IMAGE ANALYSIS FOR PLANT PHENOTYPING

by

Enyu Cai

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

School of Electrical and Computer Engineering

West Lafayette, Indiana

August 2023

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Edward J. Delp, Chair

School of Electrical and Computer Engineering

Dr. Fengqing M. Zhu

School of Electrical and Computer Engineering

Dr. Mary L. Comer

School of Electrical and Computer Engineering

Dr. Melba M. Crawford

Department of Agronomy

Approved by:

Dr. Dimitrios Peroulis

ACKNOWLEDGMENTS

First of all, I would like to thank my advisor Professor Edward J. Delp for his grateful guidance and support. He gave me an opportunity to be a member of the Video and Image Processing Laboratory (VIPER). His feedback and criticism help me to take my work to a higher level. His tremendous passion for research and leadership in research project management always inspire and motivate me.

I would like to thank Professor Melba M. Crawford for her precious advice, innovative research ideas, and invaluable suggestion on documentation.

I would like to thank Professor Fengqing M. Zhu and Professor Mary Comer for their professional suggestion and efforts in classes.

I would like to thank my colleagues Sriram Baireddy, Changye Yang, Jiaqi Guo, and Dr. Yuhao Chen for their friendship and support. I would also like to thank former or current VIPER lab colleagues Dr. Liming Wu, Dr. Hanxiang Hao, Dr. Emily Bartusiak, Dr. Javier Ribera, Dr. David Güera, Yue Han, Alain Chen, Dr. Qingshuang Chen, Zhihao Duan, Dr. Shaobo Fang, Dr. Mridul Gupta, Dr. Shuo Han, Dr. Jiangpeng He, Dr. János Horváth, Dr. David Joon Ho, Xiaoyu Ji, Zhankun Luo, Dr. Soonam Lee, Dr. Runyu Mao, Ruiting Shao, Zeman Shao, Yezhi Shen, Gautham Vinod, Ziyue Xiang, Kratika Bhagtani, Ryan Schwarz, Amit Kumar Singh Yadav, Weichen Xu, Justin Yang, Dr. Dahjung Chung, Dr. Daniel Mas Montserrat, Dr. Di Chen, and Dr. Sri Kalyan Yarlagadda.

I would like to thank my parents for their love and support.

I would like to thank Professor Ayman F. Habib and the members of the Digital Photogrammetry Research Group (DPRG) for providing data in this thesis.

The information, data, or work presented herein was funded in part by the Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy, under Award Number DE-AR0001135. The views and opinions of the authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

AP	Average Precision
CNN	Convolutional Neural Network
EM	Expectation Maximization
FP	False Positive
FPN	Feature Pyramid Network
GAN	Generative Adversarial Network
HSV	Hue, Saturation, and Value
IoU	Intersection over Union
MAE	Mean Absolute Error
MAHD	Mean Average Hausdorff Distance
MAPE	Mean Absolute Percent Error
MLP	Multilayer Perceptron
QTL	Quantitative Trait Loci
ReLU	Rectified Linear Unit
RGB	Red, Green, and Blue
RMSE	Root Mean Squared Error
ROI	Region of Interest
RPN	Regional Proposal Network
TP	True Positive
UAV	Unmanned Aerial Vehicle
WHD	Weighted Hausdorff Distance

ABSTRACT

Plant phenotyping focuses on the measurement of plant characteristics throughout the growing season, typically with the goal of evaluating genotypes for plant breeding and management practices related to nutrient applications. Estimating plant characteristics is important for finding the relationship between the plant's genetic data and observable traits, which is also related to the environment and management practices. Recent machine learning approaches provide promising capabilities for high-throughput plant phenotyping using images. In this thesis, we focus on estimating plant traits for a field-based crop using images captured by Unmanned Aerial Vehicles (UAVs). We propose a method for estimating plant centers by transferring an existing model to a new scenario using limited ground truth data. We describe the use of transfer learning using a model fine-tuned for a single field or a single type of plant on a varied set of similar crops and fields. We introduce a method for rapidly counting panicles using images acquired by UAVs. We evaluate three different deep neural network structures for panicle counting and location. We propose a method for sorghum flowering time estimation using multi-temporal panicle counting. We present an approach that uses synthetic training images from generative adversarial networks for data augmentation to enhance the performance of sorghum panicle detection and counting. We reduce the amount of training data for sorghum panicle detection via semi-supervised learning. We create synthetic sorghum and maize images using diffusion models. We propose a method for tomato plant segmentation by color correction and color space conversion. We also introduce the methods for detecting and classifying bacterial tomato wilting from images.

1. INTRODUCTION

1.1 High-Throughput Phenotyping

One of the most important concepts in plant science is the genotype-phenotype relationship, which is the link between the genetic information of plants and the observable characteristics of plants [1]–[6]. The term "phenotype" was coined by Danish plant scientist Wilhelm Johannsen for his genotype theory back in 1911 [7], [8].

Plant phenotyping is the methodology to help understand the genotype-phenotype connection by measuring structural and chemical traits such as height, shape, weight, and other properties [9], [10] as shown in Figure 1.1. Plant breeders can use phenotyping to help make decisions on breeding and crop management by evaluating various properties of a crop during the growing season.

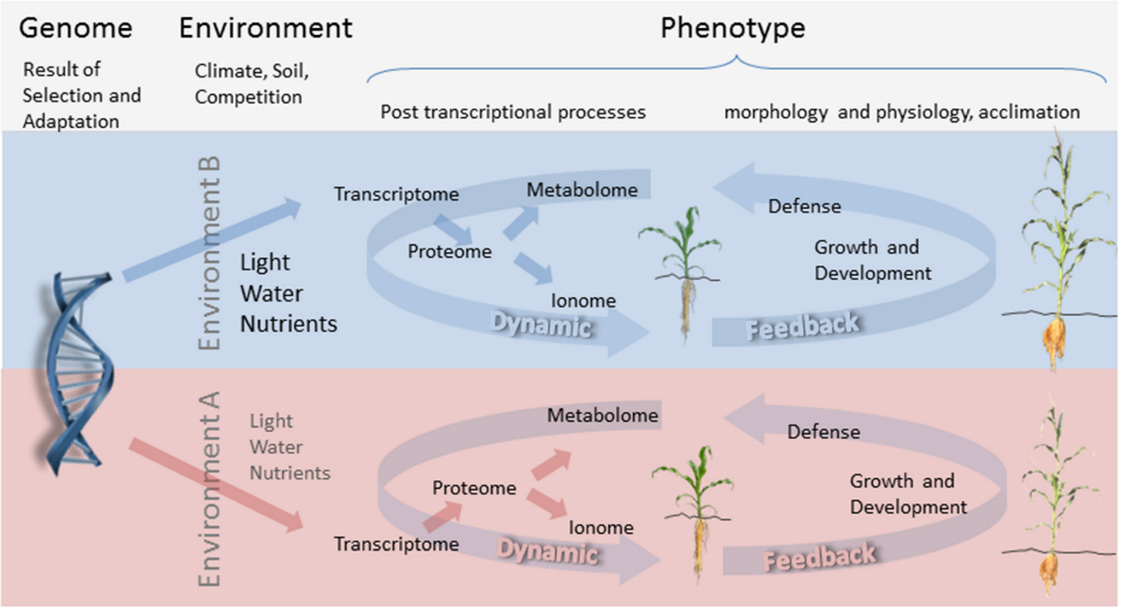


Figure 1.1. Genotype-phenotype relationship from [9].

Traditional plant phenotyping is costly, labor-intensive, and primarily destructive [11]–[13]. Traditional phenotyping involves manually collecting data, conducted by personnel walking through the field, which is not viable for large areas and typically does not provide enough information at the plot level. Modern high-throughput phenotyping [14]–[20] addresses the problems of traditional phenotyping by using remotely sensed data to measure plant properties with robotic platforms. As shown in Figure 1.2, Unmanned Aerial Vehicles (UAVs) with sensors such as RGB and multi/hyperspectral imaging, as well as LiDAR, have demonstrated the capability to reduce and, in some cases, eliminate field-based phenotyping [15], [21]–[23]. Figure 1.3 shows an example of the data collection process of a UAV platform.

