-the-art object counting techniques. However, their proposed method requires pre-training a density estimation network before performing the domain adaptation task.

Tzeng et al. [89] proposed a generalized framework to an adversarial approach for an unsupervised domain adaptation named Adversarial Discriminative Domain Adaptation (ADDA). Their proposed method takes the advantages of discriminative modeling to learn representations for input images. This results in asymmetric mappings assigning different feature space for the source and target domain data. ADDA, then, minimizes the representation shift between the two domains using a GAN loss function.

As an alternative to a GAN loss, Ganin and Lempitsky [24] presented an unsupervised domain adaptation

technique by creating a network with shared feature extraction layers and two classifiers. In their proposed method, the first classifier is a class label predictor, whereas the second one is a domain classifier. The adaptation process works by minimizing the class label prediction loss and maximizing the domain confusion. The authors showed that their proposed method learns to align the source and target domain representation for different image classification tasks.

Most of the previous domain adaptation works have only been evaluated on image classification problems. Domain adaptation has garnered less attention for other computer vision tasks, such as density estimation or regression problems.

3.2.1 Domain Adaptation in Plant Phenotyping

Valerio Giuffrida et al. [94] proposed a method to adapt a model trained to count leaves in a particular dataset to an unseen dataset in an unsupervised manner. Under the assumption that the images in the source domain are private, their method uses the representations of images from the source domain (from a pretrained network) to adapt to the target domain. They employed the Adversarial Discriminative Domain Adaptation approach to solve the domain shift problem in the leaf counting task across different datasets. Their method aligns the predicted leaf count distribution with the source domain's prior distribution, which limits the adapted model into learning a leaf count distribution similar to the source domain. They have shown their method achieved a best-case mean square error of 2.36 and 1.84 for intra-species and inter-species domain adaptation experiments, respectively.

For fruit counting in orchard images, Bellocchio et al. [8] proposed a Cycle-GAN based domain adaptation method combined with weak presence/absence labels. They demonstrated state-of-the-art counting results for many combinations of adapting between plant species (almond, apple, olive). These promising results on leaf and fruit counting motivate a broader investigation of different domain adaptation approaches, which could be used for counting different plant organs and different categories of domain shift, e.g. from indoor image to field images. To the best of our knowledge, this is the first work that has applied domain adaptation to wheat spikelet counting and to a counting task with an indoor-to-outdoor domain shift.

3.2.2 Domain Adaptation in Crowd counting

Wang et al. [101] presented a method, called Neuron Linear Transformation (NLT), to represent the domain-shift between the source and the target domain as the difference in model weights. NLT updates weights in the adapted model by leveraging a few annotated target data via linear transformation. Their synthetic-to-real experiments have shown their method outperformed other domain-adaptation techniques. However, their method requires the presence of a few annotated images.

Wang et al. [100] proposed to train and adapt crowd counting from synthetic data to real world data. They presented two approaches that could increase the performance of a crowd counting model. On the first approach, they designed a Spatial Fully Convolutional Network (SFCN) which was first pretrained with the

synthetic data and fine-tuned on real world datasets. The second approach utilizes a domain adaptation technique where the authors employed an SSIM Embedding (SE) Cycle GAN to change the synthetic images to photo-realistic images. Then these images are used to train the SFCN model. Their domain adaptation approach showed a reasonable increase in performance from their baselines.

3.3 Summary

Object counting is an important and well-studied computer vision task. Various deep-learning-based methods have been proposed to approach the object counting task. Recently, density-based count estimation methods have become the most prominent techniques to approach the problem. The evaluation of these methods resulted in high accuracy for different counting tasks.

However, changes in the distribution of test images (e.g., from indoor to outdoor) hiders the performance of object counting methods. For this reason, domain adaptation is highly relevant for counting. But previous domain adaptation methods in the literature have focused on classification or have strong assumptions about aligning source/target distributions. Therefore, a domain adaptation approach tailored to object counting with density estimation and greater flexibility than the previous leaf counting adaptation work would improve organ counting in plant phenotyping.

4 Method

In this chapter, we discuss the problem setting for our proposed method, followed by a description of the proposed architecture and cost function. Finally, we describe the training procedures used for all experiments.

4.1 Problem setting

We model the task of domain adaptation in object counting as an unsupervised domain adaptation problem. We assume that the data is sampled from two domains: a source domain ($\mathcal{D}^s$) and a target domain ($\mathcal{D}^t$). Furthermore, we assume that labels only exist for the images sampled from the source domain. Therefore, the source domain is composed of a set of images ($\mathcal{X}^s$) and their corresponding labels ($\mathcal{Y}^s$). Where as, the target domain only has images ($\mathcal{X}^t$) without labels.

4.2 Architecture

Our proposed model is a Domain-Adversarial Neural Network (DANN), a class of models designed for unsupervised domain adaptation tasks [25]. These networks are traditionally characterized by two parallel classification networks that share weights in the first n layers and a custom layer called Gradient Reversal Layer (GRL) [99]. One of the parallel networks is designed for the main classification task, while the second network is a classifier designed to discriminate whether an input is sampled from the source domain or from the target domain. We customize this general architecture by replacing the main classification network with a U-Net network [77] that is used for density map estimation.

Our proposed architecture is a fully convolutional network composed of two parallel networks, as can be seen in Figure 4.1. These parallel networks share weights from *Conv1* to *Conv8*. Given these shared weights, the architecture can be seen as one network containing three subnetworks: the downsampling subnetwork (G_d), the upsampling subnetwork (G_u), and the domain classifier subnetwork (G_c). In Figure 4.1 these subnetworks are denoted by different color boxes. The model takes an image x_j and predicts a density map as $\hat{y}_{density} = G_u(G_d(x_j))$ and a class probability prediction representing whether x_j is sampled from the source domain or from the target domain given by $\hat{y}_{domain} = G_c(G_d(x_j))$.

A downsampling subnetwork followed by an upsampling subnetwork (sometimes referred to as an Hourglass network or encoder-decoder network) is a commonly used architecture for density estimation tasks[54, 105]. The general structure for the Hourglass network we used is adapted from U-Net [77] with modifications.

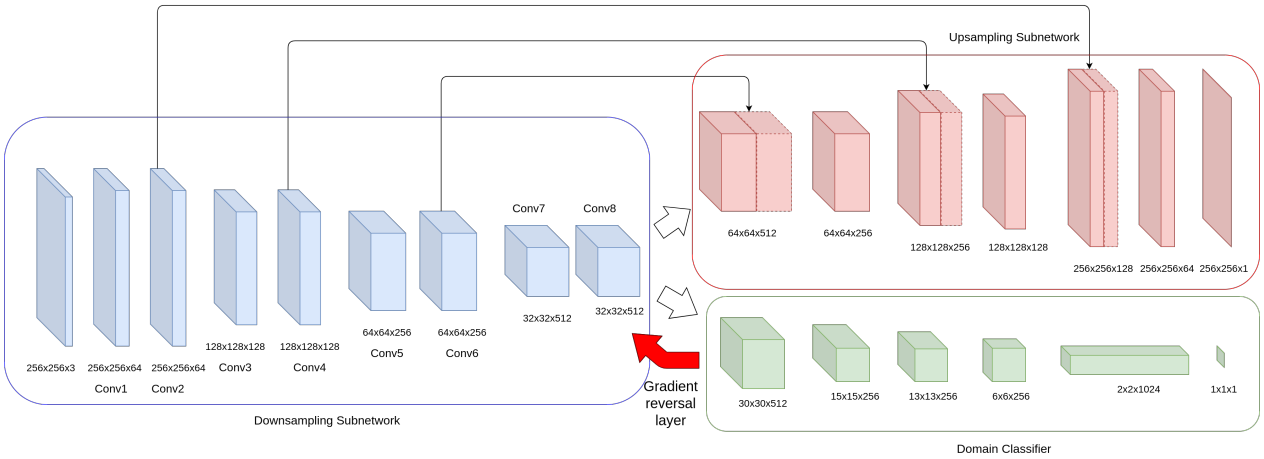

Figure 4.1: The proposed Domain-Adversarial Neural Network composed of two networks that share weights between Conv1 and Conv8. The downsampling subnetwork (G_d), the upsampling subnetwork (G_u), and the domain classifier (G_c) are denoted by the blue, red, and green boxes respectively. The red arrow shows the Gradient Reversal Layer used to reverse the gradient in the backpropagation step as proposed by [24].

The downsampling subnetwork is composed of blocks of two 3×3 padded convolutions followed by a 2×2 max pooling layer with a stride of 2. The upsampling network is composed of blocks of 2×2 transpose convolutions followed by a pair of 3×3 padded convolutions. The feature maps generated from each convolution preceded by a transpose convolution are concatenated with their corresponding feature maps from the upsampling subnetwork (shown in Figure 4.1 as dashed boxes). The domain classifier subnetwork is formed from 3×3 unpadded convolutions, a 2×2 maxpooling layers, and a 1×1 convolution (which is used to reduce the dimension of the output to a single value). All of the convolutions in our network with the exception of the ones in the output layers are followed by batch normalization and a ReLU activation. The output layer is followed by a sigmoid activation function.

The output of the downsampling subnetwork passes through a Gradient Reversal Layer (GRL) before passing through to the domain classifier subnetwork. The GRL is a custom layer proposed by Ganin and Lempitsky [24] which is designed to reverse the gradient in the backpropagation step by multiplying the gradient by a negative constant ($-\lambda$). The GRL acts like an identity function in the forward propagation steps; however, it reverses the gradient in the backward propagation steps. The purpose of reversing the gradient is to ensure the network would not recognize between the source and target domain images via the extracted features, which confuses the domain classifier. Since the density estimation sub-network is getting trained in parallel with the domain classifier, the downsampling sub-network is forced into finding domain-invariant features common to both the source and target domain. This, in turn, aligns the distributions of the representations of images sampled from the two domains.

4.3 Cost functions

The parameters for the proposed model are optimized via minimizing two cost functions. The first cost function accounts for errors in the density map estimation for samples taken from the source domain. This cost is implemented as the logarithm of mean square error between the ground truth density map sampled from the source dataset and the predicted density map from the model which is given by

$$\mathcal{L}_{density} = \log(\frac{1}{N}\sum_{i=0}^{N}(G_u(G_d(x_i^s)) - y_i^s)^2) \tag{4.1}$$

where (x_i^s, y_i^s) represents the i^{th} image and density map label sampled from the source domain ($\mathcal{D}^s$).

The second cost function is a Binary Cross Entropy loss for domain classification:

$$\begin{aligned}\mathcal{L}_{domain} = &-\mathbb{E}_{x^s\sim\mathcal{X}^s}[\log(G_c(G_d(x_i^s))] \\ &-\mathbb{E}_{x^t\sim\mathcal{X}^t}[1 - \log(G_c(G_d(x_i^t))]\end{aligned} \tag{4.2}$$

Finally the total cost is given by

$$\mathcal{L} = \mathcal{L}_{density} + \mathcal{L}_{domain} \tag{4.3}$$

The overall optimization goal in training our proposed model is a competition to ensure that the total loss ($\mathcal{L}$) decreases throughout while increasing the domain classification loss ($\mathcal{L}_{domain}$). Therefore, for successful training, the increase in the domain classification loss should be accounted for by decreasing the density loss ($\mathcal{L}_{density}$) to keep the total loss to decrease throughout the training consistently. The choice of the logarithm of mean squared error instead of mean squared error, regularly used as a density estimation loss, is intended to balance the two costs.

4.4 Training

Conceptually, the general training scheme we use to train our model can be decomposed into two sub-tasks: supervised density estimation on the source dataset, and domain classification on the merged source and target datasets. Algorithm 1 presents a pseudo-code showing the general training scheme used to optimize our proposed model's weights. The goal of this training scheme is to extract relevant features for the density estimation task and, in parallel, condition the downsampling network to extract domain-invariant features. This is achieved by minimizing the density map estimation error ($\mathcal{L}_{density}$) and maximizing the domain classification error ($\mathcal{L}_{domain}$).

In all experiments, we randomly partitioned the source domain dataset into training and validation sets. The images from the training partition of the source domain are mixed with the target domain images. Following that, the mixed training set is shuffled. Shuffling the training set is intended to remove possible bias from the dataset creation. Since we only have ground truth density maps for the source domain,

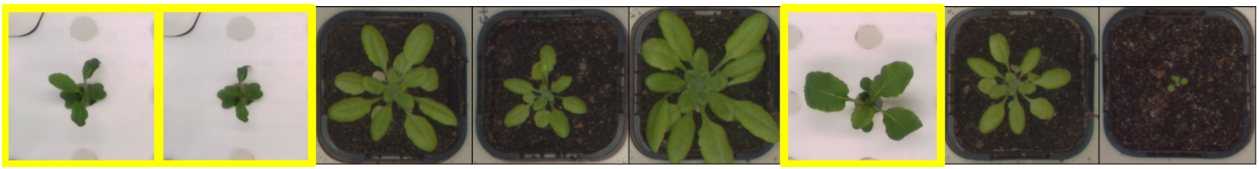

Figure 4.2: Example batch for training our proposed method where the source domain images are taken from the CVPPP dataset [65, 64], and the target domain images are taken from the KOMATSUNA Dataset [92]. The images denoted by yellow outline are taken from the target dataset.

Table 4.1: Range of parameters used in the grid search to tune the learning rate hyperparameter

	Minimum Value	Maximum Value
Downsampling subnetwork	10^{-6}	10^{-2}
Upsampling subnetwork	10^{-6}	10^{-2}
Domain classifier	10^{-6}	10^{-2}

in-training validations are carried out using the validation set taken from the source domain. In-training validation is intended to prevent overfitting by evaluating our model's performance in estimating density in each epoch.

The model was trained with a batch size of 8. The input images in the training set were resized to 256×256 pixels. These images can optionally be small patches taken from high-resolution images. Figure 4.2 shows example training images used to create one batch, where the images from the target domain are denoted by the yellow outline. We used the Adam optimizer with tuned learning rates for each subnetwork. To tune the learning rate hyperparameters, we leveraged a grid search method. Table 4.1 shows the range of values used in the grid search to tune the three subnetworks' learning rate hyperparameters. We used a learning rate of 10^{-3} for the downsampling and upsampling subnetworks, and 10^{-4} for the domain classifier.