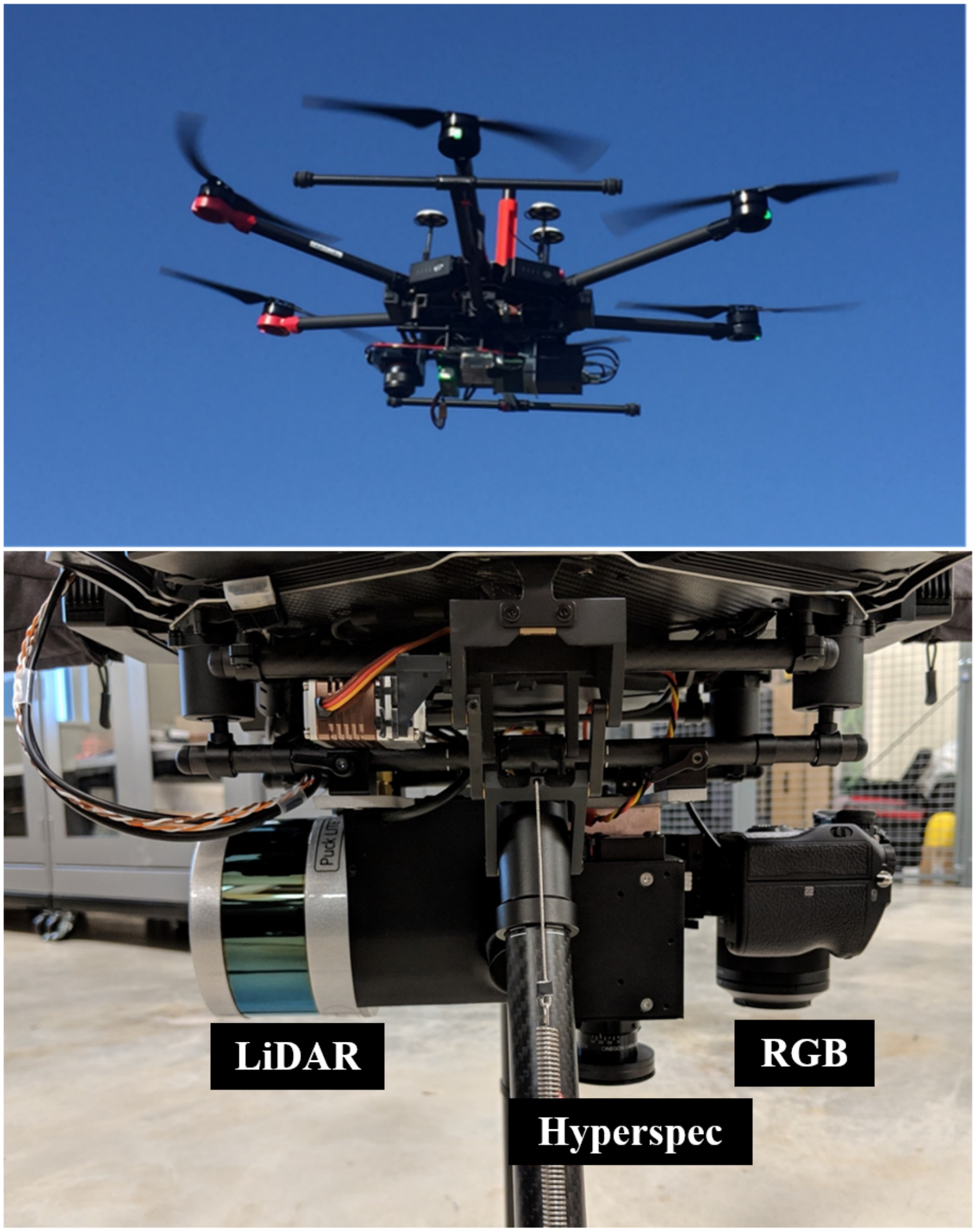


Figure 1.2. A UAV platform equipped with multiple sensors

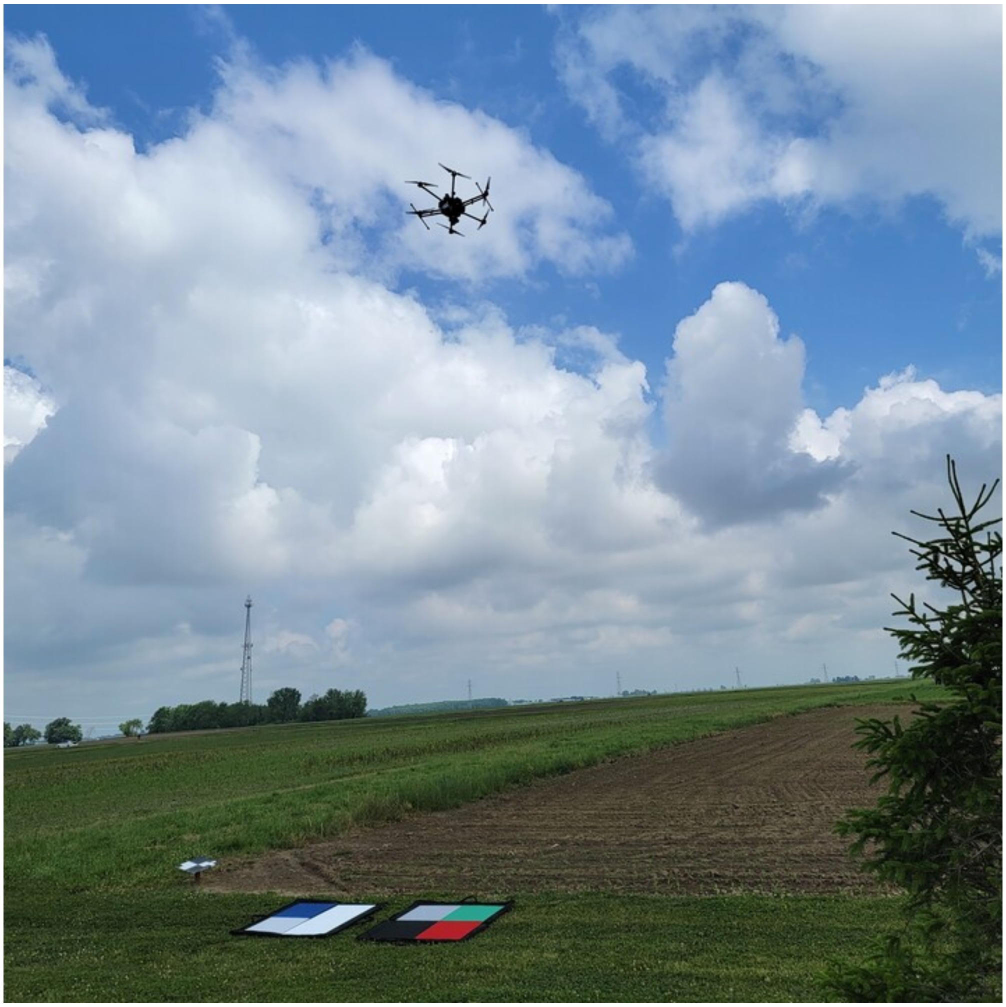


Figure 1.3. A UAV platform collecting data in the field.

UAVs are suitable for high-throughput phenotyping because of their ability to non-invasively collect data from a field in a short time. Compared to traditional phenotyping, using UAVs to collect data has a lower cost and can cover more area in the same period of time. As shown in Figure 1.4, the field aerial images acquired with UAVs need to be geometrically rectified and mosaiced [24] with accurate location properties, which is critical for developing reliable methods for plant location at the field scale. An example of a basic field layout is in Figure 1.5. Inside the field, the experiment is planted in multiple plots according to a statistical experimental design (e.g. randomized block). Fields typically included multiple experiments, but the same crop. One reason for this is that fields are assigned to researchers that are specified to be wheat or soybean or maize researchers. The plant researchers do not conduct plant experiments in each other's fields. Within each plot, crops are planted using research planters that drop seeds at a given rate in rows of a given length. Inside the plot, there are multiple row segments of plants. Figure 1.6 provides an example of the panel layout.

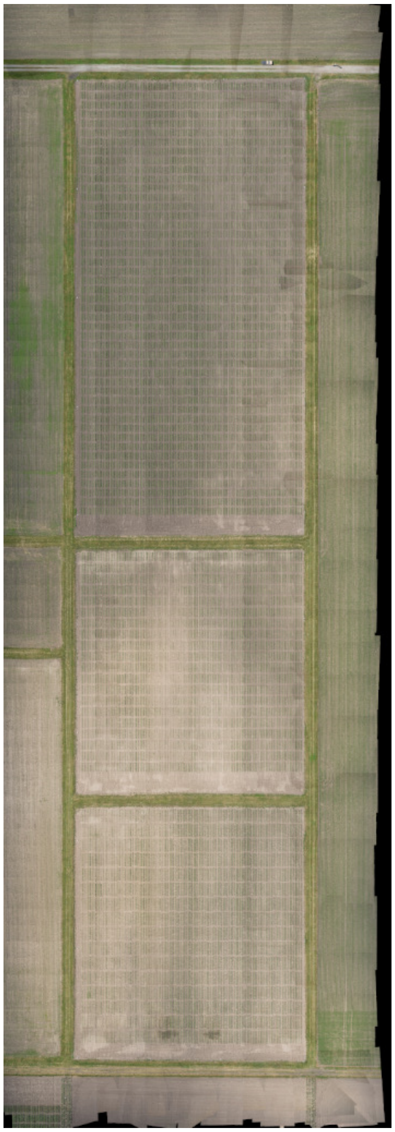


Figure 1.4. An orthorectified maize field RGB image [24] from June 4, 2018, at an altitude of 50 meters and resolution of 1 cm/pixel.

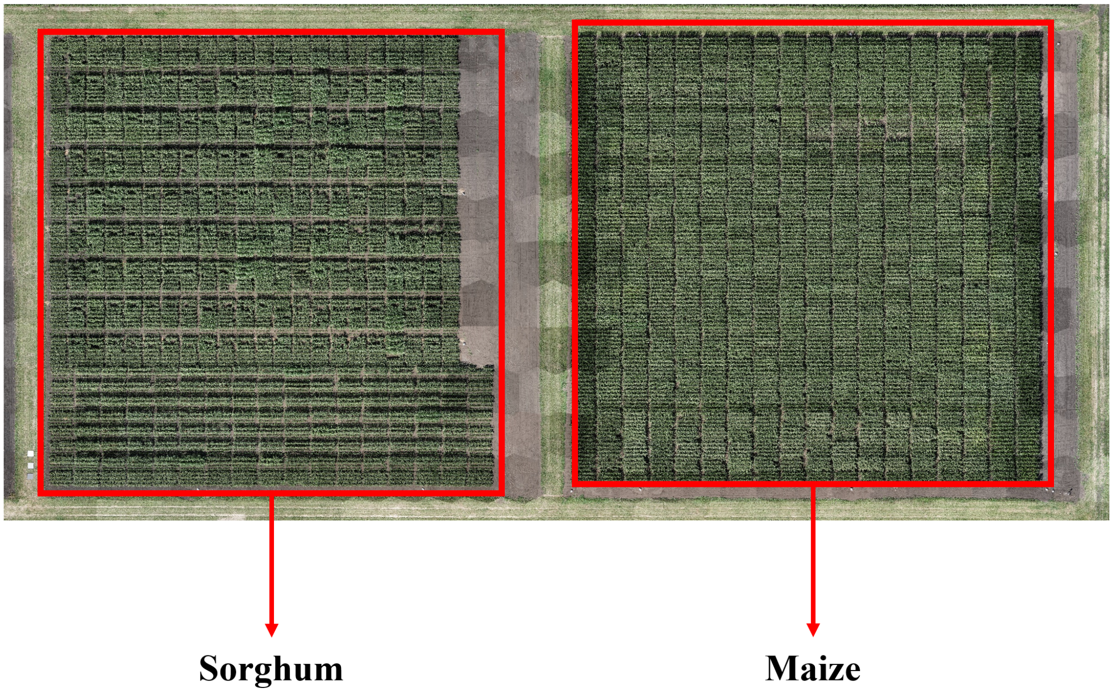


Figure 1.5. An example of a sorghum field and a maize field.

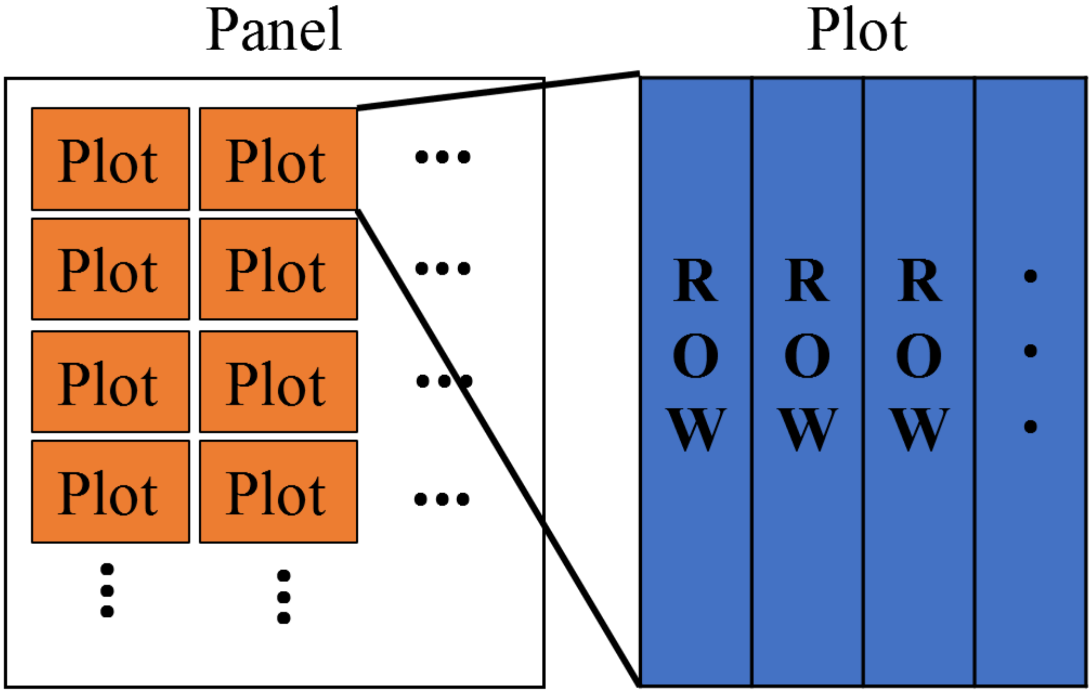


Figure 1.6. An example of the panel (experiment) layout.

1.2 Sorghum Growth

Crop-based biofuels, an environmentally sustainable energy resource derived from plant matter, can help reduce greenhouse gas emissions and dependency on exhaustible or foreign resources [25]–[31]. In the past, sorghum (*Sorghum bicolor* (L.) Moench) was used in forage, grain, and food due to its ability to resist harsh conditions [32]–[39]. The high level of sugar contents in the sorghum stalk and the low input resources compared to other input-intensive food crops make it become an attractive plant for biofuel production [26], [40]–[45]. Biofuel from sorghum is less controversial than biofuel from main food crops such as corn and sugar cane since the production of biofuel from these food crops could lead to supply shortages in food production [42], [46]–[49]. An example of sorghum structure is shown in Figure 1.7. A sorghum plant has 10 official growing stages as shown in Figure 1.8. Among those 10 stages, there are three important stages of sorghum [50], [51]. In the first stage, the plant develops its leaves and stems. In the second stage, the plant starts to form its panicles which is the cluster of the grain on a branch. In the third stage, the panicle starts to flower and the grains become mature. Figure 1.9 shows an example of a sorghum field in the third stage.

Flowering time (time to flower after planting) is an important phenotypic trait related to plant development and grain yield in sorghum [53]–[59]. A sorghum plant is considered ”flowering” when the 50% of grains on a panicle are flowering (or blooming), and a plot (a section of the crop field) is flowering when 50% of the sorghum plants have reached this stage [50]. Figure 1.10 shows an example of 50% flowering sorghum panicle.

1.3 Bacterial Wilt of Tomatoes

Bacterial wilt by the soil-borne bacterium (*Ralstonia Solanacearum*) is a major constraint to tomato production, causing over 90 percent disease loss during epidemics [60]–[65]. The bacteria infect tomatoes through roots and cause plant leaves to wilt for a few days. Figure 1.11 shows an example of an infected tomato plant. It starts to wilt on the second day after treating with bacteria.

The resistance to bacteria in tomato varieties is quantitative and best controlled by resistant varieties but no quantitative trait loci (QTL) for resistance to US strains have

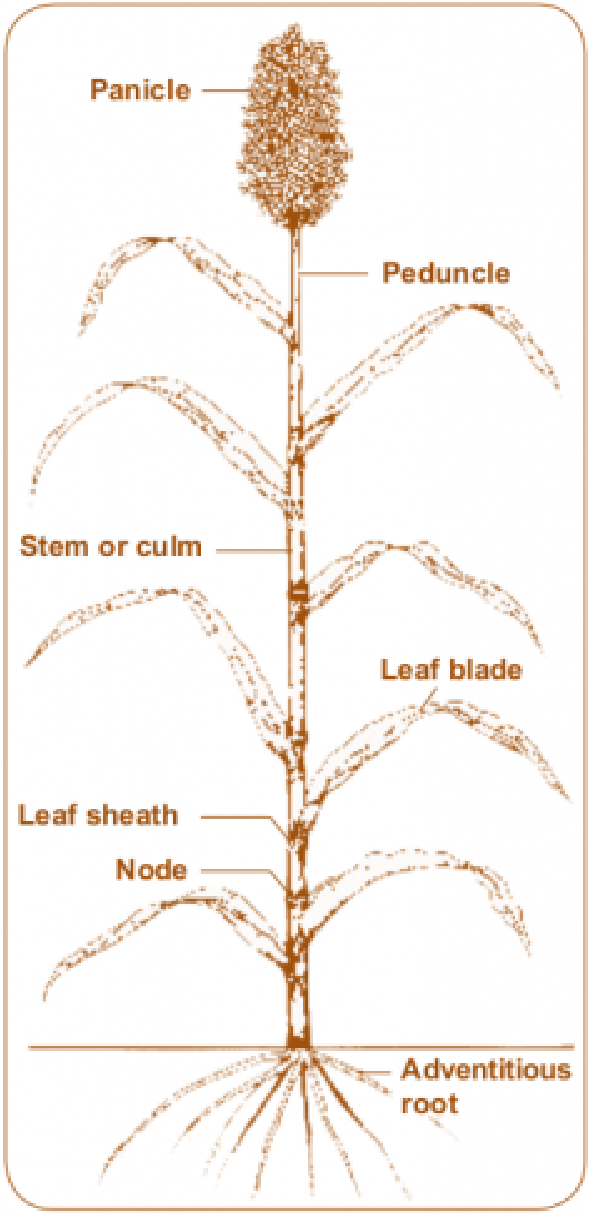


Figure 1.7. Anatomy of sorghum from [52].

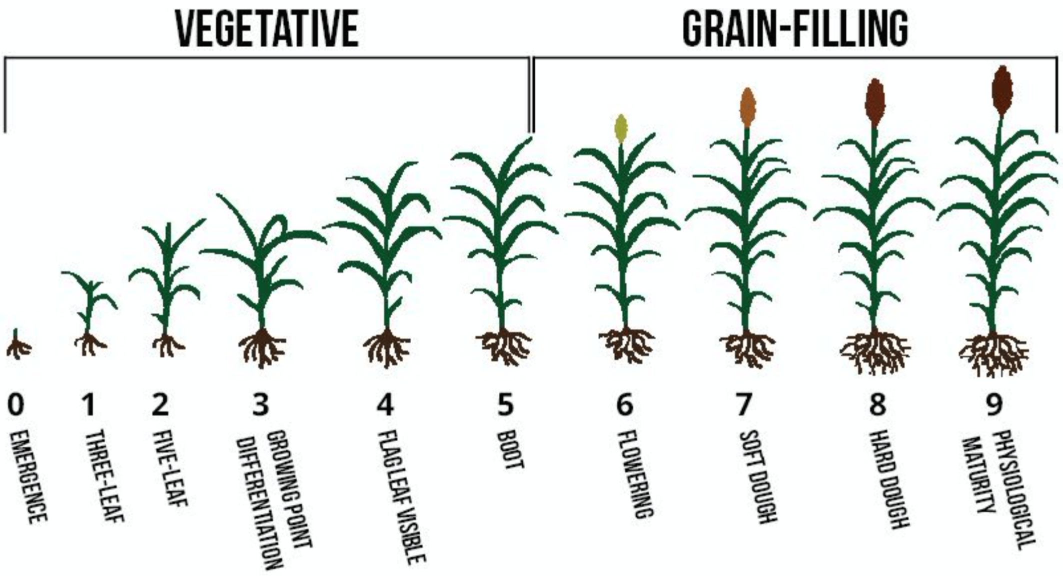


Figure 1.8. Growing stage of sorghum from [50].