The constant parameter λ — which multiplies the gradient in the GRL layer — is updated on each iteration during training. Following Ganin et al. [25], the value is provided to the network as a scheduler in an incremental manner ranging from 0 to 1.

$$\lambda_p = \frac{2}{1 + \exp\left(-10p\right)} - 1 \tag{4.4}$$

As the training progresses, λ increases the effect of the reversed gradient, which is intended to have less effect in the initial stages when the model's parameters are noisy.

Algorithm 1: Training Algorithm

Input: 256×256 RGB image and label pairs from the source domain $\{(x_i^s, y_i^s)\}_{i=1}^n \sim \mathcal{D}^s$,
256×256 RGB images from the target domain $\{x_i^t\}_{i=1}^n \sim \mathcal{D}^t$

Result: learned weights

initialize G_d, G_u and G_c

merge x_i^s and x_i^t and shuffle

for *each training epoch e* **do**

  for *each batch b* **do**

    for *each data point* x_i *in b* **do**

      if $x_i \sim \mathcal{D}^s$ **then**

        $\hat{y}_{density} = G_d(G_u(x_i))$

        $\hat{y}_{domain} = G_d(G_c(x_i))$

        $\mathcal{L} = \mathcal{L}_{density}(\hat{y}_{density}, y_i) + \mathcal{L}_{domain}(\hat{y}_{domain}, 1)$

      else

        $\hat{y}_{domain} = G_d(G_c(x_i))$

        $\mathcal{L} = \mathcal{L}_{domain}(\hat{y}_{domain}, 0)$

      end

    end

    update weights based on the $\nabla\mathcal{L}$

  end

end

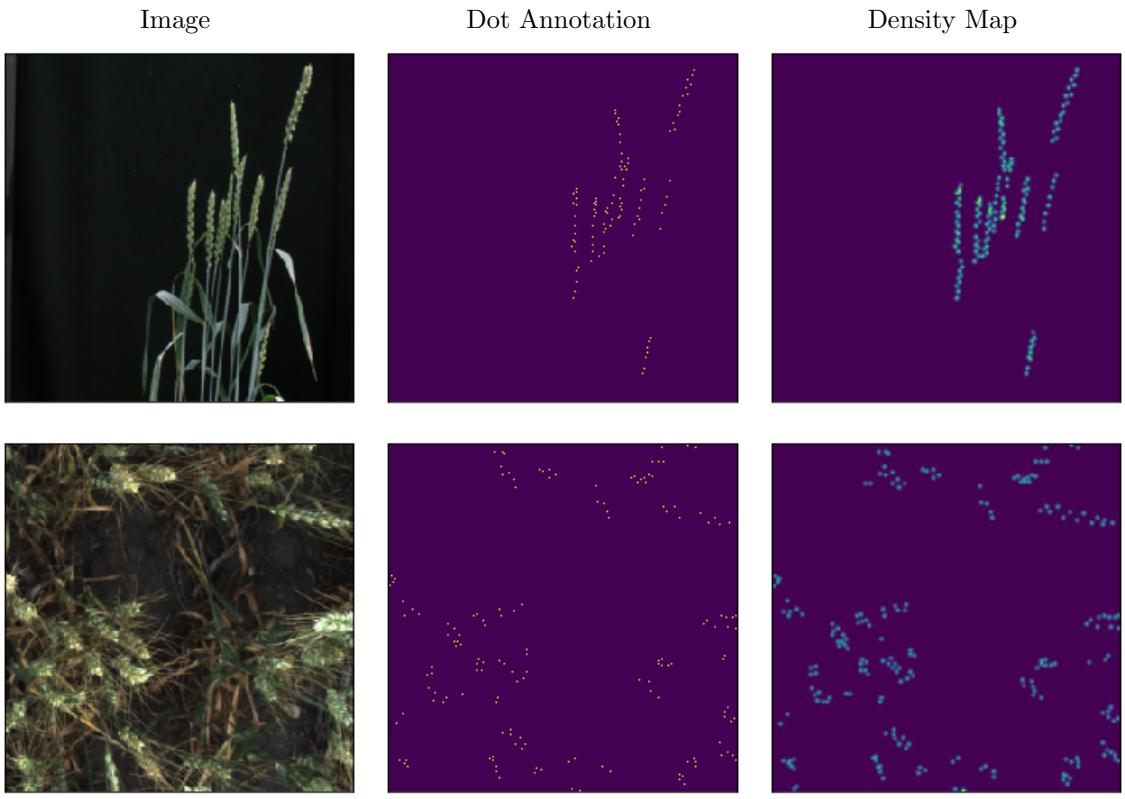

Figure 4.3: Sample ground truth dot annotations and their corresponding density maps

4.5 Ground truth generation

Training our proposed model requires density maps as labels for the source domain images. The density maps are generated from 2D point coordinate, (x, y), ground truth depicting objects' position on images. For all of our experiments, the coordinates of the 2D points are rescaled to the range 0 to 255 in both directions. Afterward, the rescaled coordinates are mapped to a 256×256 image creating dot annotations. Finally, the dot annotations are filtered using a normalized Gaussian kernel with a standard deviation σ provided depending on the size of the object the dots represent. Figure 4.3 shows dot annotations and density maps for sample images. We selected the standard deviation values for each experiment based on the objects' area represented by the dot annotations.

4.6 Baseline models

To demonstrate the counting performance improvement provided by our domain adaptation method, we use a vanilla U-Net as a baseline model by removing the domain classifier sub-network from our proposed architecture. We train this baseline model exclusively on the source domain and evaluate performance with the target domain data. This baseline serves as the expected lower-bound of our experiment. We used a learning rate of 10^{-3} and $\mathcal{L}_{density}$ as the cost function to train the model.

For the target datasets that come with dot annotations, we train a second baseline model — Baseline (Fine-tuned) — in a supervised manner. Two steps are taken to train this model: a pre-training step with the source dataset followed by fine-tuning with the target dataset. On the pre-training step, we used the same setting as the first baseline model. On the other hand, the fine-tuning step starts with a lower learning rate of 10^{-6}, which is typical for fine-tuning. Furthermore, the learning rate decays with a step size of 5 by a multiplicative factor of $\gamma = 0.1$. The results form the fine-tuned models are expected to be better than unsupervised domain adaption because they are fine-tuned with labeled training data from the target domain. Therefore, we consider them as an expected upper-bound to our experiments.

4.7 Experiments

In this section, we present three object counting tasks to evaluate the proposed method: wheat spikelet counting, leaf counting, and crowd counting.

The performance of our proposed method in all of the experiments are evaluated using target domain data as our test set. Our experiments are mainly assessed using Root Mean Square Error(RMSE) and Mean

Absolute Error(MAE):

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=0}^{N}(\hat{y}_i - y_i)^2} \tag{4.5}$$

$$MAE = \frac{1}{N}\sum_{i=0}^{N}|\hat{y}_i - y_i| \tag{4.6}$$

where N represents the number of test images, $\hat{y}_i$ and y_i represent the predicted value and the ground truth for the ith sample, respectively.

In addition to these metrics, we include task specific metrics that are commonly used in the literature for each task. For the wheat spikelet counting task, we provide coefficient of determination (R^2) between the predicted and ground truth counts:

$$R^2 = 1 - \frac{\sum_{i=0}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{N}(y_i - \bar{y})^2} \tag{4.7}$$

where $\bar{y}$ represents the mean of the ground truth values .

For the leaf counting task, we report metrics that are provided for the CVPPP 2017 leaf counting challenge. These metrics are Difference in Count (DiC), absolute Difference in Count ($|DiC|$), mean squared error (MSE) and percentage agreement(%):

$$DiC = \frac{1}{N}\sum_{i=0}^{N}(\hat{y}_i - y_i) \tag{4.8}$$

$$|DiC| = \frac{1}{N}\sum_{i=0}^{N}|\hat{y}_i - y_i| \tag{4.9}$$

$$\% = \frac{1}{N}\sum_{i=0}^{N}\mathbb{1}_{y_i=\hat{y}_i} \tag{4.10}$$

where $\mathbb{1}_{y_i=\hat{y}_i}$ represents an indicator function that takes value 1 when the prediction is equal to the ground truth and 0 otherwise.

All experiments were performed on a GeForce RTX 2070 GPU with 8GB memory using the Pytorch framework. All of our implementations are available at `https://github.com/p2irc/UDA4POC`

4.7.1 Wheat Spikelet Counting

For the wheat spikelet counting task, we used the general problem definition of wheat spikelet counting as defined by Alkhudaydi et al. [2]. Under this setting, the wheat spikelet counting experiment is trying to adapt wheat spikelet density estimation task from images taken in a greenhouse to images capture in an outdoor field. The datasets used for this task are

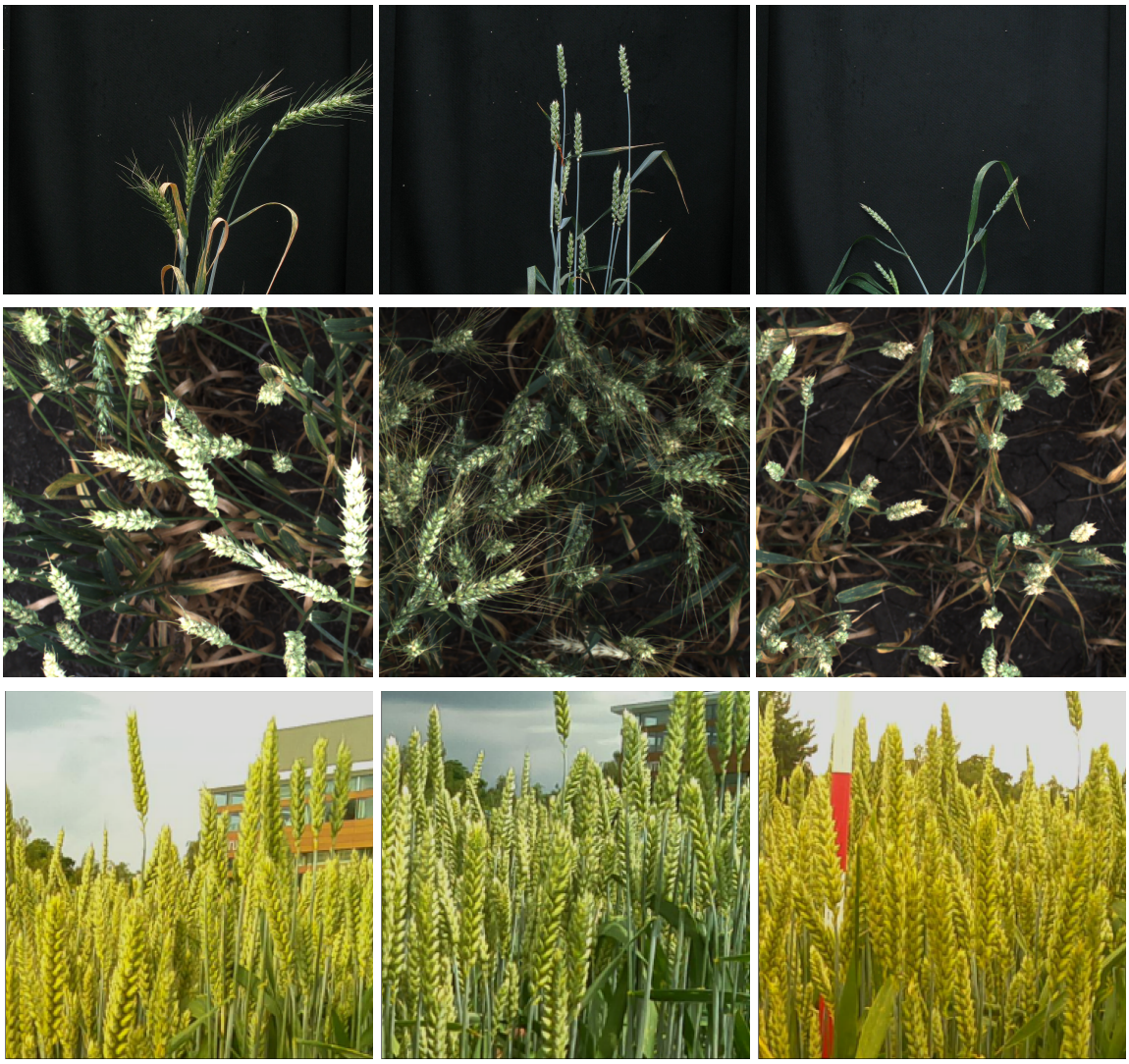

Figure 4.4: Example images for wheat spikelet counting experiments: (Top) Source dataset: ACID [74], (Middle) Target dataset 1: Global Wheat Dataset [17], and (Bottom) Target dataset 2: CropQuant dataset [117].

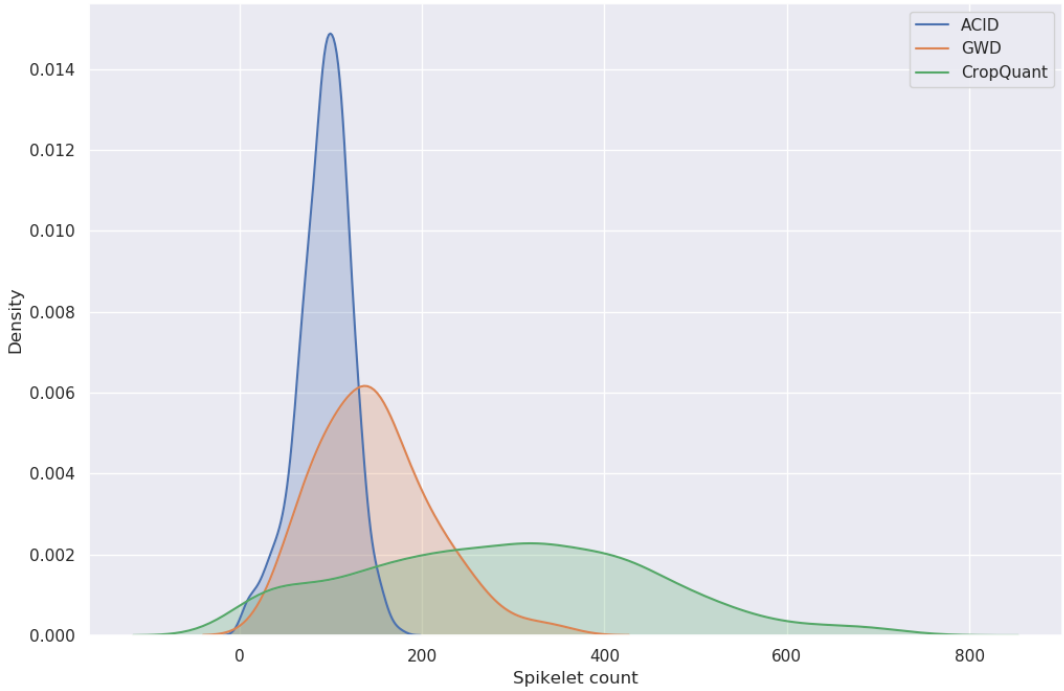

Figure 4.5: spikelet count distribution of the datasets in the wheat spikelet counting experiment

1. *ACID dataset* [74]: This dataset is composed of wheat plant images taken in a greenhouse setting. The dataset is composed of 520 images with a total of 48,000 annotated spikelets. Each image has dot annotations representing the position of each spikelet.

2. *Global Wheat Dataset* (GWD) [17]: A dataset presented in the Kaggle wheat head detection competition. The dataset comes with bounding box annotations for each wheat head, but these are not used in the present study.

3. *CropQuant dataset* [117]: This dataset is composed of images collected by the Norwich Research Park (NRP). Each of the images have corresponding segmentation masks for wheat heads, but these are not used in this study. As a ground truth, we used the dot annotations created by Alkhudaydi et al. [2], where they annotated 15 of the images from the CropQuant dataset. The ground truth has a total of 63,006 spikelets.

Sample images from these datasets are displayed in Figure 4.4. Figure 4.5 presents the distribution of wheat spikelet counts from the three datasets used in the experiment. The plot demonstrates that the wheat spikelet count distributions in our experiments are sampled from three different Gaussian distributions with $\mu = 93.68$ and $\sigma = 28.82$ for ACID dataset, $\mu = 147.46$ and $\sigma = 61.55$ for GWD, and $\mu = 290.43$ and $\sigma = 157.85$ for the patches taken from the CropQuant dataset.

Using these datasets, we designed two indoor-to-outdoor domain adaptation experiments. We chose the indoor ACID dataset as the source domain in these experiments. The groudtruth density maps for the source dataset were generated by filtering the dot annotations with a 2D gaussian filter with $\sigma = 1$.

In the first experiment, we used the GWD dataset as the target domain. We randomly sampled 80% of the images in the source dataset for training. These images were mixed with 200 images taken from the target dataset, making up the training set. Using this as the training set, the network is trained on the ACID dataset for spikelet counting and in parallel adapted to the GWD. To evaluate our method, we created dot annotations for 67 images from the GWD which are used as ground truth. All the annotations are made with a single annotator. These annotations are made publicly available at `https://doi.org/10.6084/m9.figshare.12652973.v3`.

In the second experiment, we used the CropQuant dataset as the target domain. To be consistent with the experiments presented in Alkhudaydi et al. [2], we randomly extracted 512×512 sized patches from each image in the CropQuant dataset. We generated 300 patches as the target domain. We partitioned the source domain with a 80:20 training-validation split. We merged the training split with the generated patches and trained the model. For testing, we randomly extracted 300 patches where the annotations are provided by Alkhudaydi et al. [2].

Figure 4.6: Example images for leaf counting experiments: (Top) Source dataset: Arabidopsis, CVPPP Dataset[65, 64], (Middle) Target dataset 1: KOMATSUNA Dataset [92], and (Bottom) Target dataset 2: Arabidopsis, MSU-PID Dataset [16].

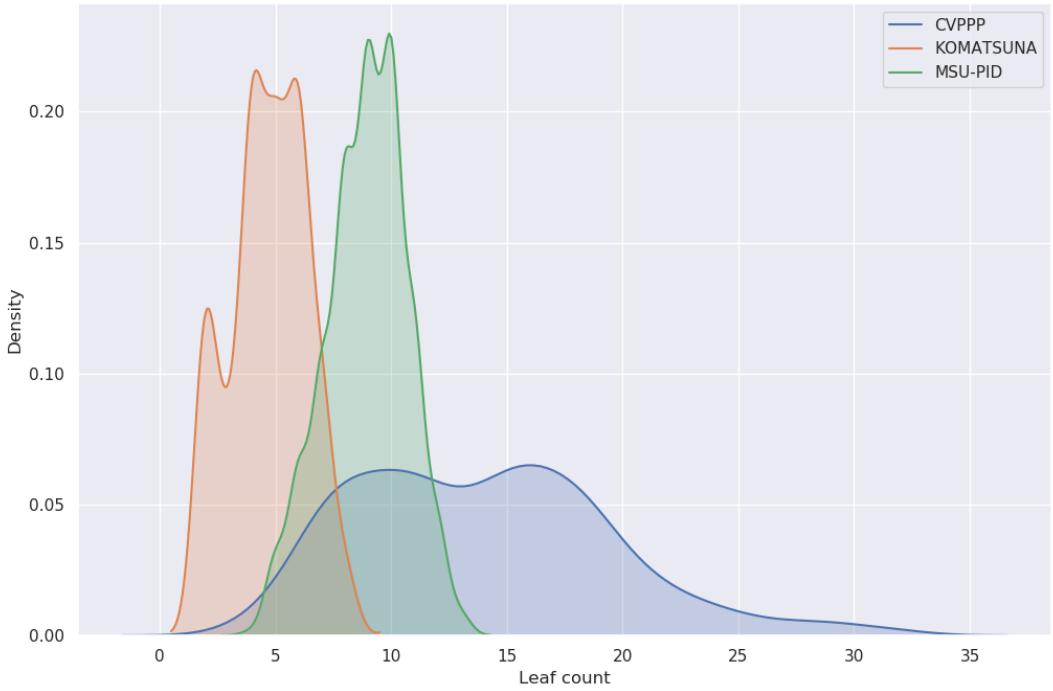

Figure 4.7: Leaf count distribution of the datasets in the leaf counting experiment

4.7.2 Leaf counting

For the leaf counting task, we follow [94] in order to compare to their baseline results. Therefore, we used the following datasets:

1. *CVPPP 2017 LCC Dataset* [65, 64]: A dataset compiled for the CVPPP leaf counting competition, containing top-view images of *Arabidopsis thaliana* and tobacco plants. To be consistent with Valerio Giuffrida et al. [94], we use the A1, A2 and A4 *Arabidopsis* image subsets with a total of 783 images. The dataset includes annotations for leaf segmentation masks, leaf bounding boxes, and leaf center dot annotations.

2. *KOMATSUNA Dataset* [92]: Top-view images of Komatsuna plants developed for 3D plant phenotyping tasks. We use the RGB images for the experiments presented here. The dataset contains 300 images.

3. *MSU-PID*[16]: A multi-modal imagery dataset which contains images of *Arabidopsis* and bean plants. Among these, we use the RGB images of the *Arabidopsis* subset which has 576 images.

Figure 4.6 shows sample images from these datasets where the top and bottom rows are Arabidopsis

Figure 4.8: Example composite images generated by randomly sampling four images from KOMATSUNA dataset and stitching them together.

plants and the the middle row is Komatsuna plants. The leaf count distribution for each of these datasets is shown in Figure 4.7. The distributions show the leaf counts from the three datasets are sampled from different Gaussian distributions with $\mu = 13.98$ and $\sigma = 5.50$ for the CVPPP 2017 LCC Dataset, $\mu = 4.71$ and $\sigma = 1.63$ for the KOMATSUNA Dataset, and $\mu = 8.93$ and $\sigma = 1.72$ for the MSU-PID. Using these datasets, we performed three leaf-counting experiments in total. In all experiments, the CVPPP dataset was used as the source domain. Taking the leaf center annotations, we generated the groudtruth density map by filtering the center points with a 2D gaussian filter with $\sigma = 3$.

The first experiment involved adapting leaf counting from the CVPPP dataset to the KOMATSUNA dataset. For this experiment, we randomly selected 80% of the images from the CVPPP dataset and merged them with the images from the KOMATSUNA dataset. The remaining 20% of the source domain is used for validation. With this setting, we trained our model for 150 epochs, which took around 2hrs on average. Finally, we evaluated performance on the 300 images taken from the KOMATSUNA dataset.

In the second experiment, we used the MSU-PID dataset as the target domain. The data from the source domain was partitioned into training and validation sets in a similar procedure as experiment 1. After partitioning the data, we trained the model for 150 epochs. The model was then evaluated using the 576 images taken from MSU-PID dataset.

In the final experiment, we aimed to verify that our domain adaptation scheme is independent of the leaf count distributions in the source domain. To test this, we took the model trained in the first experiment and evaluated it on composite images created from the KOMATSUNA dataset containing multiple plants. The composite images are generated by randomly selecting four images from the KOMATSUNA dataset and stitching them together to form a 2×2 grid (Figure 4.8).

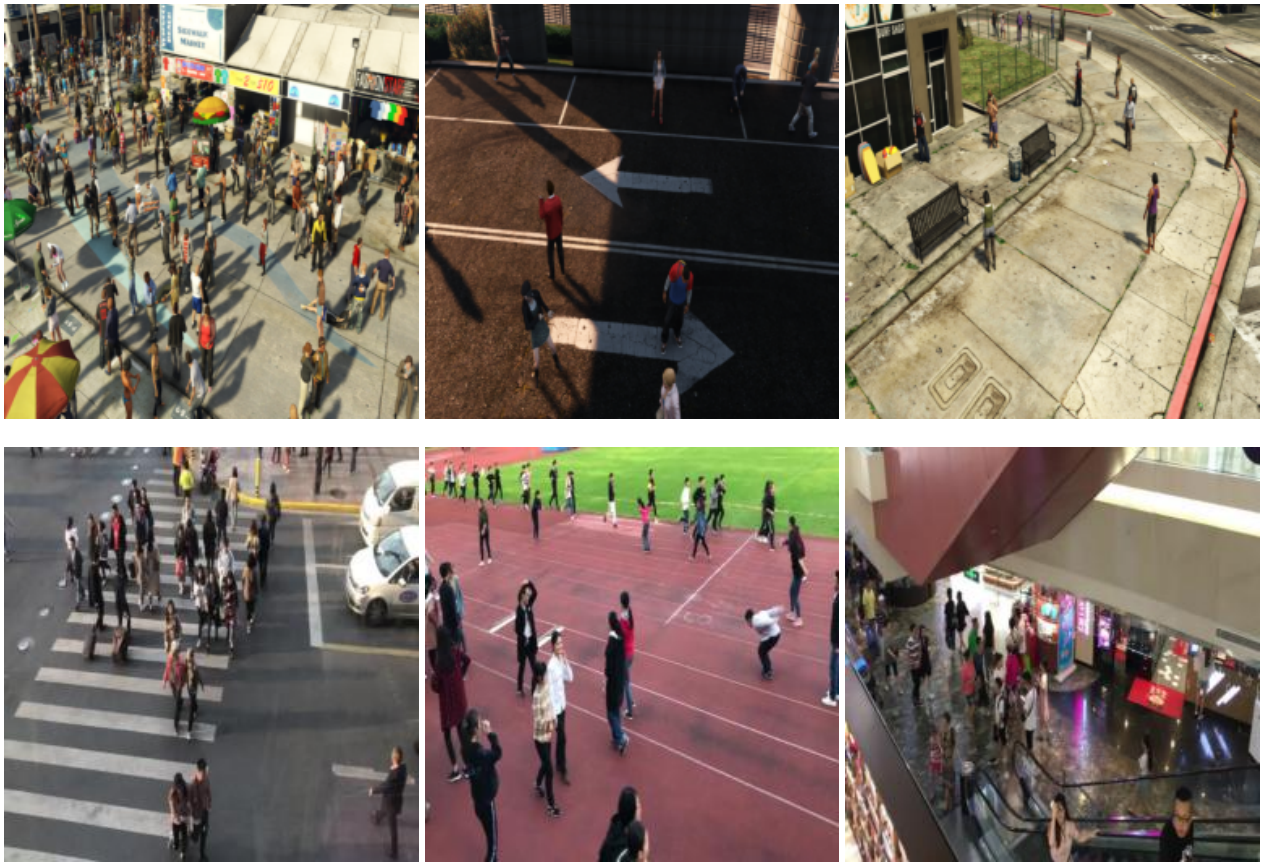

Figure 4.9: Example images used for the domain adaptation task in crowd counting: (Top) images taken from the GCC Dataset [100] and (Bottom) images taken from the FDST Dataset [21].

4.7.3 Crowd Counting

In this task, we performed domain adaptation for the crowd counting task from a dataset of synthetic imagery to a dataset of natural imagery. We used the following two datasets:

1. *GTA5 Crowd Counting(GCC) Dataset*[100]: a synthetic crowd counting dataset collected from the video game Grand Theft Auto V (GTA5). The dataset contains 15,212 images with 7,625,843 annotated heads. The images are generated in different settings including variation in location, weather condition, crowd size, etc.

2. *Fudan-ShanghaiTech(FDST) Dataset*[21]: a large-scale video crowd counting dataset with 15,000 frames. The dataset has a total of 394,081 annotated heads.

For the crowd counting experiment, we used the synthetic GCC dataset as the source domain and the FDST dataset as the target domain. Example images from the two datasets are shown in Figure 4.9. Taking the annotated heads, we generated the groudtruth density map by filtering the center points with a 2D gaussian filter with $\sigma = 1$.

We randomly sampled 80% of the source data and mixed it with the target domain making up the training set and the remaining 20% from the source domain is used as a validation set for the density estimation task. Our model is trained with the mixed training set where crowd density estimation is trained using the GCC dataset and, at the same time, adapted to the FDST dataset. Training our proposed model for this task took an average of 21hrs. We evaluated the model using the FDST dataset's test set, which is composed of 6000 images.