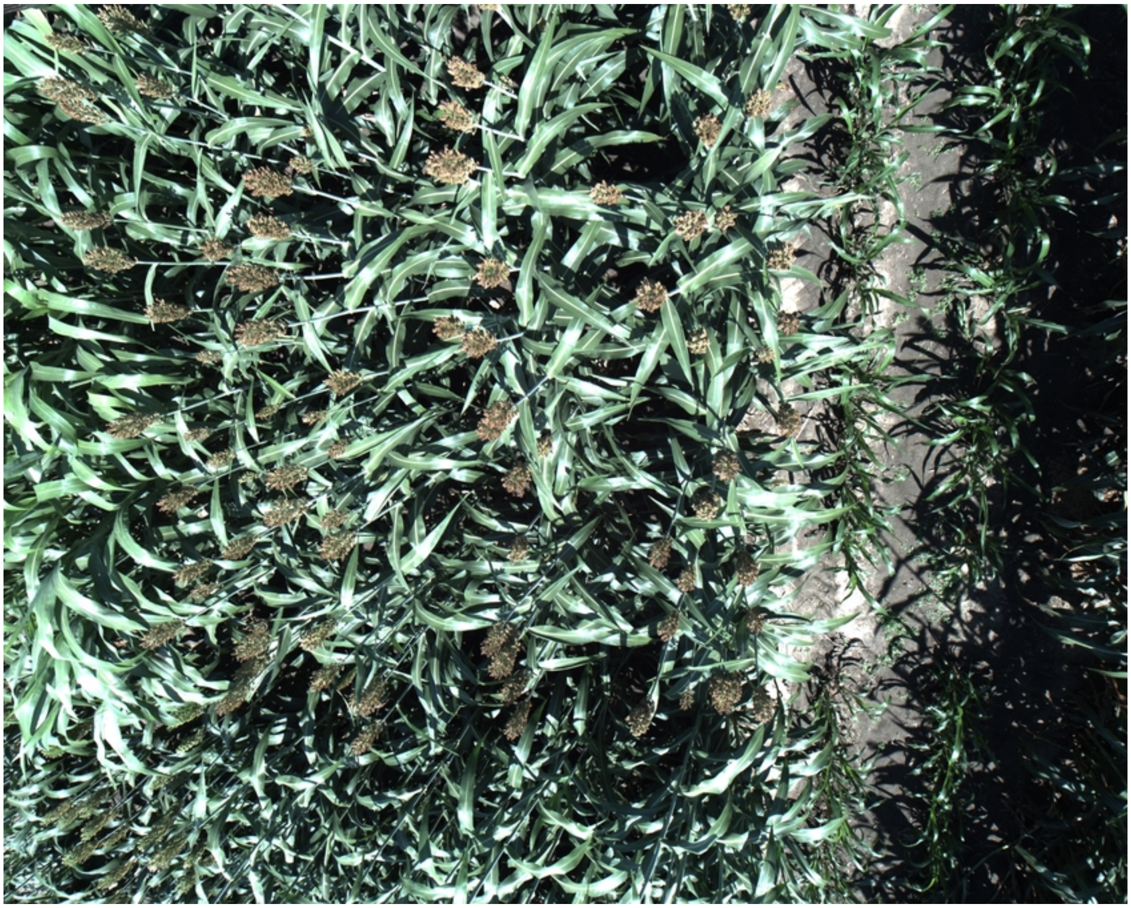


Figure 1.9. A field of sorghum in the V stage.

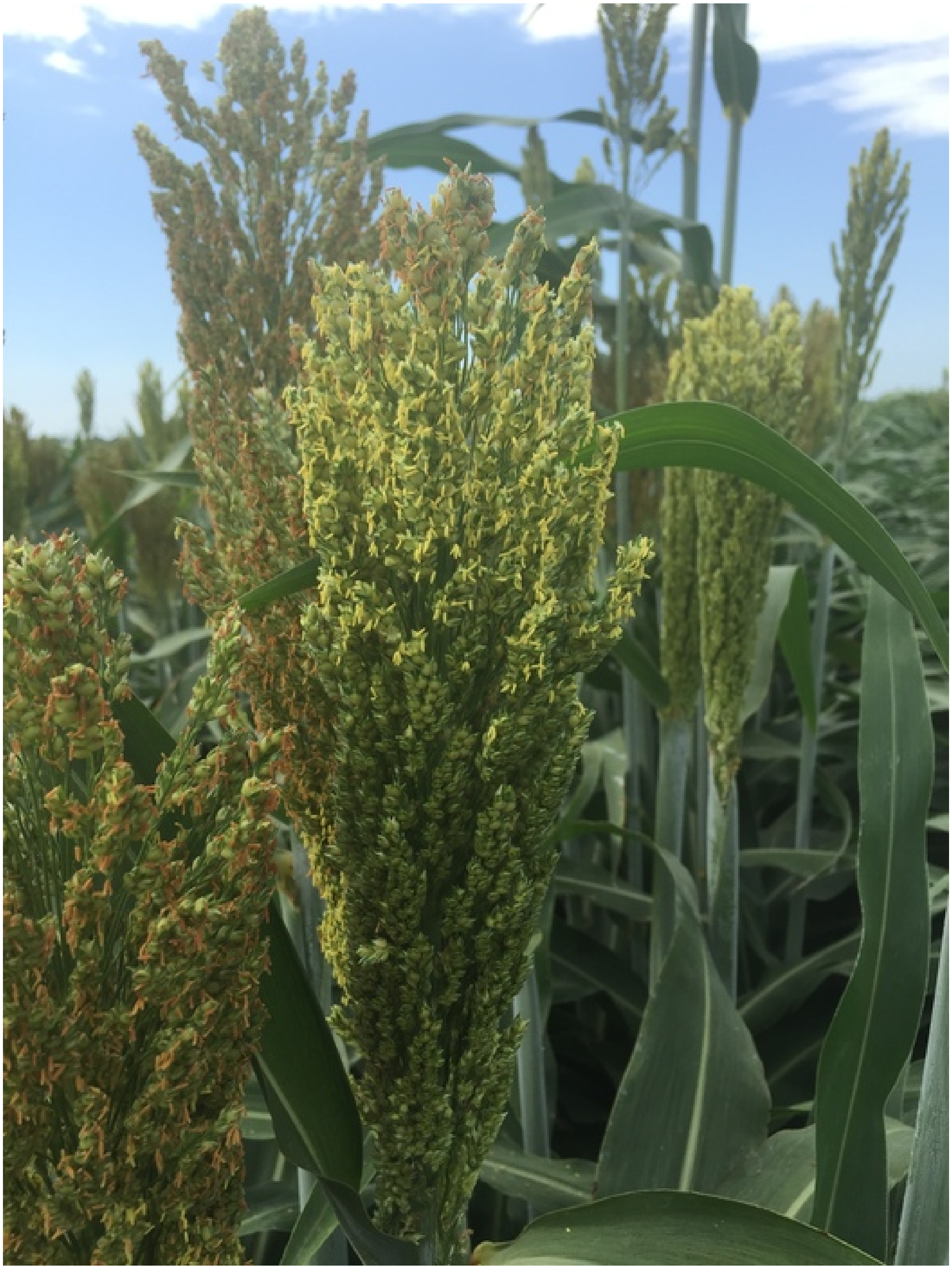
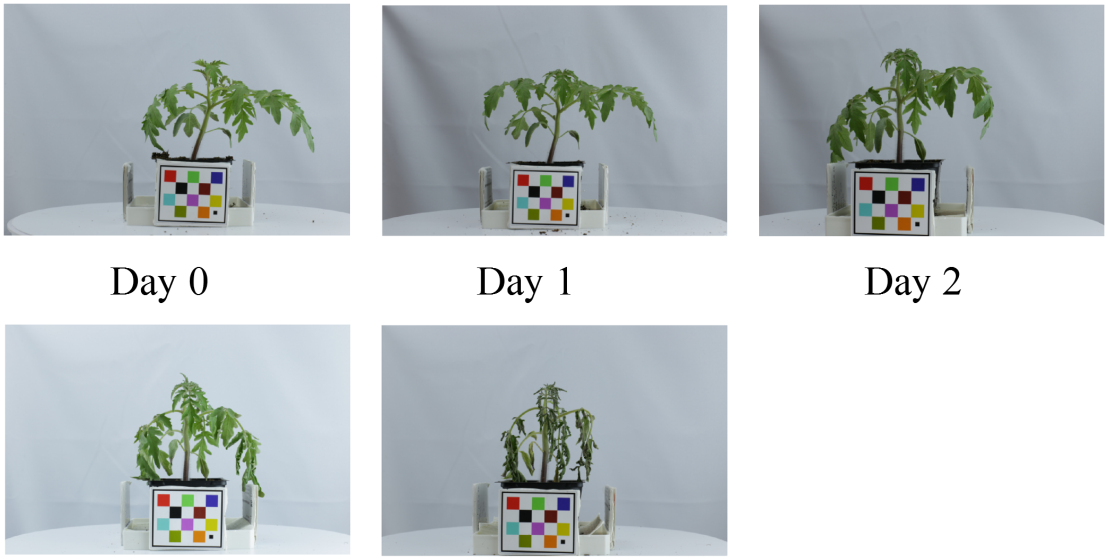


Figure 1.10. A panicle at 50% flowering stage [50].



Day 0

Day 1

Day 2

Day 3

Day 5

Figure 1.11. An infected tomato plant. The 'day' means days after treatment with bacteria.

been identified. This is in part due to the difficulty in identifying disease traits. Developing phenotyping methods for wilting will enhance the ability to identify QTL for resistance to bacteria [11]. In this thesis, We are addressing the methods for detecting and classifying above-ground tomato wilting from RGB images.

1.4 Contribution of This Thesis

In this thesis, we develop methods for plant center localization, sorghum flowering time estimation, and tomato wilting classification. The main contributions of this work are:

- We develop an approach for plant center locations using transfer learning.
- We propose a method for sorghum panicle detection using Convolutional Neural Network (CNN).
- We propose a method for sorghum flowering time estimation using the counts of multi-temporal sorghum panicles.
- We develop a method to improve the performance of sorghum panicle detection via data augmentation.
- We demonstrate an approach to reduce the amount of training data needed for training panicle detection networks using semi-supervised learning.
- We investigate generating synthetic crop images using diffusion models.
- We develop a method for tomato plant segmentation.
- We develop an approach for tomato wilting classification.

1.5 Publications Resulting From This Thesis

1. **E. Cai**, J. Guo, C. Yang, and E. J. Delp, "Semi-Supervised Object Detection for Sorghum Panicles in UAV Imagery", *Proceedings of the IEEE International Symposium on Geoscience and Remote Sensing*, July 2023, Pasadena, CA.