4.8 Summary

In this chapter, we presented an unsupervised domain adaptation technique for the object counting task. We used a density estimation based counting technique. We designed three experiments: wheat spikelet counting, leaf counting, and crowd counting. In the next chapter, we will present the results of the experiments described above.

5 Results

In this chapter, we present the performance of the proposed model on three counting tasks. We also compare these results with the baseline model and with other existing methods, where possible. Additionally, we include supervised methods where the proposed model is trained on the source datasets and fine-tuned on the target datasets. The supervised methods serve as our expected upper-bound on expected performance.

5.1 Wheat Spikelet Counting

5.1.1 ACID to GWD

Table 5.1 shows the results from the wheat spikelet counting adaptation experiment. Our method reduced the MAE by 67.8% and the RMSE by 68.7% as compared to the baseline model without adaptation. The proposed method also achieved an R^2 value of 0.66 on the target domain. The scatter plots (Figure 5.2) corroborate the performance evaluated by the R^2 value. It also shows that the baseline model's predictions generally overestimated the number of spikelets in a given image. Figure 5.1 (left) shows the training and validation loss as our proposed model converges. Figure 5.3 provides a qualitative comparison of example density maps output by the baseline model and the proposed model.

Training the baseline model with the source dataset and fine-tuning it on the target dataset (i.e., GWD) resulted in a 72.15% MAE drop, a 70.33% RMSE decrease, and a 0.715 point increase in R^2 compared to the baseline model without adaptation. Our method has a close performance to the fine-tuned baseline model even though our method does not require annotations in the target domain.

5.1.2 ACID to CropQuant

The results of evaluating our model with patches extracted from *CropQuant* dataset are presented in Table 5.1. We compared the baseline model trained exclusively on the source dataset, methods presented in Alkhudaydi et al. [2], and our proposed method. Our method achieved a 59.3% and 58.0% decrease in MAE and RMSE respectively compared to the baseline model. It also had a 2.13 point increase in R^2. However, the negative R^2 value of the baseline (No Adaptation) model shows the model has a poor performance; therefore, the comparison to SpikeletFCN [2] is more appropriate. Figure 5.1 (right) shows the training and validation loss as our proposed model is trained. Figure 5.4 shows the predictions from our adapted model have stronger correlations with the true prediction compared to the baseline model.

Table 5.1: Domain adaptation results for the wheat spikelet counting task.

	MAE	$RMSE$	R^2
ACID to GWD			
Baseline (No Adaptation)	91.65	114.4	0.005
Baseline (Fine-tuned)	25.52	33.94	0.72
Our method	29.48	35.80	0.66
ACID to CropQuant			
Baseline (No Adaptation)	443.61	547.98	-1.70
SpikeletFCN[2] (Scratch)	498.0	543.5	-
SpikeletFCN[2] (Fine-tuned)	77.12	107.1	-
Baseline (Fine-tuned)	104.15	136.14	0.48
Our method	180.69	230.12	0.43

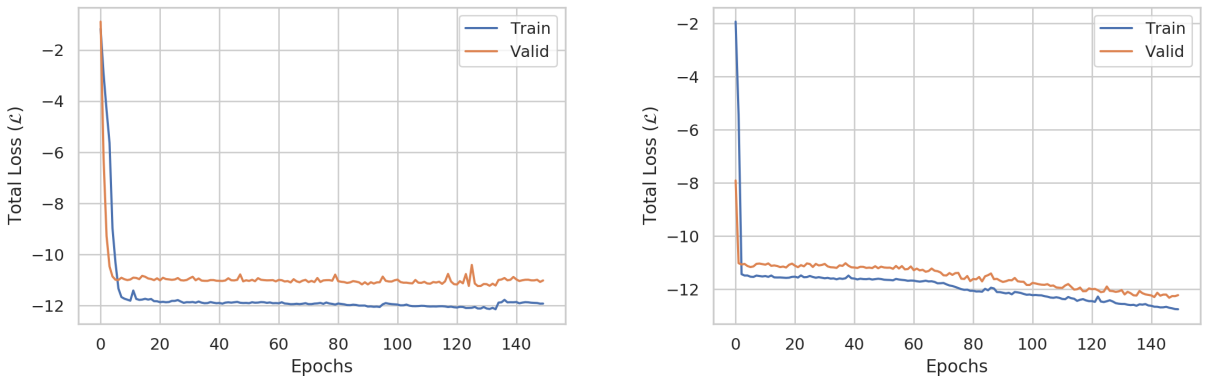

Figure 5.1: Training and validation loss for (left) the **ACID to GWD** and (right) **ACID to CropQuant** domain adaptation experiments

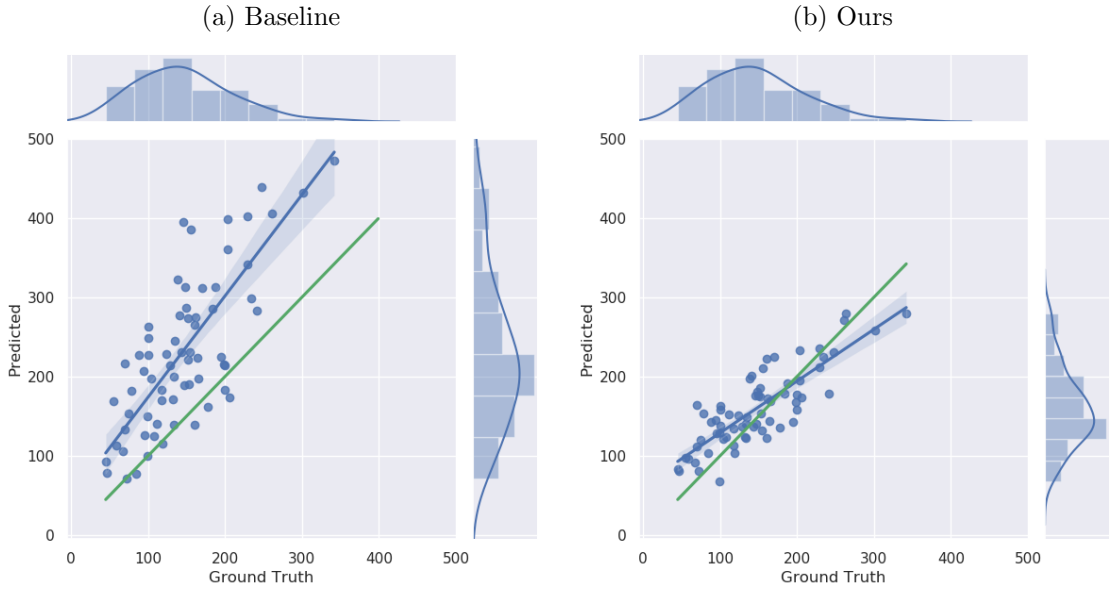

Figure 5.2: Observed vs. Predicted scatter plot for the **ACID to GWD** domain adaptation task for (left) the baseline model and (right) our proposed method.

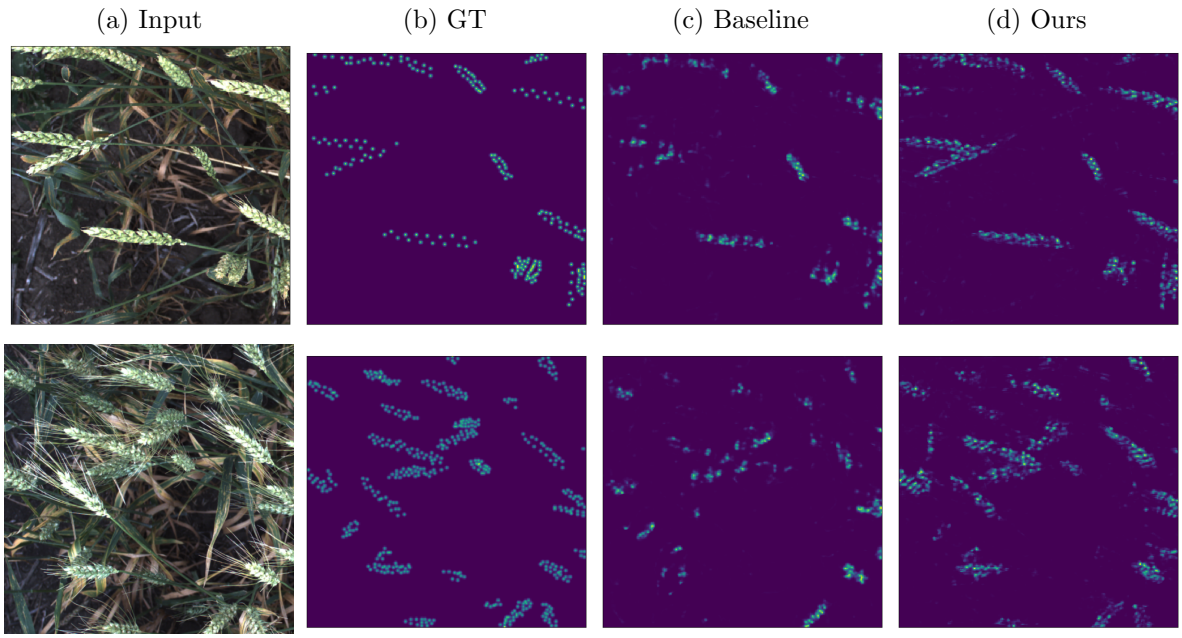

Figure 5.3: Qualitative results for density map estimation. Test images were sampled from the Global Wheat Dataset. (a) Input images. (b) Ground truth. (c) Predicted density map from the baseline model. (d) Predicted density map from the proposed model.

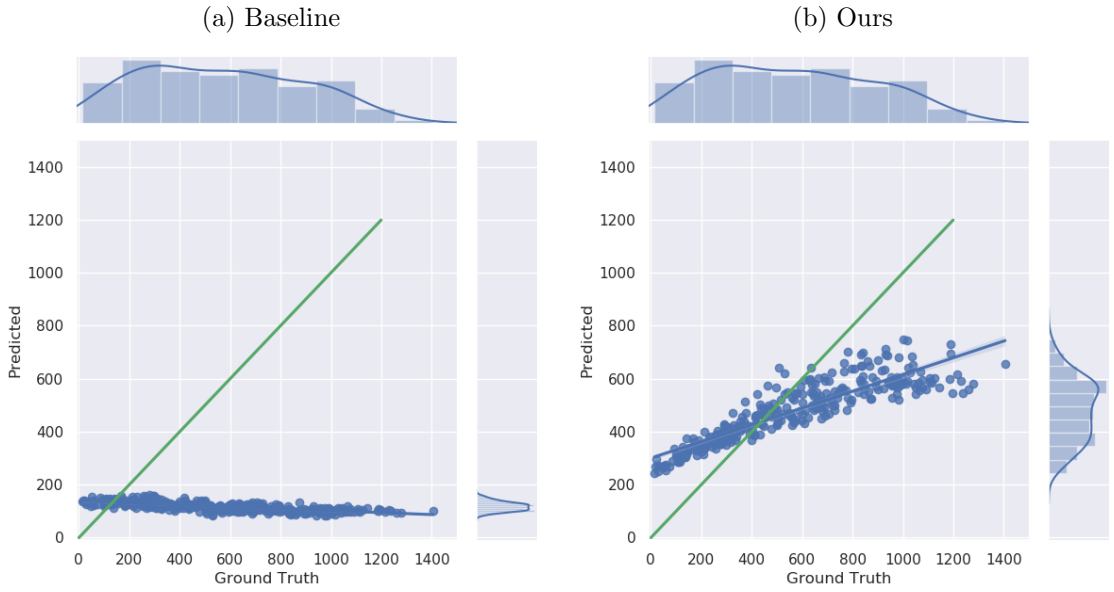

Figure 5.4: Observed vs. Predicted scatter plot for the **ACID to CropQuant** domain adaptation task for (left) the baseline model and (right) our proposed method.

The experiments presented in Alkhudaydi et al. [2] are supervised methods where one was trained solely on the target domain (i.e., scratch), and the other was pre-trained on the ACID dataset and fine-tuned on the target domain (i.e., Fine-tuned) Our model outperformed one of the supervised methods — trained from scratch on patches extracted from the target dataset — presented in Alkhudaydi et al. [2] by 63.72%. Their fine-tuned model has a better performance than our model. However, their methods require annotated datasets in the target domain while ours does not. Figure 5.5 shows the density map estimated by the baseline model and our proposed model.

As an expected upper-bound model, we trained the baseline model with the source dataset and fine-tuned it on the target dataset. This model resulted in a 76.52% and 75.16% decrease in MAE and RMSE, respectively. It also achieved a 2.18 point increase in R^2.

5.2 Leaf counting

5.2.1 CVPPP to KOMATSUNA

We report results from the baseline model trained without adaptation, a previously proposed domain adaptation method [94], and our proposed method. Because of the absence of dot annotations for the KOMATSUNA dataset, we have no fine-tuned baseline model for this experiment. Figure 5.6 (left) shows the training and validation loss when our proposed model is trained for the CVPPP to KOMATSUNA domain adaptation

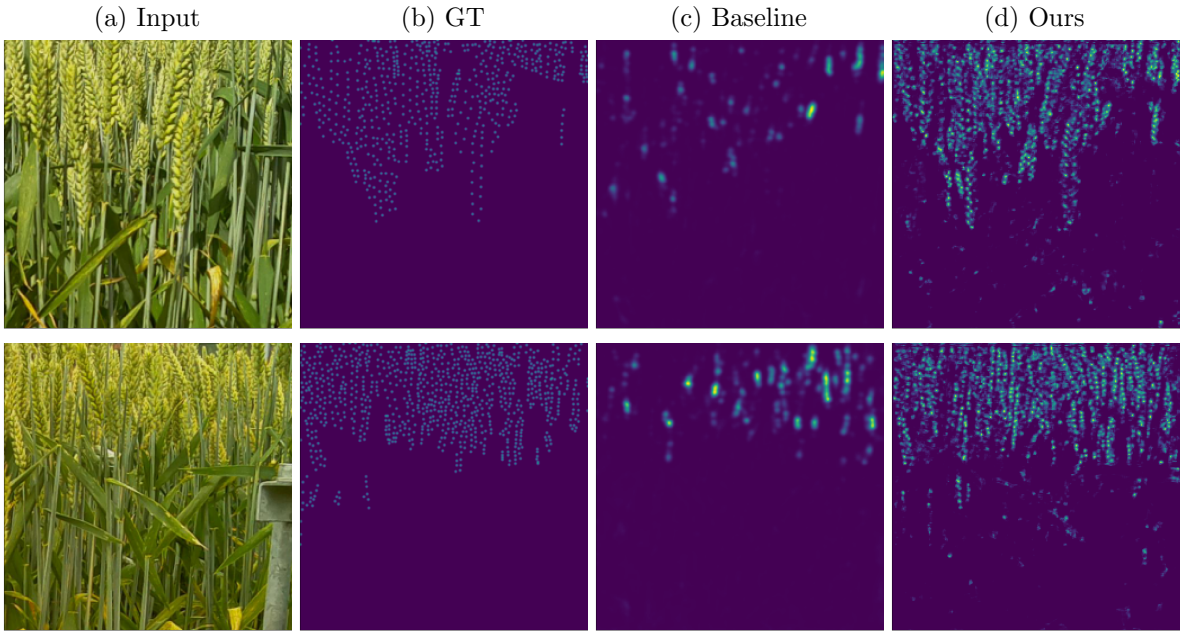

Figure 5.5: Qualitative results for density map estimation. Test images were 512×512 patches extracted from CropQuant dataset. (a) Input images. (b) Ground truth. (c) Predicted density map from the baseline model. (d) Predicted density map from the proposed model.