2. **E. Cai**, Z. Luo, S. Baireddy, J. Guo, C. Yang, and E. J. Delp, "High-Resolution UAV Image Generation for Sorghum Panicle Detection", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), the 3rd International Workshop and Prize Challenge on Agriculture-Vision: Challenges & Opportunities for Computer Vision in Agriculture*, June 2022, New Orleans, LA.

3. **E. Cai**, S. Baireddy, C. Yang, M. Crawford, and E. J. Delp, "Panicle Counting in UAV Images For Estimating Flowering Time in Sorghum", *Proceedings of the IEEE International Symposium on Geoscience and Remote Sensing*, July 2021, Brussels, Belgium.

4. **E. Cai**, S. Baireddy, C. Yang, M. Crawford, and E. J. Delp, "Deep Transfer Learning For Plant Center Localization", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, the 1st International Workshop and Prize Challenge on Agriculture-Vision: Challenges & Opportunities for Computer Vision in Agriculture*, June 2020, Seattle, WA.

5. C. Yang, S. Baireddy, Y. Chen, **E. Cai**, D. Caldwell, V. Méline, A. S. Iyer-Pascuzzi, E. J. Delp, "Plant Stem Segmentation Using Fast Ground Truth Generation", *Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpretation*, March 2020, Santa Fe, NM.

1.6 Publications Not Resulting From This Thesis

1. J. Guo, C. Yang, **E. Cai**, and E. J. Delp, "Rotation Adaptive Plot Extraction from UAV RGB Images", *Proceedings of the IEEE International Symposium on Geoscience and Remote Sensing*, July 2023, Pasadena, CA.

2. **E. Cai**, R. Rossi, and C. Xiao, "Improving Learning-based Camera Pose Estimation for Image-based Augmented Reality Applications", *In Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems (CHI EA '23)*, April 2023, Hamburg, Germany.

3. C. Yang, S. Baireddy, **E. Cai**, V. Meline, D. Caldwell, A. S. Iyer-Pascuzzi and E. J. Delp, "Image-Based Plant Wilting Estimation", *arXiv preprint arXiv:2105.12926*, 2021.

4. C. Yang, S. Baireddy, **E. Cai**, M. Crawford, and E. J. Delp, "Field-Based Plot Extraction Using UAV RGB Images", *Proceedings of the IEEE International Conference on Computer Vision(ICCV), Workshop on Computer Vision in Plant Phenotyping and Agriculture(CVPPA)*, October 2021, Montreal, Canada.

5. Y. Chen, S. Baireddy, **E. Cai**, C. Yang, and E. J. Delp, "Leaf Segmentation by Functional Modeling", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Workshop on Computer Vision Problems in Plant Phenotyping*, June 2019, Long Beach, CA.

2. PLANT CENTER LOCALIZATION

2.1 Introduction

Locating plant centers from UAV images with deep learning is not a trivial problem. Because of the altitude of most UAV flights, field-scale aerial images have a spatial resolution of 1 cm per pixel or less. The problem is even more difficult when plants are in an early stage of growth and are very small. Flying at a lower altitude increases the spatial resolution, but the data sets are larger and additional flight lines are required to cover the field, even necessitating multiple flights due to limited battery time.

Traditional methods for image-based plant localization are often related to modeling the plants before detection [66]. The widely varying plant features such as plant shapes and plant overlap impact the capability of modeling and detecting using traditional methods. Using deep learning approaches, the system learns the features during training instead of modeling features before training to avoid the problems associated with traditional methods. In recent years, deep learning has been successfully used for the detection of objects from UAV images. In [67], an approach is developed for detecting objects in UAV images by combining a radial basis function neural network with a thresholding operation. In [68], a method is used to detect and count citrus trees by integrating two convolutional neural networks (CNN). In [69], the plant centers are detected from orthorectified images [24] using a deep binary classifier.

Deep learning is highly dependent on the quality and quantity of the available training data. Large amounts of high-quality ground truth data are needed to achieve good performance. Deep learning models usually perform well if training and testing data are from the same type of data (e.g. in our case the same field, same time, and with the same type of plants). If we apply the same model to different data, the results are often degraded. For example, the color of the soil and the plant size can vary across different types of fields and plants. These variables can cause a network fine-tuned on a single field with a single plant type to fail when used on other types of plants. In this case, training a new network to achieve high performance requires the acquisition of ground truth data on a different field with the associated large quantities of training data, creating a major bottleneck. In this chapter, we present a method for estimating plant centers for two crops and dates with a limited quantity of training data using a transfer learning approach.

2.2 Overview Of Related Work

2.2.1 Network-Based Transfer Learning

As noted previously, deep learning methods usually require significantly more training data than traditional machine learning [70] due to the increased number of parameters. The number of parameters of a 16-layer CNN, for example, can easily exceed millions [71]–[73]. Training with insufficient data often results in poor performance. A few thousand images are inadequate to properly train most deep neural networks from scratch. The results reflect the inability of the model to converge with limited data. Collecting more training data (ground truth) is labor-intensive and costly. Transfer learning leverages the issue of lack of training data by transferring the knowledge from the source domain to the target domain [70] as shown in Figure 2.1. There are multiple categories of transfer learning including instance-based [74]–[79], mapping-based [80], [81], network-based [82]–[85] and adversarial-based [86]–[90]. In this chapter, we only discuss network-based since it has the best fit for our application.

As shown in Figure 2.2, network-based transfer learning addresses the problem of insufficient data by transferring a model pre-trained on larger, more general datasets such as ImageNet [92] to the target task [70]. During the transfer learning process, the weights of the pre-trained network are copied to the new network for the target task. In deep neural networks, the first few layers can be considered as a general feature extractor for the input image [93]. For example, in [94], the weights of a pre-trained CNN are transferred to improve the performance of the network with a small amount of training data. In [95], a pre-trained CNN is finetuned for emotion recognition on small datasets. In [96], a pre-trained GoogLeNet [97] is fine-tuned to classify Arabidopsis and Tobacco plants images. In [98], the results show retraining pre-trained networks on plant images can improve the performance compared to training from scratch.

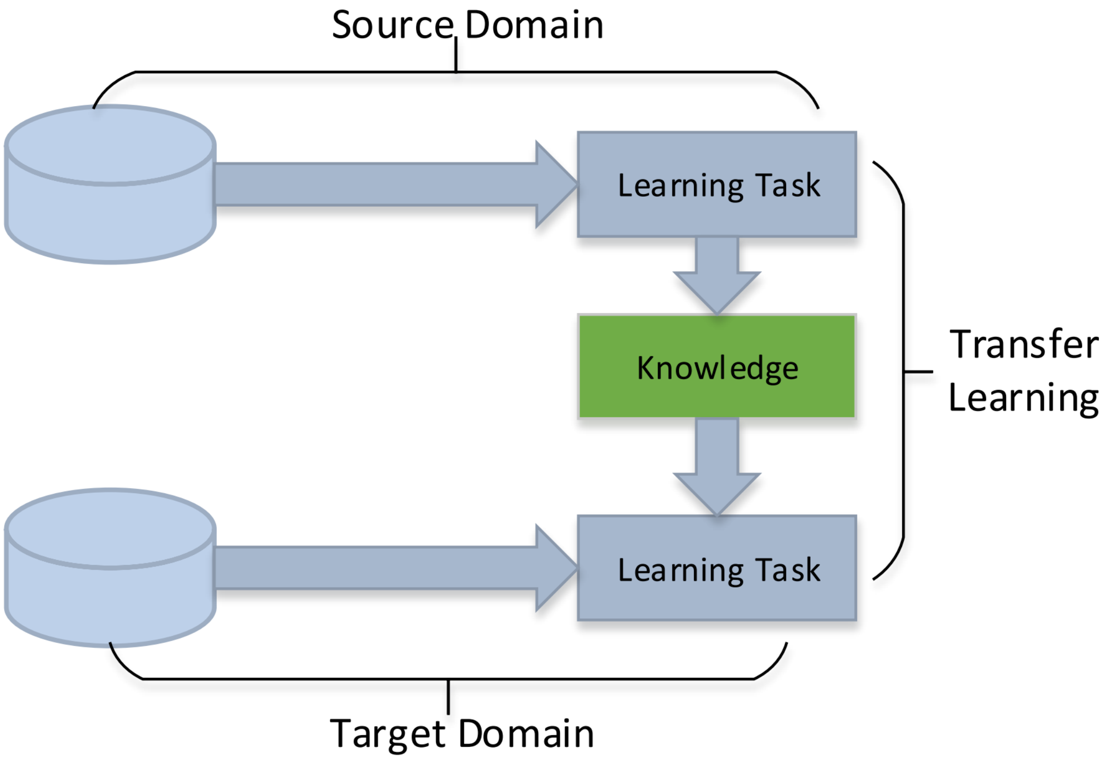


Figure 2.1. Transfer learning process from [70].

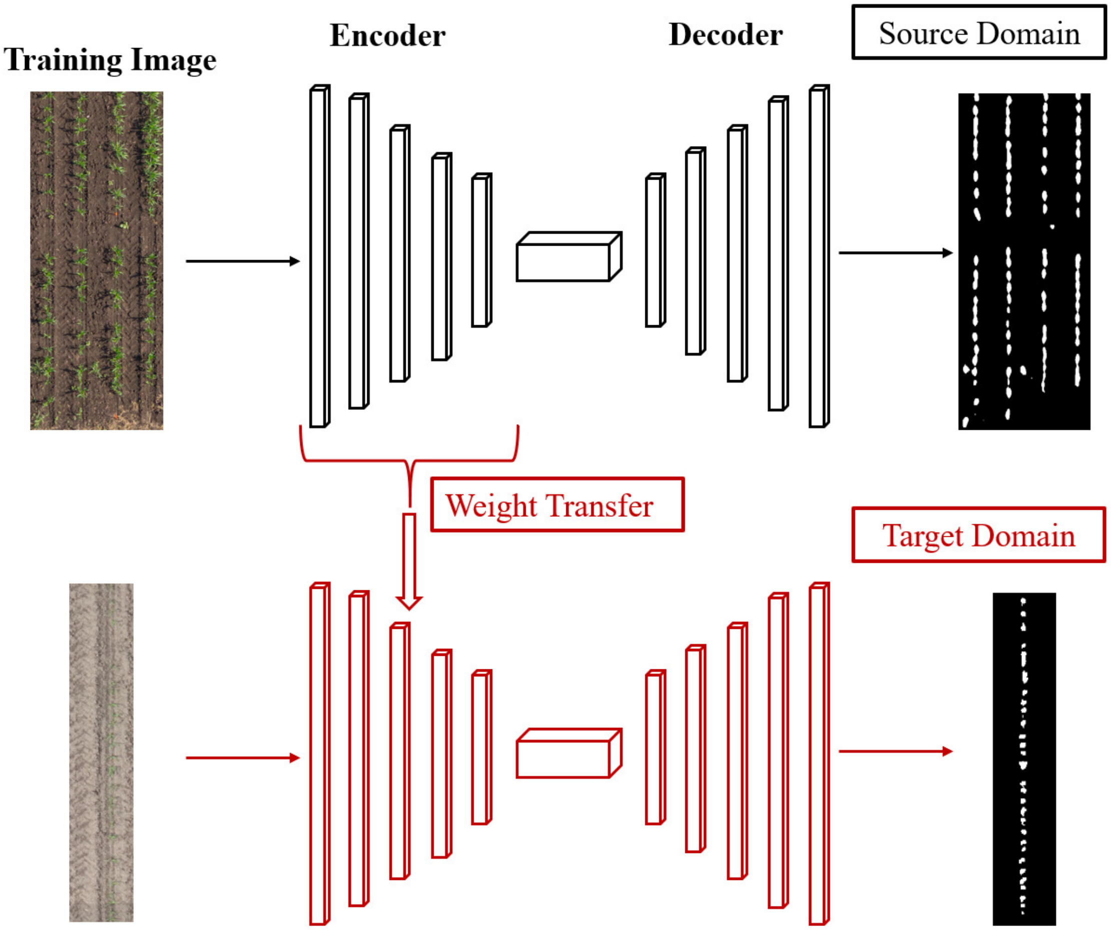


Figure 2.2. Network-based transfer learning. The encoder of U-Net [91] is transferred to the new network for target task training.

2.2.2 Object Detection

Faster R-CNN [99] and Mask R-CNN [100] are object detectors commonly used for general object detection. As shown in Figure 2.3, a region proposal network is used to search for regions of interest in a feature map. The output of the regional proposal network is connected to convolutional layers for object detection and bounding box regression. Based on Faster R-CNN [99], Mask R-CNN adds additional layers to generate segmentation masks for objects in the image. The Mask R-CNN structure is shown in Figure 2.4. The ground truth of these networks is based on bounding boxes or masks. Bounding box-type ground truth often results in inaccurate location estimation when the object is very small. Using bounding boxes to define ground truth is also tedious and time-consuming. Plant centers are small objects, so detecting their location precisely is an important objective for the network. Recent work shows locating and counting objects can be achieved without bounding boxes [101]. In [102], a segmentation map generated from a CNN is developed to count the number of wheat plants. In [103], an estimated map from a CNN is used to estimate the number of rice seedlings from UAV images. In [104], a few-shot learning framework with point annotation is used to locate and count plants.

2.3 Network Architecture and Transfer Learning

Our task can be defined as locating plant centers in orthorectified images [24] with different types of crops, fields, and image acquisition dates. We represent plant centers as points in our ground truth because they are more accurate than bounding boxes in terms of localization, and are relatively easier to use for labeling. Since our task is localization, our ground truth masks are very sparse. We cannot use pixel-wise losses as they do not represent the distance between the prediction and the ground truth unless they perfectly overlap. This is especially true for the task of point localization. Due to this, our approach is based on locating objects without bounding boxes [101], which is used for plant localization, eye pupil

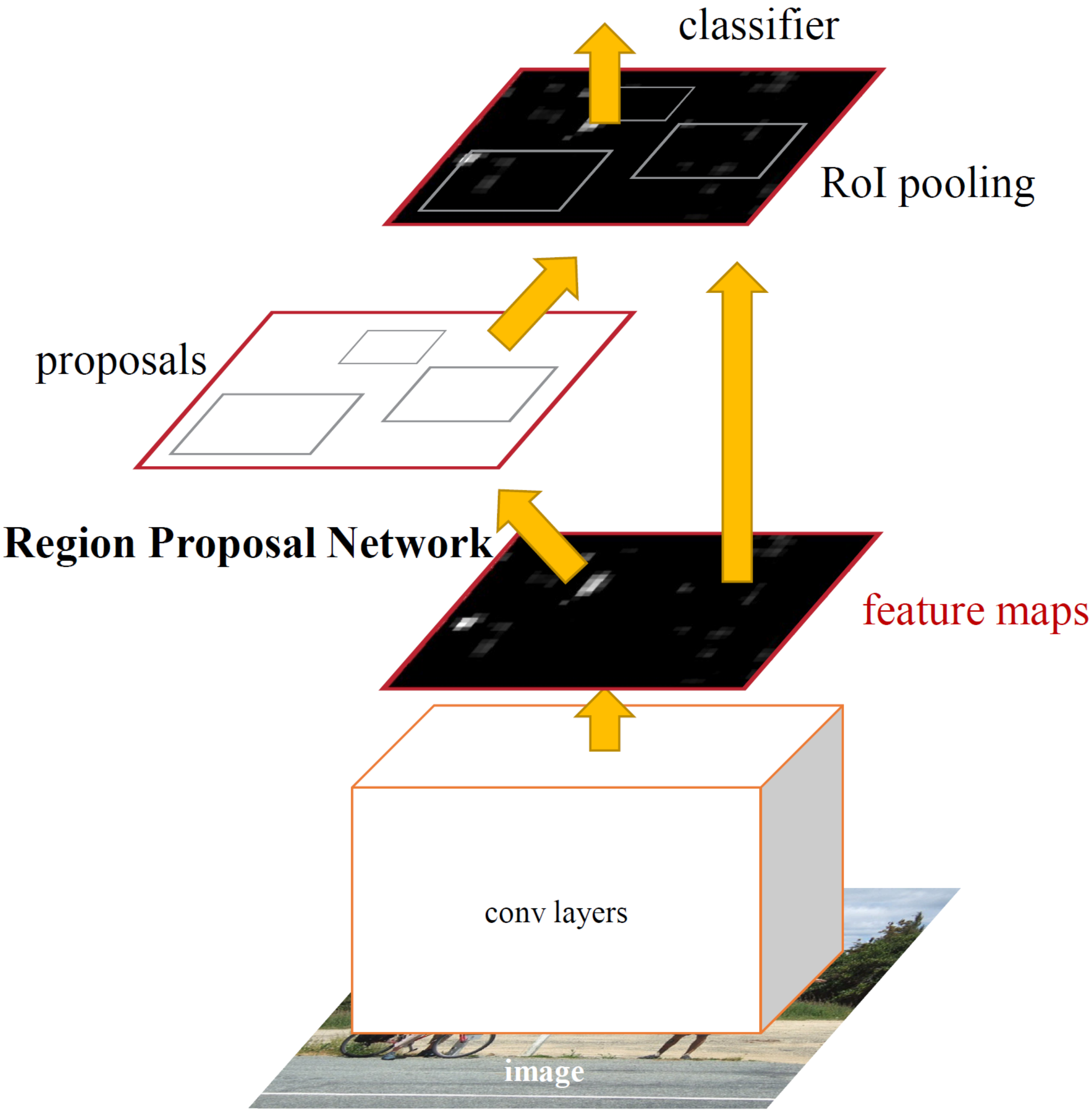


Figure 2.3. Faster R-CNN structure [99].

identification, and people counting. The major contribution in [101], is the proposed loss function: the weighted Hausdorff distance (WHD),

$$d_{\mathrm{WH}}(p, Y) = \frac{1}{\mathcal{S} + \epsilon} \sum_{x \in \Omega} p_x \min_{y \in Y} d(x, y) + \frac{1}{|Y|} \sum_{y \in Y} \underset{x \in \Omega}{M_\alpha} \left[\, p_x d(x, y) + (1 - p_x) d_{max} \,\right], \tag{2.1}$$

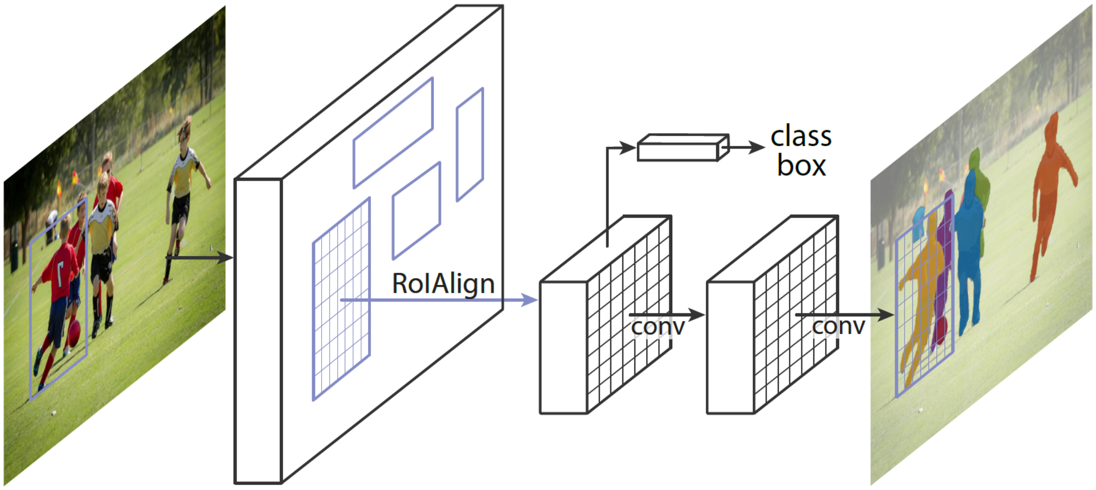


Figure 2.4. Mask R-CNN structure [100].