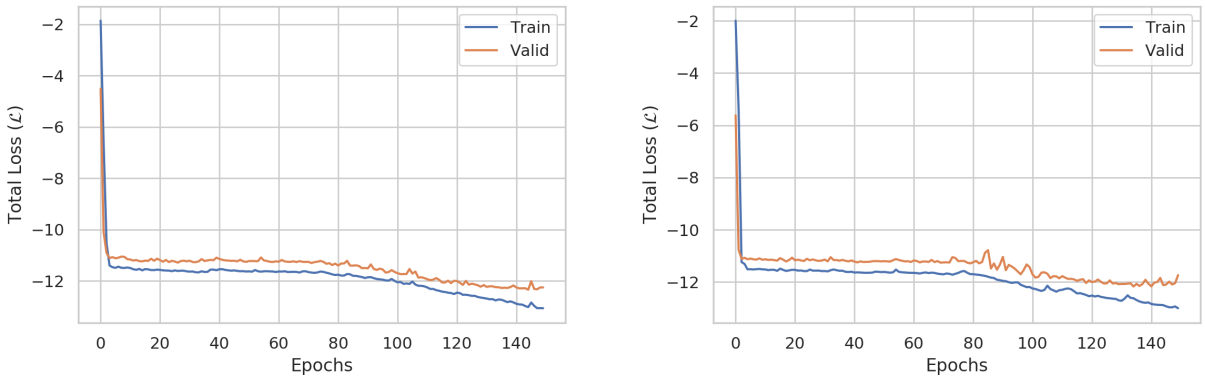

Figure 5.6: Training and validation loss for (left) the **CVPPP to KOMATSUNA** and (right) **CVPPP to MSU-PID** domain adaptation experiments

Table 5.2: Leaf counting results for the leaf counting tasks. XE and LS represent cross entropy and least-squares adversarial losses used in the experiments presented in [94]. ↓ denotes lower is better, ↑ denotes higher is better.

	$DiC \downarrow$	$\lvert DiC \rvert \downarrow$	$\% \uparrow$	$MSE \downarrow$
CVPPP to KOMATSUNA				
Baseline (No Adaptation)	4.09 (1.32)	4.09 (1.32)	0	18.49
Giuffrida, et al. [94] (XE)	-0.78 (1.12)	1.04 (0.87)	26	1.84
Giuffrida, et al. [94] (LS)	-3.72 (1.93)	3.72 (1.93)	2	17.5
Our method	-0.95 (2.09)	1.56 (1.67)	29.33	5.26
CVPPP to MSU-PID				
Baseline (No Adaptation)	1.21 (2.04)	1.83 (1.52)	0	5.65
Giuffrida, et al. [94] (XE)	-0.81 (2.03)	1.68 (1.39)	20	4.78
Giuffrida, et al. [94] (LS)	-0.39 (1.49)	1.18 (0.98)	26	2.36
Our method	1.17 (1.85)	1.63 (1.62)	23.96	4.25

experiment. The adapted model resulted in a 71.6% drop in MSE, while the percentage agreement (%) increased from 0% to 29.33% (Table 5.2) compared to the baseline trained exclusively on the CVPPP dataset. Figure 5.7 presents a scatter plot of actual and predicted leaf counts from the baseline model and our proposed model.

Additionally, we compared the proposed approach with a previous domain adaptation technique proposed by Valerio Giuffrida et al. [94]. Our method outperformed the previous result when using the least-squares adversarial loss (LS), and came close to similar performance when using the cross-entropy loss (XE). Figure 5.8 shows sample images taken from the CVPPP and KOMOTSUNA datasets, showing that the proposed method accurately positions the gaussian kernel around the center of each leaf.

5.2.2 CVPPP to MSU-PID

Our method outperforms the no-adaptation baseline model in all metrics (Table 5.2). The proposed model provided a 23.96 point increase in percentage agreement (%) and a 10.93% decrease in $|DiC|$. The proposed method outperformed the previous domain adaptation technique in this task when using the least-squares (LS) loss function, but under-performed it when the cross-entropy (XE) loss is used. As can be seen from the scatter plots in Figure 5.9, the baseline model (without adaptation) undercounts leaves. On the other hand, except for some anomalies (discussed in Chapter 6), our proposed model's predictions are close to the ground truth. Figure 5.6 (right) shows the training and validation loss when our proposed model is trained for the CVPPP to MSU-PID domain adaptation experiment.

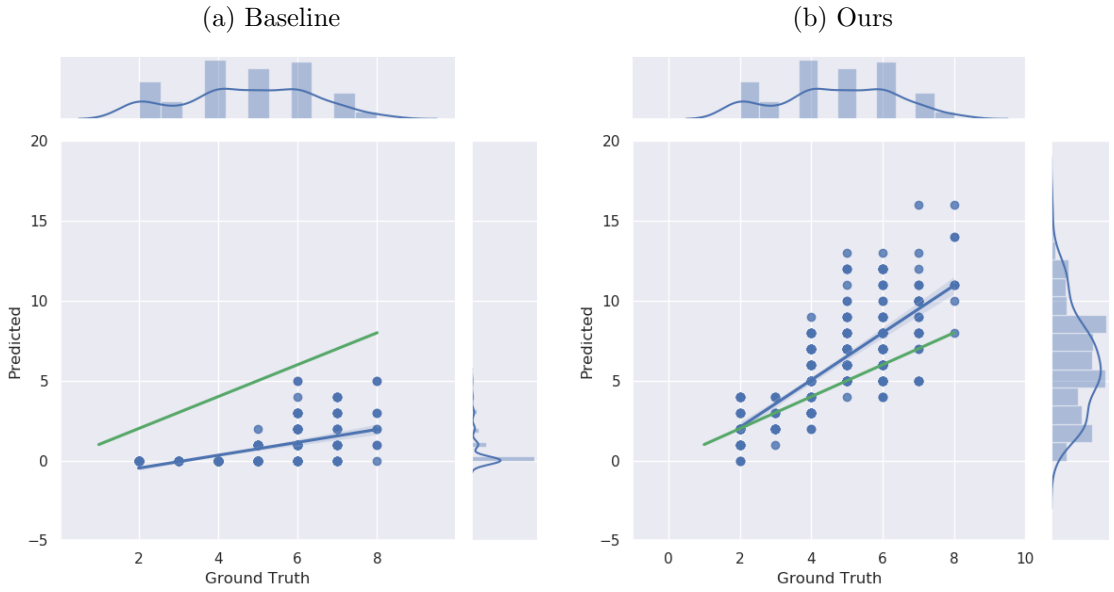

Figure 5.7: Observed vs. Predicted scatter plot for the **CVPPP to KOMATSUNA** domain adaptation task for (left) the baseline model and (right) our proposed method.

5.2.3 Composite-KOMATSUNA

Without retraining the model from the *CVPPP to KOMATSUNA* experiment, we evaluated it on our composite images. It resulted in a mean DiC of 9.89 and mean $|DiC|$ of 9.90. Figure 5.10 shows a sample output from our model on the synthetic dataset. The results from this experiment validates that our model does not have any assumption about distribution of the object counts in the source domain. The method proposed by Valerio Giuffrida et al. [94] assumes that the distribution of leaves in the source domain would be similar to that of the target domain. This assumption was included to avoid posterior collapse (also called mode collapse). The authors used the KL divergence cost function to estimate the difference between the distribution of leaf counts in the source domain and their proposed model predictions. The target of including this cost function is to align the two distributions.

5.3 Crowd Counting

5.3.1 GCC to FDST

We compared our results with the baseline model solely trained on the source dataset, Locality-constrained Spatial Transformer Network (LSTN) [21], the baseline model fine-tuned with the target data, and our proposed model. With the proposed method we observed a 51.4% decrease in MAE and a 39.1% decrease in RMSE compared to the baseline model without domain adaptation (Table 5.3). Figure 5.11 presents a

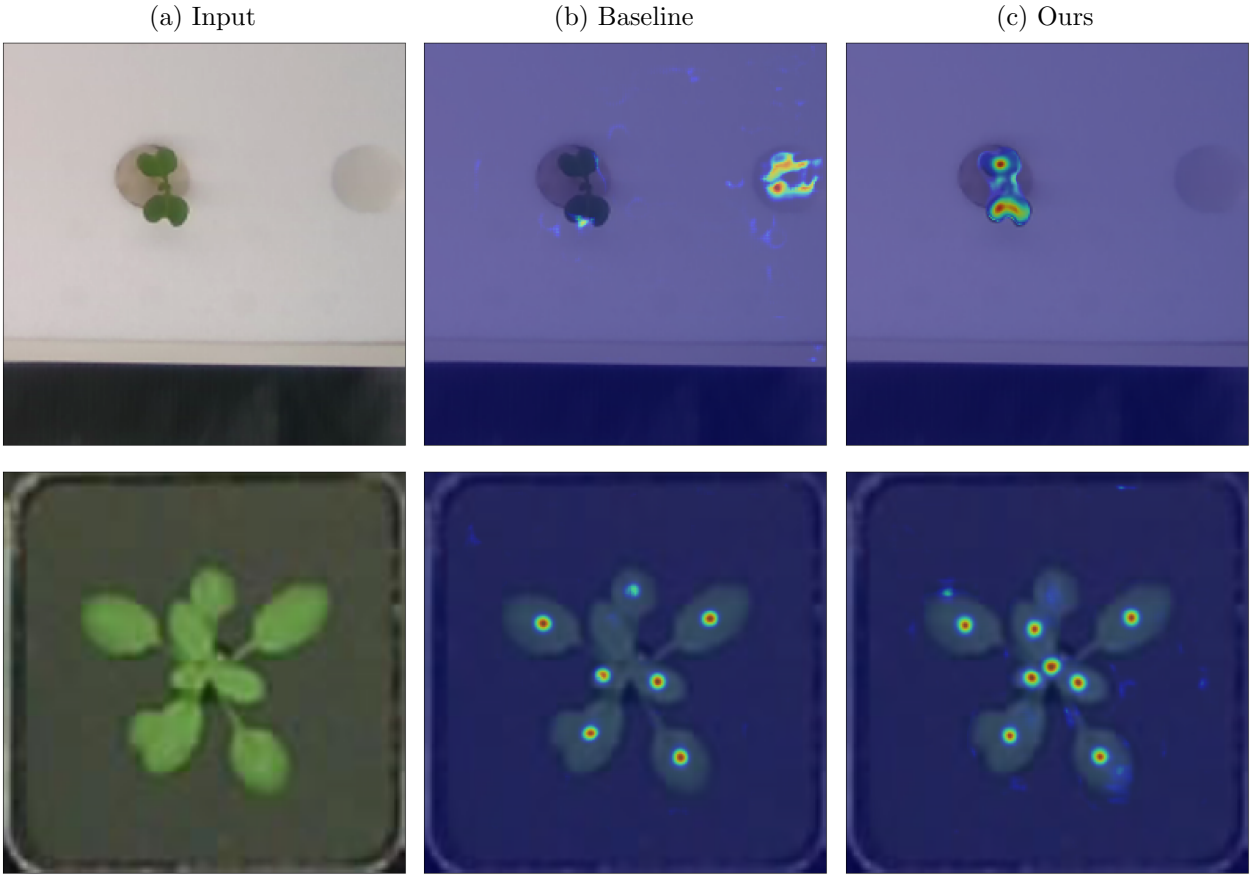

Figure 5.8: Sample density map estimations from the baseline model and from the adapted model in the leaf counting task. Top row: **CVPPP to KOMATSUNA** experiment. Bottom row: **CVPPP to MSU-PID** experiment

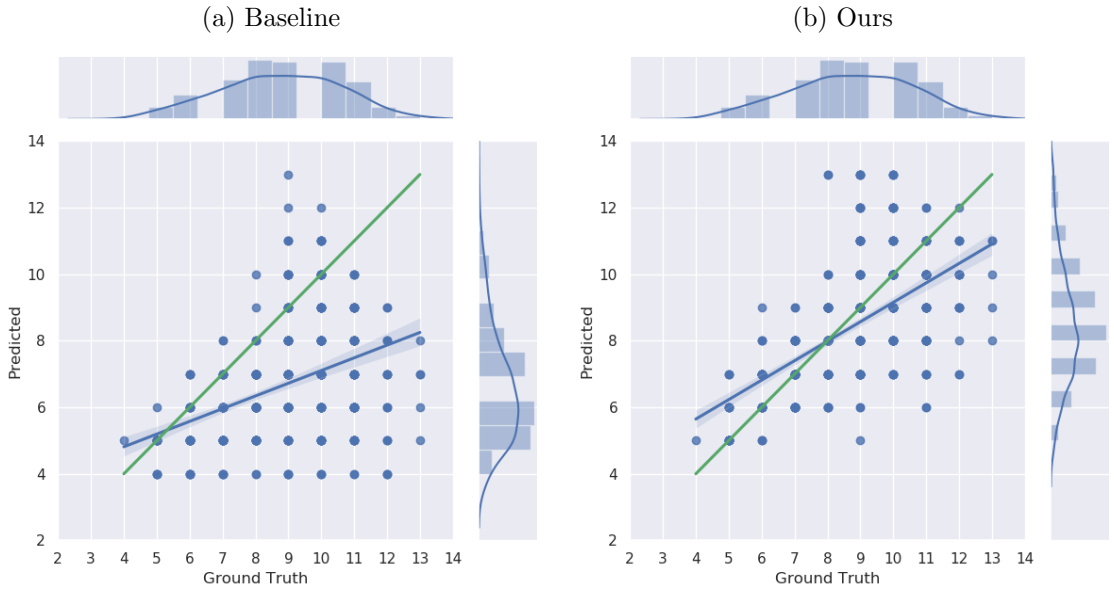

Figure 5.9: Observed vs. Predicted scatter plot for the **CVPPP to MSU-PID** domain adaptation task for (left) the baseline model and (right) our proposed method.