where

$$\mathcal{S} = \sum_{x\in\Omega} p_x, \tag{2.2}$$

$$\underset{a\in A}{M_\alpha}\left[f(a)\right] = \left(\frac{1}{|A|}\sum_{a\in A} f^\alpha(a)\right)^{\frac{1}{\alpha}}, \tag{2.3}$$

is the generalized mean, $p_x \in [0,1]$ is the output at pixel x and the function $d(\cdot,\cdot)$ is the Euclidean distance. The ϵ in the denominator of the first term is a small positive number that provides stability if the network detects no objects. Multiplying by p_x in the first term ensures that high activations at locations with no ground truth are penalized. The second term has two parts. The expression $f(\cdot) = p_x d(x,y) + (1-p_x)d_{max}$ is used to enforce the constraints $f_{p_x=1} = d(x,y)$ and $f_{p_x=0} = d_{max}$. Now, note that M_α corresponds to the minimum function when $\alpha = -\infty$. So, ideally, if $\alpha = -\infty$, the minimum of the function is obtained, meaning the second constraint $f_{p_x=0} = d_{max}$ will penalize low activations around ground truth points. However, the minimum function makes training difficult as it is not a smooth function w.r.t. its inputs, so in [101], it is approximated with $\alpha < 0$. The best values are empirically found to be $\epsilon = 10^{-6}$ and $\alpha = -1$. One of the strengths of the WHD and the approach of object localization as minimizing the distance between points is that it is independent of the CNN architecture used.

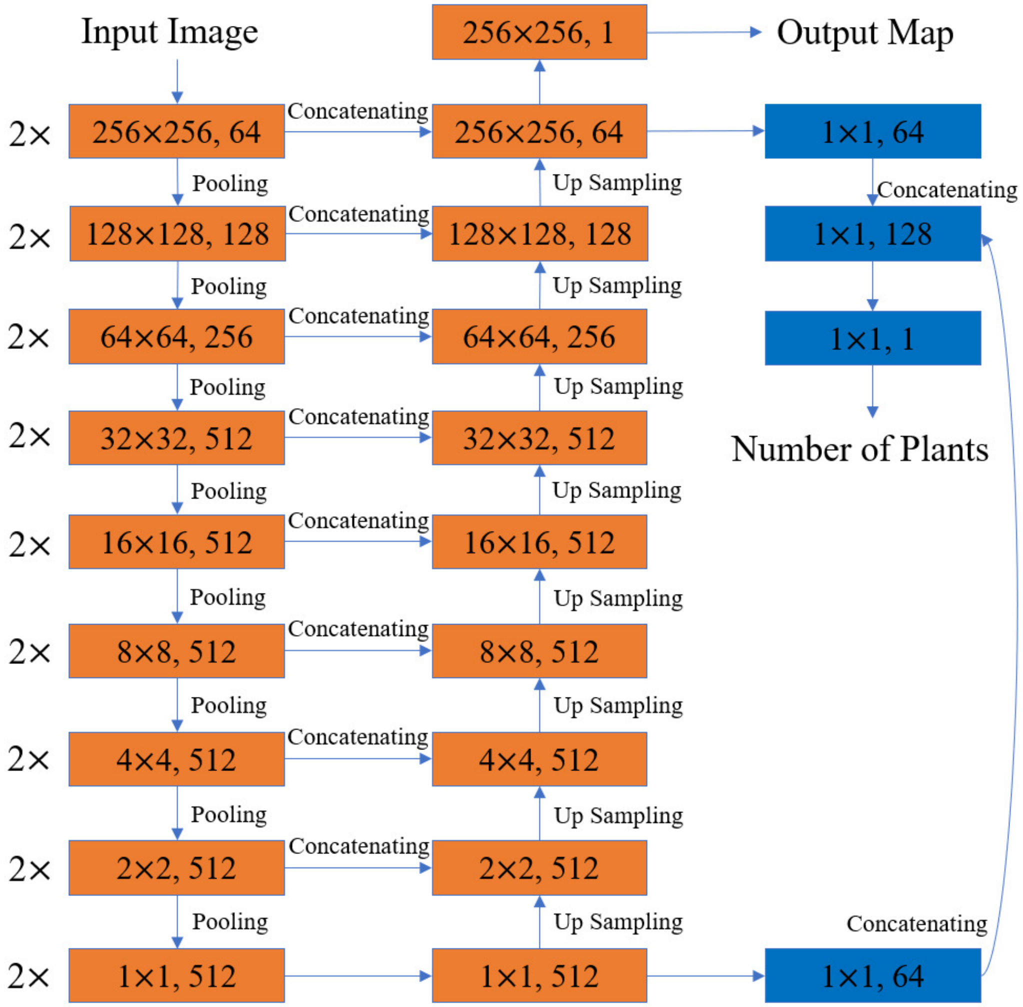


Figure 2.5. Modified U-Net architecture from [101] Each orange block represents a convolutional layer with output shape and the number of channels. Each upsampling output concatenates with the encoder layers with the same shape. The blue block represents a fully connected layer.

We use the modified U-Net architecture from [101], shown in Figure 2.5. The original U-Net structure is shown in Figure 2.6. The left block represents the downsampling (encoder) and the right block shows the upsampling (decoder). During the transfer learning process, only the weights of the encoder are copied to the target network for fine-tuning. The input image is size 256 × 256 and the encoder has 8 downsampling blocks. Each downsampling block consists of two 3 × 3 convolutional layers, each followed by batch normalization and a Rectified Linear Unit (ReLU). After the ReLU, the input is downsampled by a 2 × 2 max

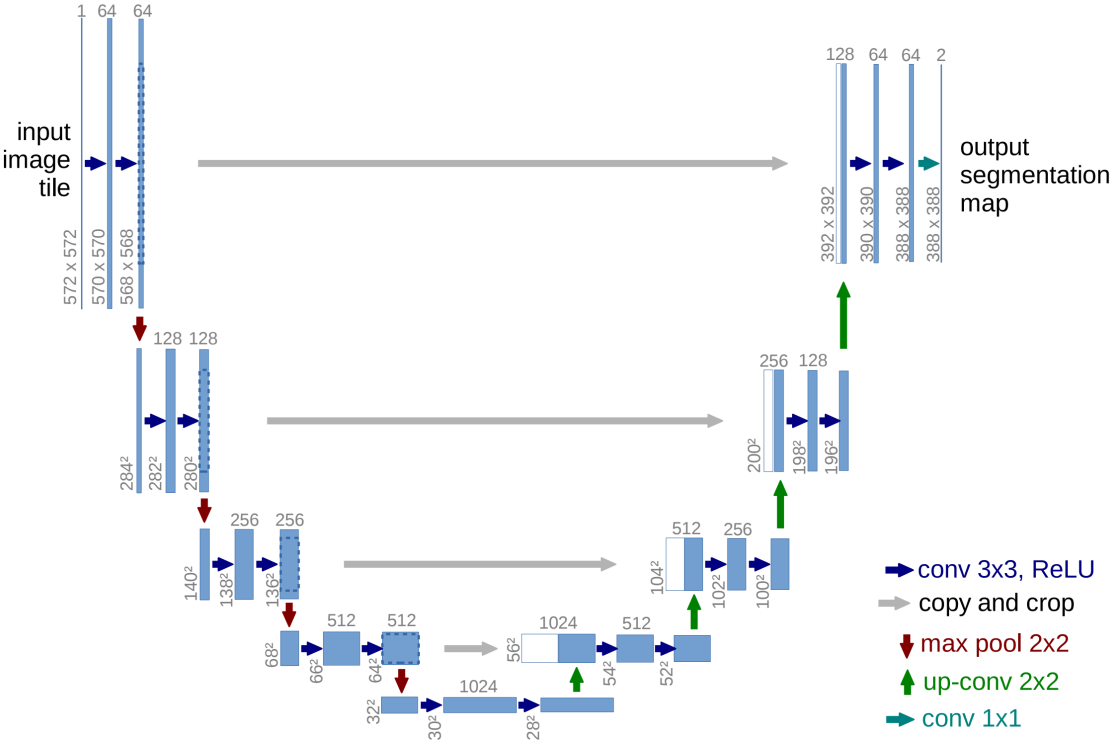


Figure 2.6. Original U-Net structure [91].

pooling layer with stride 2. The number of channels doubles in the first five blocks, going from 64 to 512, while the last three are kept at 512 while still being downsampled. Compared to the original U-Net [91] architecture, this network has 4 more downsampling blocks. It also removes the convolutional bridge structure after the last downsampling block in the original U-Net [91]. The upsampling block is similar to the one in the original U-Net [91] architecture. It concatenates two inputs, one from the previous upsampling block output, and the other from the downsampling block with the same shape as the previous upsampling block output. The number of channels doubles during concatenation but eventually returns to the original number of channels when sent to the last convolutional layer of each upsampling block. The network decoder output is a saliency map, shown in Figure 2.7 as the “Estimated Map”. A pixel on the saliency map has a range of $[0, 1]$ to indicate the object existence in the image. Otsu thresholding [105] is used on the saliency map to generate the threshold image. Additionally, the network has fully connected layers that concatenate the input of the last layer of the encoder and the last layer of the decoder. The output of these fully connected