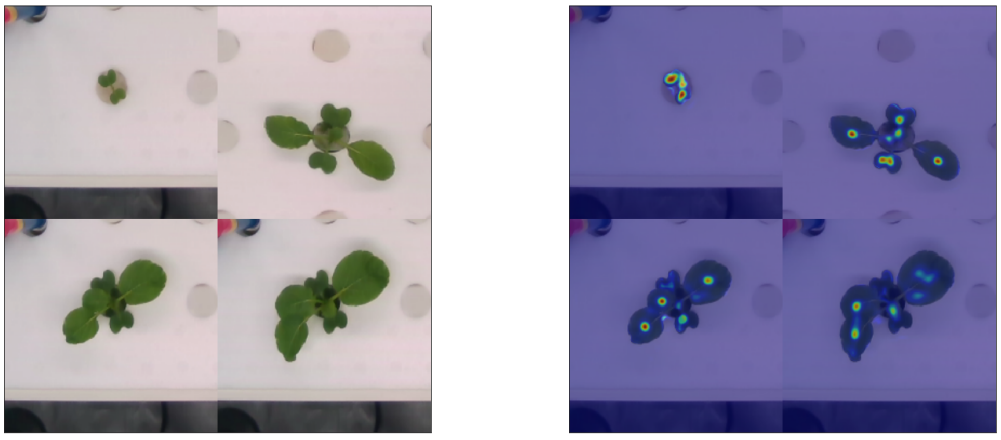

Figure 5.10: Density map estimation for a sample image taken from the composite dataset without retraining the model trained to adapt leaf counting using domain and target datasets with one plant per image. (left) A sample input image and (right) is the density map prediction of our model overlaid on top of the input image

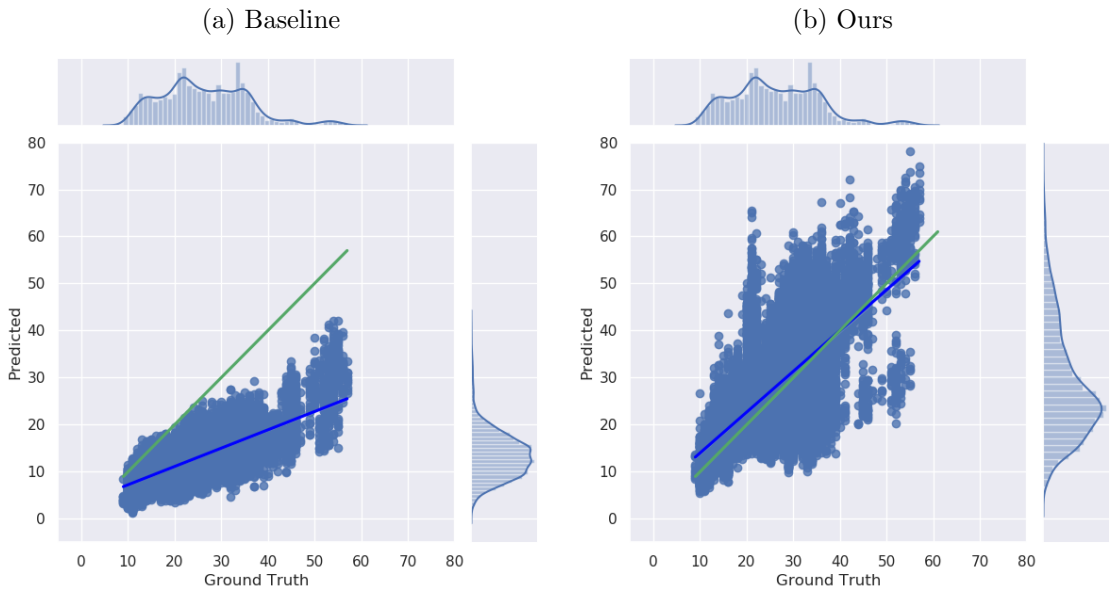

Figure 5.11: Observed vs. Predicted scatter plot for the **GCC to FDST** domain adaptation task for (left) the baseline model and (right) our proposed method.

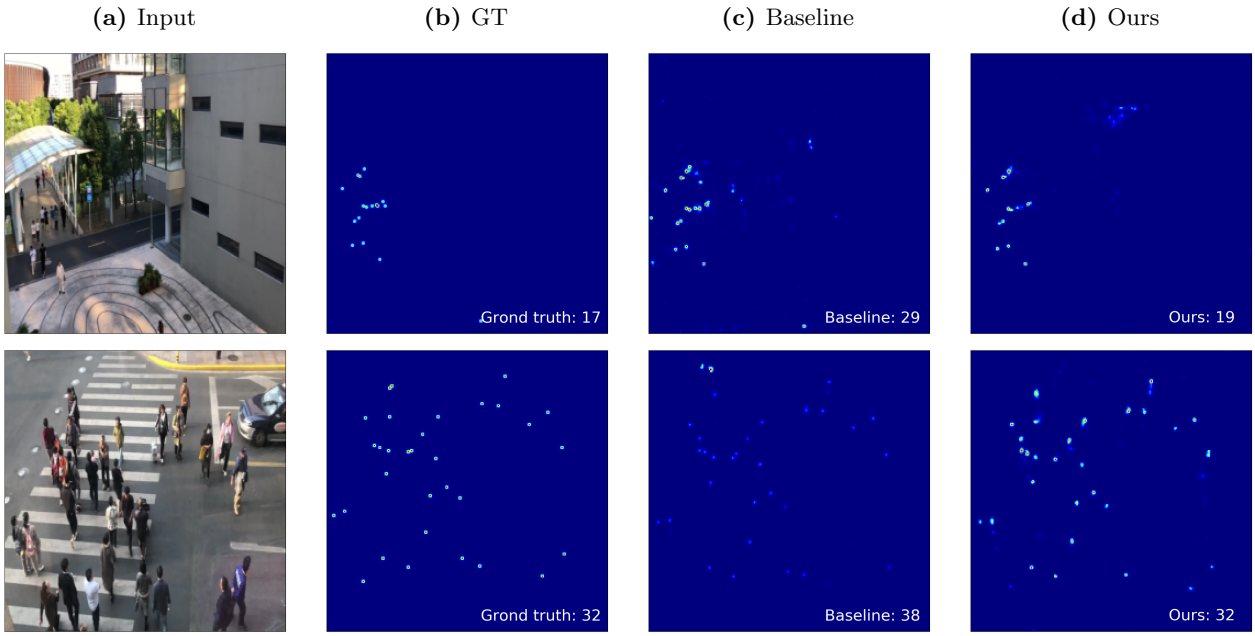

Figure 5.12: Example results demonstrating the proposed domain adaptation technique in a crowd counting task from *GCC* to *FDST*. (a) Input image. (b) Ground truth. (c) Density map predicted by the proposed model.

Table 5.3: Domain adaptation results for the crowd counting task.

	MAE	$RMSE$
Baseline (No Adapt)	12.72	14.33
Our method	6.18	8.37
LSTN [21]	3.35	4.45
Baseline (Fine-tuned)	0.71	1.02

scatter plot of the ground truth and predicted values for the target dataset. It shows that the baseline model generally undercounts the number of people in an image. Figure 5.12 shows the qualitative results of the adapted model on sample images taken from FDST.

Locality-constrained Spatial Transformer Network (LSTN) [21] is a supervised density-based crowd counting method proposed to count people from video-frames. LSTN leverages a CNN followed by a Locality-constrained Spatial Transformer to account for the sequential nature of the video frames. Their method outperformed our proposed model by lowering both the MAE and RMSE. However, this method requires annotation from FDST while ours does not.

The baseline (Fine-tuned) model is initially trained with the source domain dataset and fine-tuned with the target domain data. This model lowered the MAE and RMSE by 94.42% and 92.88%, respectively. The fine-tuned baseline model has also outperformed LSTN by lowering the MAE by 78.8% without leveraging spatiotemporal information.

6 Discussion and Future Work

Deep learning methods, particularly Convolutional Neural Networks, have shown state-of-the-art performance in several computer vision tasks. However, in the absence of large annotated data, deep learning methods have a hard-time learning generalizable parameters, which hinders the model's performance on unseen data. Several techniques have been proposed to tackle this problem. Among these techniques, the most prominent ones are fine-tuning and domain adapatation.

This work addresses generalization issues in density-estimation-based object counting tasks, specifically the performance issue caused by domain-shift. Domain-shift is a practical problem that hinders the performance of deep learning models caused by differences in the distribution of training datasets and test datasets. In image-based applications, domain-shift can be caused by simple variations such as a change in illumination or background. We proposed an unsupervised domain adaptation technique to learn counting from a labeled dataset and, in parallel, adapts to the unlabeled data via the density-estimation based counting techniques. In the following sections, we will analyze our results, discuss the drawbacks of our method, and examine potential future directions following our work.

6.1 Discussion

In this section, we start by discussing results from each experiment, followed by presenting general observations from the experiments.

The results from Table 5.1 show our method outperforms the baseline model by a wide margin in the indoor-to-outdoor (the ACID to GWD and ACID to CropQuant) adaptation task. In the ACID to GWD adaptation task, our model achieved a closer performance to the fine-tuned model even though our proposed model does not require annotation for the target domain. The scatter plot in Figures 5.2 shows that the baseline model's predictions generally overcount the number of spikelets. However, the density maps predictions from the baseline model seem to be undercounting (as in Figure 5.3), which is counterintuitive. When we take a closer look at the predictions, each accurately localized spikelet is assigned a high density resulting in overall high density-estimation. On the other hand, Figure 5.4 shows the predictions from the baseline model tested on the CropQuant dataset generally undercount the number of spikelets, which was consistent with the quantitative results (Figure 5.5). The results from the spikelet counting experiments are empirical evidence showing that our proposed method could be used to mitigate the indoor-to-outdoor domain-shift.

The results from the species-to-species domain adaptation experiments (Table 5.2) show that our method

has a better performance than the baseline model in the leaf counting experiments, namely the intraspecies (i.e., CVPPP to MSU-PID) and interspecies (i.e., CVPPP to KOMATSUNA) domain adaptation experiments. The scatter plots from Figure 5.7 indicate that the baseline model consistently undercounted the number of leaves in the intraspecies experiment, whereas our method had a better performance. From the same figure, it can also be visually validated that our model approximated the leaf count distribution very well compared to the baseline. The scatter plot for the interspecies domain adaptation experiment (Figure 5.9) depicts the baseline model mostly, undercounts leaves. The scatter plots for the two leaf counting experiments show the leaf counting results perform well in the mean however have high variance, which is consistent with results from previous works using the Leaf Counting Competition dataset [94, 1, 18].

It can be seen from the density map estimations in Figure 5.8 that our proposed model learned to localize the leaves much better than the baseline model, especially for the tests done with the Komatsuna plant. A more interesting case that can be seen in the same figure is the performance of the shared baseline model (i.e., trained exclusively on the CVPPP dataset) on the Komatsuna sample and the MSU-PID sample. In the Komatsuna sample, the baseline model completely misses the plant, whereas, the same model successfully localized most of the leaves in the MSU-PID sample. This can be attributed to the difference in species used as the target data.

Learning to localize and count leaves in the interspecies task is a more complex task than the intraspecies given the fact that the Arabidopsis and Komatsuna plants have different shaped leaves in addition to the typical background shift. Given the differences in the complexity of the two leaf counting experiments, it can be concluded that the relative gain in performance between the shared baseline (i.e., trained exclusively on the CVPPP dataset) and adapted models has implications for quantifying the gap in the domain-shifts. Since CVPPP to MSU-PID is an intraspecies domain adapatation task, our proposed method's performance gain is much smaller than the interspecies experiment (i.e., CVPPP to KOMATSUNA).

The results from the composite-KOMATSUNA experiment (samples displayed in Figure 5.10) show that our proposed method works well even when the distribution of leaves and plants changes in the target domain. As shown in Figures 4.5, and 4.7, the distribution of objects in the source and target domain used for the experiments are different. Since our method does not align the object counts in the two distributions and our proposed architecture uses a local regression technique, we can conclude that our technique's performance will not be hindered by changes in the distribution of object counts as long as the domain-shift stays consistent.

Table 5.3 shows that our model has learned to minimize the domain-gap in synthetic-to-real domain adaptation for the crowd counting task. Our method has shown drastic improvements from the baseline model that was exclusively trained with synthetic data. The scatter plots in Figure 5.11 corroborate the results form the table showing the our adapted mode is a better predictive model than the baseline. Moreover, the scatter plots show that the baseline model's predictions generally undercount the number of heads (people). As can be seen from Figure 5.12, the adapted model is more accurate and confident in predicting position of heads compared to the baseline model.

In all of our experiments, we observed the baseline model converges faster than the proposed model. The differences in convergence speed can be mainly attributed to two significant differences: the optimization problems each of the models is tasked to solve and the difference in the number of parameters. The optimization task the baseline model is given to solve is minimizing a distance between the prediction and ground truth. On the other hand, the proposed model has to find a saddle point that minimizes a similar distance as the baseline model and maximizes the domain confusion. Furthermore, we have removed the domain classifier subnetwork for the baseline model giving it fewer parameters to optimize for than our model.

Through ad-hoc hyperparameter tuning, we have identified that the results are most sensitive to the learning rate parameter. Therefore we employed a grid search on the learning rate parameter. The relative difference between the learning rates of the three subnetworks (i.e., downsampling subnetwork (G_d), upsampling subnetwork (G_u), and domain classifier (G_c)) affects the competition to decrease the total cost while increasing the domain classification loss. The learning rates identified with the grid search worked consistently for all the experiments.

The results from fine-tuning the baseline model on the target-domain dataset have better performance than unsupervised domain adaptation. However, fine-tuning a model requires sufficient annotated target-domain datasets. The number of annotated samples required from the target-domain is less than the source-domain dataset required for initial training. To get the best performance from fine-tuning, we need to provide annotated data sampled from the dataset expected to be seen when the models are deployed. On the other hand, exclusively training on a target dataset requires the existence of a large annotated dataset. Creating or acquiring such datasets is costly. Therefore, in real-world applications, fine-tuning models trained on preexisting data is the preferred approach.

Domain adaptation has most potential in scenarios where a model trained with existing data experiences poor performance when deployed and when the expense of annotating data for the target task outweighs the model's performance requirement. In such situations, using domain adaptation techniques improves the performance of the model. The usefulness of an adapted model depends on the accuracy requirement of the particular task. For instance, our proposed model adapted for the wheat spikelet counting task can be used by wheat breeders to discriminate between different wheat genotypes based on their wheat spikelet density.

In conclusion, the results from all the experiments in the three tasks demonstrate that our method learns to minimize domain-shifts without the need for labeled data in the target domain. We also have shown that our method does not assume the alignment of object distributions in the source and target domain data.

6.2 Limitations and Future Work

Although the evaluations from each of our experiments show encouraging results, our proposed model has several limitations. This section discusses these weaknesses, starting with general drawbacks followed by task-specific ones. For the limitations, we try to propose potential directions as future work. Finally, we

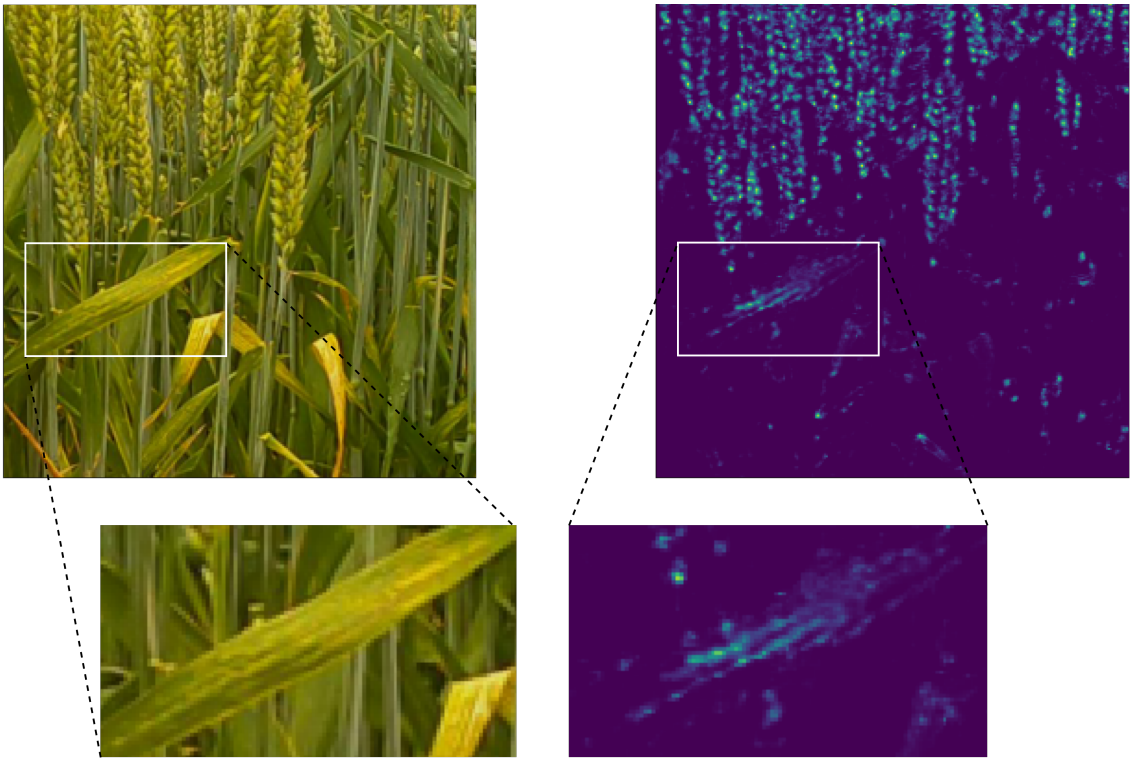

Figure 6.1: Sample issue on the **ACID to CropQuant** adapted model where spots on a discolored leaf (zoomed-in) are predicted as spikelets

present possible extensions of our work to different domain adaptation tasks and experiments.

Commonly counting applications that are modeled with a density-based method try to minimize a density cost making the task a minimization problem. However, our proposed method is tasked to minimize the density loss ($\mathcal{L}_{density}$) and additionally maximize the domain classification loss ($\mathcal{L}_{domain}$). The extra loss in our method removes viable solutions from the solution set of density-loss-only objectives, lowering our model's performance in the source domain. A potential solution to avoid such problems is to try to create separate representations for each domain with shared feature extraction.

In all of our experiments, we assumed the existence of balanced source and target domain data. In cases where class imbalance exists among the two classes, our method's performance might drop. However, in such scenarios employing techniques like oversampling or replacing the binary cross-entropy with weighted cost functions such as Focal loss [53] could alleviate the problem.

After adapting the wheat counting density estimation form the ACID to CropQuant domain adaptation experiment, we noticed that the density map predictions for patches that have leaves with partial discoloration contained high-density predictions for spots in the leaves. As can be seen in the zoomed-in part of Figure 6.1, high-density values are given to the spots on a leaf. A potential explanation for the cause of the problem is the color similarities between the spikelets and the spots on the partially discolored leaves in the UK spring

(a) Input (b) Prediction

Figure 6.2: Sample counting and localization issues from the interspecies experiment (i.e., **CVPPP to KOMATSUNA**)

(a) Input (b) Prediction (c) Prediction

Figure 6.3: Density estimation issues arising due to perspective effects

wheat genotype used in the CropQuant experiment. A possible solution to overcome this problem can be annotating a small subset of the patches to be provided as a label in the training steps.

Figure 6.2 shows two typical issues that occur after adapting our proposed model in the interspecies experiment: completely missed leaves and overcounted leaves. The adapted model misses heart-shaped leaves and wilted leaves, and it overcounts broad leaves. These problems can be attributed to the shape and size differences of the leaves in the source dataset (i.e., Arabidopsis) and target dataset (i.e., Komatsuna).

In the crowd counting task, our proposed method has performance issues due to scale variation in people (or head) sizes caused by the perspective effect. As can be seen from the density map prediction in Figure 6.3, the model has a harder time detecting people close to the camera than those far from the camera. This problem can potentially be resolved by modifying our proposed architecture to include multi-scale feature extraction by employing different size kernels or Feature Pyramid Network (FPN).

Using synthetic data is becoming popular for image-based plant phenotyping [102, 93, 75]. Since it is less expensive to generate synthetic images with various annotations, it is convenient to train CNNs with such datasets. However, the domain-gap between synthetic and real-world data would require adapting the trained models to the latter. Therefore, domain adaptation for synthetic plant images is a natural extension to our proposed method.

In addition, our proposed method can potentially be applied to similar counting tasks with different domain-shifts such as cross-modality domain adaptation tasks. Cross-modality domain adaptation tasks learn from datasets collected using one sensor to adapt the knowledge to a dataset collected with a different sensor, which can be employed for tasks such as learning to count plant organs across multiple imaging sensors. Our method can be applied to these tasks without changing any of the hyperparameters or losses. Additionally, our method could also be easily extended to multi-target domain adaptation tasks, where we would have a set of target datasets sampled from multiple distributions.

Theoretically, our method could also be employed to domain adaptation in image segmentation tasks. However, segmentation tasks require modifications to the loss functions, output layers in the upsampling network, and tuning hyperparameters. This could be explored in future work for leaf or organ segmentation to estimate shape and size in detail.

7 Conclusion

In this thesis, we attempted to address performance problems that arise in object counting due to domain-shift. Our study focuses specifically on density based object counting, which is a common method for plant organ counting and crowd counting tasks. We proposed a custom Domain-Adversarial Neural Network architecture for domain adaptation in density-based object counting. We trained the network using an adversarial learning framework in an unsupervised domain adaptation setting without using any annotation in the target datasets.

Our evaluation showed performance improvements compared to the baseline models trained without domain adaptation for wheat spikelet, leaf, and crowd counting. Each of these tasks addresses different domain-shift issues: indoor-to-outdoor, species-to-species, and synthetic-to-real. For leaf counting, our results show similar performance to a previously proposed domain adaptation approach without the need to switch loss functions between datasets. Our study is the first to investigate the use of domain adaptation from an indoor source dataset to an outdoor target dataset. This may be a viable method for many plant phenotyping contexts, such as plant breeding experiments with controlled environments and field trials useful because indoor images are easier to annotate. Additionally, we have shown that our method could adapt density estimation from synthetic data to real-world data. The ability to adapt from synthetic-to-real generally helps minimize the effort needed to annotate data since it is cheaper to get annotations for synthetic data.

7.1 Contributions

In this thesis, we present an extension of a domain-adversarial technique for unsupervised domain adaptation for density map estimation. We proposed two fully convolutional networks with shared layers designed to learn domain-invariant features. We include detailed discussions of our proposed method, including our customized architecture, how we tuned the hyperparameters, and the training scheme. Following these discussions, we provide a detailed analysis of the results and drawbacks of our proposed model.

We evaluated our method on three diverse counting tasks: wheat spikelet counting, leaf counting, and crowd counting. Each of these tasks is selected to have different domain-shift: the wheat spikelet counting has indoor-to-outdoor, the leaf counting has a species-to-species, and the crowd counting task has synthetic-to-real. To the best of our knowledge, this is the first work to apply domain adaptation for indoor-to-outdoor spikelet counting.

Another contribuiton of this thesis is a new public dataset with ground truth for outdoor wheat spikelet

counting task taking the Usask_1 subset of the Global Wheat Head Dataset. The annotated dataset can be used to train and test machine learning models for wheat spikelet localization and counting tasks. Using this dataset, we have created baseline models for domain adaptation and supervised spikelet counting tasks.

References

[1] Shubhra Aich and Ian Stavness. Leaf counting with deep convolutional and deconvolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 2080–2089, 2017.

[2] Tahani Alkhudaydi, Ji Zhou, et al. Spikeletfcn: Counting spikelets from infield wheat crop images using fully convolutional networks. In *International Conference on Artificial Intelligence and Soft Computing*, pages 3–13. Springer, 2019.

[3] Senjian An, Wanquan Liu, and Svetha Venkatesh. Face recognition using kernel ridge regression. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7. IEEE, 2007.

[4] Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (gans). *arXiv preprint arXiv:1703.00573*, 2017.

[5] C. Arteta, V. Lempitsky, and A. Zisserman. Counting in the wild. In *European Conference on Computer Vision*, 2016.

[6] Deepak Babu Sam, Shiv Surya, and R Venkatesh Babu. Switching convolutional neural network for crowd counting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5744–5752, 2017.

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

[8] Enrico Bellocchio, Gabriele Costante, Silvia Cascianelli, Mario Luca Fravolini, and Paolo Valigi. Combining domain adaptation and spatial consistency for unseen fruits counting: a quasi-unsupervised approach. *IEEE Robotics and Automation Letters*, 5(2):1079–1086, 2020.

[9] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

[10] John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman. Learning bounds for domain adaptation. *Advances in neural information processing systems*, 20:129–136, 2007.

[11] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3722–3731, 2017.

[12] Antoni B Chan, Zhang-Sheng John Liang, and Nuno Vasconcelos. Privacy preserving crowd monitoring: Counting people without people models or tracking. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7. IEEE, 2008.

[13] Zhi-Qi Cheng, Jun-Xiu Li, Qi Dai, Xiao Wu, Jun-Yan He, and Alexander G Hauptmann. Improving the learning of multi-column convolutional neural network for crowd counting. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 1897–1906, 2019.

[14] Jaehoon Choi, Taekyung Kim, and Changick Kim. Self-ensembling with gan-based data augmentation for domain adaptation in semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 6830–6840, 2019.

[15] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8789–8797, 2018.

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

[17] Etienne David, Simon Madec, Pouria Sadeghi-Tehran, Helge Aasen, Bangyou Zheng, Shouyang Liu, Norbert Kirchgessner, Goro Ishikawa, Koichi Nagasawa, Minhajul Arifin Badhon, et al. Global wheat head detection (gwhd) dataset: a large and diverse dataset of high resolution rgb labelled images to develop and benchmark wheat head detection methods. *arXiv preprint arXiv:2005.02162*, 2020.

[18] Andrei Dobrescu, Mario Valerio Giuffrida, and Sotirios A Tsaftaris. Leveraging multiple datasets for deep leaf counting. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 2072–2079, 2017.

[19] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014.

[20] Hady Elsahar and Matthias Gallé. To annotate or not? predicting performance drop under domain shift. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2163–2173, 2019.

[21] Yanyan Fang, Biyun Zhan, Wandi Cai, Shenghua Gao, and Bo Hu. Locality-constrained spatial transformer network for video crowd counting. *arXiv preprint arXiv:1907.07911*, 2019.

[22] Min Fu, Pei Xu, Xudong Li, Qihe Liu, Mao Ye, and Ce Zhu. Fast crowd density estimation with convolutional neural networks. *Engineering Applications of Artificial Intelligence*, 43:81–88, 2015.

[23] Hao Gan, Won Suk Lee, Natalia Peres, and Clyde Fraisse. Development of a multi-angle imaging system for automatic straw-berry flower counting. 2019.

[24] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189, 2015.

[25] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[26] Guangshuai Gao, Junyu Gao, Qingjie Liu, Qi Wang, and Yunhong Wang. Cnn-based density estimation and crowd counting: A survey. *arXiv preprint arXiv:2003.12783*, 2020.

[27] Sambuddha Ghosal, Bangyou Zheng, Scott C Chapman, Andries B Potgieter, David R Jordan, Xuemin Wang, Asheesh K Singh, Arti Singh, Masayuki Hirafuji, Seishi Ninomiya, et al. A weakly supervised deep learning framework for sorghum head detection and counting. *Plant Phenomics*, 2019:1525874, 2019.

[28] Jonathon A Gibbs, Alexandra J Burgess, Michael P Pound, Tony P Pridmore, and Erik H Murchie. Recovering wind-induced plant motion in dense field environments via deep learning and multiple object tracking. *Plant physiology*, 181(1):28–42, 2019.

[29] M Valerio Giuffrida, Feng Chen, Hanno Scharr, and Sotirios A Tsaftaris. Citizen crowds and experts: observer variability in image-based plant phenotyping. *Plant methods*, 14(1):12, 2018.

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

[31] Mario Valerio Giuffrida, Peter Doerner, and Sotirios A Tsaftaris. Pheno-deep counter: a unified and versatile deep learning architecture for leaf counting. *The Plant Journal*, 96(4):880–890, 2018.

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

[33] Te Han, Chao Liu, Wenguang Yang, and Dongxiang Jiang. A novel adversarial learning framework in deep convolutional neural network for intelligent diagnosis of mechanical faults. *Knowledge-Based Systems*, 165:474–487, 2019.

[34] Md Mehedi Hasan, Joshua P Chopin, Hamid Laga, and Stanley J Miklavcic. Detection and analysis of wheat spikes using convolutional neural networks. *Plant Methods*, 14(1):100, 2018.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[36] Carlos X Hernández, Mohammad M Sultan, and Vijay S Pande. Using deep learning for segmentation and counting within microscopy data. *arXiv preprint arXiv:1802.10548*, 2018.

[37] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998. PMLR, 2018.

[38] Haroon Idrees, Imran Saleemi, Cody Seibert, and Mubarak Shah. Multi-source multi-scale counting in extremely dense crowd images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2547–2554, 2013.

[39] Yotam Itzhaky, Guy Farjon, Faina Khoroshevsky, Alon Shpigler, and Aharon Bar-Hillel. Leaf counting: Multiple scale regression and detection using deep cnns. In *BMVC*, page 328, 2018.

[40] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[41] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[42] Piotr Koniusz, Yusuf Tas, and Fatih Porikli. Domain adaptation by mixture of alignments of second-or higher-order scatter tensors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4478–4487, 2017.

[43] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[44] Dmitry Kuznichov, Alon Zvirin, Yaron Honen, and Ron Kimmel. Data augmentation for leaf segmentation and counting tasks in rosette plants. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[45] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[46] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[47] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[48] Bastian Leibe, Edgar Seemann, and Bernt Schiele. Pedestrian detection in crowded scenes. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 878–885. IEEE, 2005.

[49] Victor Lempitsky and Andrew Zisserman. Learning to count objects in images. In *Advances in neural information processing systems*, pages 1324–1332, 2010.

[50] Min Li, Zhaoxiang Zhang, Kaiqi Huang, and Tieniu Tan. Estimating the number of people in crowded scenes by mid based foreground segmentation and head-shoulder detection. In *2008 19th International Conference on Pattern Recognition*, pages 1–4. IEEE, 2008.

[51] Wang Li, Li Yongbo, and Xue Xiangyang. Coda: Counting objects via scale-aware adversarial density adaption. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 193–198. IEEE, 2019.

[52] Ping Lin and Yongming Chen. Detection of strawberry flowers in outdoor field by deep neural network. In *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, pages 482–486. IEEE, 2018.

[53] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[54] Ming Liu, Jue Jiang, Zhenqei Guo, Zenan Wang, and Yang Liu. Crowd counting with fully convolutional neural network. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 953–957. IEEE, 2018.

[55] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *Advances in neural information processing systems*, pages 469–477, 2016.

[56] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[57] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105, 2015.

[58] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2208–2217. JMLR. org, 2017.

[59] Hao Lu, Zhiguo Cao, Yang Xiao, Bohan Zhuang, and Chunhua Shen. Tasselnet: counting maize tassels in the wild via local counts regression network. *Plant methods*, 13(1):79, 2017.

[60] Yawei Luo, Liang Zheng, Tao Guan, Junqing Yu, and Yi Yang. Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2507–2516, 2019.

[61] Simon Madec, Xiuliang Jin, Hao Lu, Benoit De Solan, Shouyang Liu, Florent Duyme, Emmanuelle Heritier, and Frédéric Baret. Ear density estimation from high resolution rgb imagery using deep learning technique. *Agricultural and forest meteorology*, 264:225–234, 2019.

[62] Jacob Menick and Nal Kalchbrenner. Generating high fidelity images with subscale pixel networks and multidimensional upscaling. *arXiv preprint arXiv:1812.01608*, 2018.

[63] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[64] M. Minervini, A. Fischbach, H. Scharr, and S.A. Tsaftaris. Plant phenotyping datasets, 2015. URL `http://www.plant-phenotyping.org/datasets`.

[65] Massimo Minervini, Andreas Fischbach, Hanno Scharr, and Sotirios A. Tsaftaris. Finely-grained annotated datasets for image-based plant phenotyping. *Pattern Recognition Letters*, pages –, 2015. ISSN 0167-8655. doi: http://dx.doi.org/10.1016/j.patrec.2015.10.013. URL `http://www.sciencedirect.com/science/article/pii/S0167865515003645`.

[66] Sharada P Mohanty, David P Hughes, and Marcel Salathé. Using deep learning for image-based plant disease detection. *Frontiers in plant science*, 7:1419, 2016.

[67] Saeid Motiian, Marco Piccirilli, Donald A Adjeroh, and Gianfranco Doretto. Unified deep supervised domain adaptation and generalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5715–5725, 2017.

[68] Greg Olmschenk, Hao Tang, and Zhigang Zhu. Crowd counting with minimal data using generative adversarial networks for multiple target regression. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1151–1159. IEEE, 2018.

[69] Greg Olmschenk, Jin Chen, Hao Tang, and Zhigang Zhu. Dense crowd counting convolutional neural networks with minimal data using semi-supervised dual-goal generative adversarial networks. In *CVPR Workshops*, 2019.

[70] Greg Olmschenk, Zhigang Zhu, and Hao Tang. Generalizing semi-supervised generative adversarial networks to regression using feature contrasting. *Computer Vision and Image Understanding*, 186: 1–12, 2019.

[71] Niall O'Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. Deep learning vs. traditional computer vision. In *Science and Information Conference*, pages 128–144. Springer, 2019.

[72] Joseph Paul Cohen, Genevieve Boucher, Craig A Glastonbury, Henry Z Lo, and Yoshua Bengio. Count-ception: Counting by fully convolutional redundant counting. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 18–26, 2017.

[73] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[74] Michael P Pound, Jonathan A Atkinson, Darren M Wells, Tony P Pridmore, and Andrew P French. Deep learning for multi-task plant phenotyping. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 2055–2063, 2017.

[75] Maryam Rahnemoonfar and Clay Sheppard. Deep count: fruit counting based on deep simulated learning. *Sensors*, 17(4):905, 2017.

[76] Viresh Ranjan, Hieu Le, and Minh Hoai. Iterative crowd counting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 270–285, 2018.

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

[78] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[79] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[80] Pouria Sadeghi-Tehran, Nicolas Virlet, Eva M Ampe, Piet Reyns, and Malcolm John Hawkesford. Deepcount: in-field automatic quantification of wheat spikes using simple linear iterative clustering and deep convolutional neural networks. *Frontiers in plant science*, 10:1176, 2019.

[81] Faraz Saeedan, Nicolas Weber, Michael Goesele, and Stefan Roth. Detail-preserving pooling in deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9108–9116, 2018.

[82] Zan Shen, Yi Xu, Bingbing Ni, Minsi Wang, Jianguo Hu, and Xiaokang Yang. Crowd counting via adversarial cross-scale consistency pursuit. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5245–5254, 2018.

[83] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[84] Vishwanath A Sindagi and Vishal M Patel. A survey of recent advances in cnn-based single image crowd counting and density estimation. *Pattern Recognition Letters*, 107:3–16, 2018.

[85] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[86] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[87] Hilal Tayara, Kim Gil Soo, and Kil To Chong. Vehicle detection and counting in high-resolution aerial images using convolutional regression neural network. *IEEE Access*, 6:2220–2230, 2017.

[88] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4068–4076, 2015.

[89] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017.

[90] Jordan Ubbens, Mikolaj Cieslak, Przemyslaw Prusinkiewicz, and Ian Stavness. The use of plant models in deep learning: an application to leaf counting in rosette plants. *Plant methods*, 14(1):6, 2018.

[91] Jordan R Ubbens and Ian Stavness. Deep plant phenomics: a deep learning platform for complex plant phenotyping tasks. *Frontiers in plant science*, 8:1190, 2017.

[92] Hideaki Uchiyama, Shunsuke Sakurai, Masashi Mishima, Daisaku Arita, Takashi Okayasu, Atsushi Shimada, and Rin-ichiro Taniguchi. An easy-to-setup 3d phenotyping platform for komatsuna dataset. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 2038–2045, 2017.

[93] Mario Valerio Giuffrida, Hanno Scharr, and Sotirios A Tsaftaris. Arigan: Synthetic arabidopsis plants using generative adversarial network. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 2064–2071, 2017.

[94] Mario Valerio Giuffrida, Andrei Dobrescu, Peter Doerner, and Sotirios A Tsaftaris. Leaf counting without annotations using adversarial unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[95] Varun Kannadi Valloli and Kinal Mehta. W-net: Reinforced u-net for density map estimation. *arXiv preprint arXiv:1903.11249*, 2019.

[96] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[97] Elad Walach and Lior Wolf. Learning to count with cnn boosting. In *European conference on computer vision*, pages 660–676. Springer, 2016.

[98] Chuan Wang, Hua Zhang, Liang Yang, Si Liu, and Xiaochun Cao. Deep people counting in extremely dense crowds. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1299–1302, 2015.

[99] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312: 135–153, 2018.

[100] Qi Wang, Junyu Gao, Wei Lin, and Yuan Yuan. Learning from synthetic data for crowd counting in the wild. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8198–8207, 2019.

[101] Qi Wang, Tao Han, Junyu Gao, and Yuan Yuan. Neuron linear transformation: Modeling the domain shift for crowd counting. *arXiv preprint arXiv:2004.02133*, 2020.

[102] Daniel Ward, Peyman Moghadam, and Nicolas Hudson. Deep leaf segmentation using synthetic data. *arXiv preprint arXiv:1807.10931*, 2018.

[103] Garrett Wilson and Diane J Cook. A survey of unsupervised deep domain adaptation. *arXiv preprint arXiv:1812.02849*, 2018.

[104] Xinyu Wu, Guoyuan Liang, Ka Keung Lee, and Yangsheng Xu. Crowd density estimation using texture analysis and learning. In *2006 IEEE international conference on robotics and biomimetics*, pages 214–219. IEEE, 2006.

[105] Weidi Xie, J Alison Noble, and Andrew Zisserman. Microscopy cell counting and detection with fully convolutional regression networks. *Computer methods in biomechanics and biomedical engineering: Imaging & Visualization*, 6(3):283–292, 2018.

[106] Haipeng Xiong, Zhiguo Cao, Hao Lu, Simon Madec, Liang Liu, and Chunhua Shen. Tasselnetv2: in-field counting of wheat spikes with context-augmented local regression networks. *Plant methods*, 15(1): 150, 2019.

[107] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[108] Chenfeng Xu, Kai Qiu, Jianlong Fu, Song Bai, Yongchao Xu, and Xiang Bai. Learn to scale: Generating multipolar normalized density maps for crowd counting. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8382–8390, 2019.

[109] Ting Yao, Yingwei Pan, Chong-Wah Ngo, Houqiang Li, and Tao Mei. Semi-supervised domain adaptation with subspace learning for visual recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2142–2150, 2015.

[110] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[111] Qian Yu, Feng Liu, Yi-Zhe Song, Tao Xiang, Timothy M Hospedales, and Chen-Change Loy. Sketch me that shoe. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 799–807, 2016.

[112] Aston Zhang, Zachary C Lipton, Mu Li, and Alexander J Smola. Dive into deep learning. 2019.

[113] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[114] Dexuan Zhang and Tatsuya Harada. A general upper bound for unsupervised domain adaptation. *arXiv preprint arXiv:1910.01409*, 2019.

[115] Shanghang Zhang, Guanhang Wu, Joao P Costeira, and José MF Moura. Fcn-rlstm: Deep spatio-temporal neural networks for vehicle counting in city cameras. In *Proceedings of the IEEE international conference on computer vision*, pages 3667–3676, 2017.

[116] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 589–597, 2016.

[117] Ji Zhou, Daniel Reynolds, Danny Websdale, Thomas Le Cornu, Oscar Gonzalez-Navarro, Clare Lister, Simon Orford, Stephen Laycock, Graham Finlayson, Tim Stitt, et al. Cropquant: An automated and scalable field phenotyping platform for crop monitoring and trait measurements to facilitate breeding and digital agriculture. *BioRxiv*, page 161547, 2017.

[118] Naihui Zhou, Zachary D Siegel, Scott Zarecor, Nigel Lee, Darwin A Campbell, Carson M Andorf, Dan Nettleton, Carolyn J Lawrence-Dill, Baskar Ganapathysubramanian, Jonathan W Kelly, et al. Crowdsourcing image analysis for plant phenomics to generate ground truth data for machine learning. *PLoS computational biology*, 14(7):e1006337, 2018.

[119] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[120] Runkai Zhu, Dong Sui, Hong Qin, and Aimin Hao. An extended type cell detection and counting method based on fcn. In *2017 IEEE 17th International Conference on Bioinformatics and Bioengineering (BIBE)*, pages 51–56. IEEE, 2017.