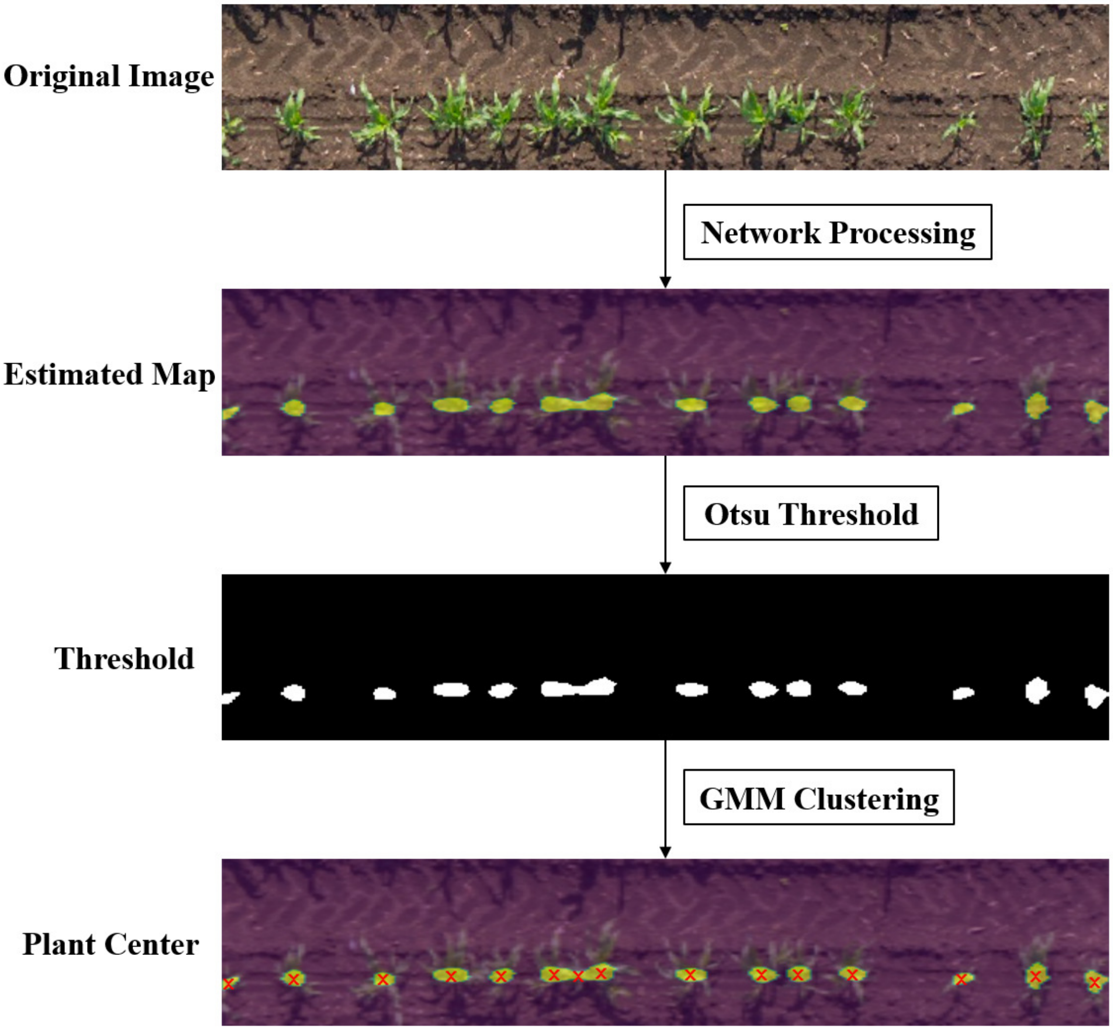


Figure 2.7. Plant center estimation diagram. Each plant center is the center of a cluster and is labeled with a red cross.

layers is the estimated number of plant centers. The plant centers are estimated with a Gaussian mixture model using expectation maximization (EM) [106]. In a Gaussian mixture model, each plant segmentation is considered a cluster, and the number of plant centers is the number of clusters. The cluster centers are the estimated plant centers.

2.4 Experimental Results

In the experiments, our datasets are extracted from an orthomosaic image [24] of a maize field captured using a UAV on May 22, 2018. The UAV was flying at an altitude of 50m.

The orthomosaic image [24] has a spatial resolution of 1cm/pixel. The ground truth region where manual plant center labeling was performed is shown in the blue, green, and red boxes in Figure 2.8 (b), consisting of 5,500 individual plants and their labeled centers. The ground truth region was split into 80% for training (blue box in Figure 2.8 (b)), 10% for validation (green box in Figure 2.8 (b)), and 10% for testing (red box in Figure 2.8 (b)). We randomly extract 2,000 images from the training region as the training dataset and 200 images from the validation region as the validation dataset. The testing dataset also consists of 200 randomly extracted images from the region captured in the red box in Figure 2.8 (b). Because of this random extraction, all three datasets consist of images that can have high overlap. Since the ground truth region was first split into separate regions before the extraction, the datasets have no common images, which prevents testing on training data. The width and height of the randomly extracted images are uniformly distributed between 100 pixels and 500 pixels.

We use Precision [107], Recall [107], F1 Score [107], Mean Average Hausdorff Distance (MAHD), Mean Absolute Percent Error (MAPE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE) related to plant location as our testing metrics. These are defined as:

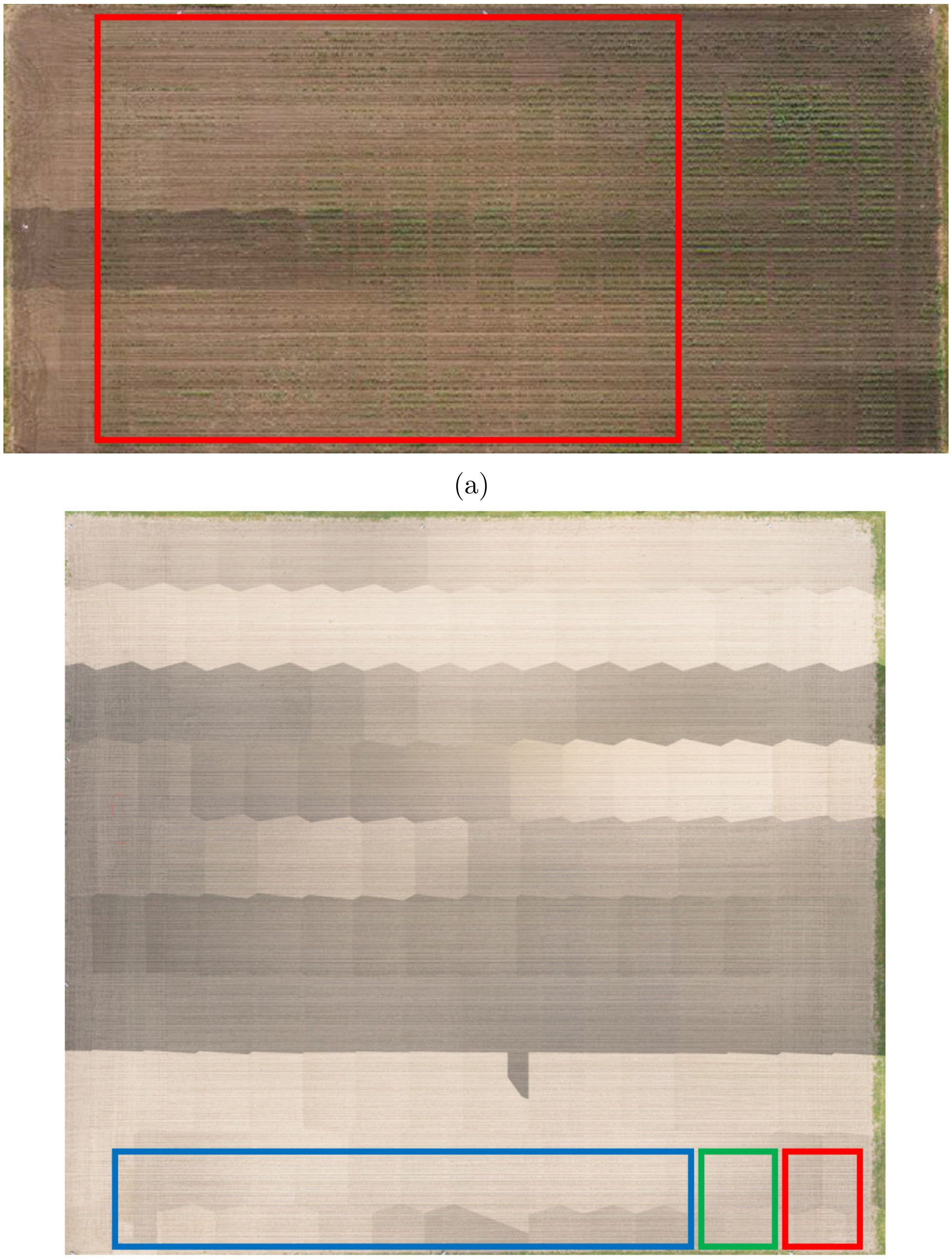
$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.4}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.5}$$

$$\text{F1 Score} = \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.6}$$

$$\text{MAHD} = \frac{1}{|X|} \sum_{x \in X} \min_{y \in Y} d(x, y) + \frac{1}{|Y|} \sum_{y \in Y} \min_{x \in X} d(x, y) \tag{2.7}$$

$$\text{MAPE} = 100 \frac{1}{N} \sum_{\substack{\mathrm{i}=1 \\ C_\mathrm{i} \neq 0}}^{N} \frac{|\mathrm{e_i}|}{C_\mathrm{i}} \tag{2.8}$$



(a)

(b)

Figure 2.8. a) An orthorectified image [24] of a sorghum field on June 13, 2016. The pre-trained network is trained on the data in the red region. b) An orthorectified image [24] of a maize field on May 22, 2018. The blue region is for training. The red region is for validation, and the green region is for testing. These orthorectified images [24] are not color balanced, resulting in flightline-dependent patterns in intensity.

$$\mathrm{MAE} = \frac{1}{N}\sum_{\mathrm{i}=1}^{N}|\mathrm{e_i}| \tag{2.9}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{\mathrm{i}=1}^{N}|\mathrm{e_i}|^2} \tag{2.10}$$

True positive (TP) is the number of detected plants located in the range of pixels r of the plant center ground reference. False positive (FP) is the number of detected plants located outside the range of pixels r of the plant center ground reference. False negative (FN) is the number of failed detected plants located in the range of pixels r of the plant center ground reference. We find setting $r = 5$ is reasonable for the plant center detection application because is about 5cm, which is within the RMSE of the geometric targets. In Equation 2.7, X and Y are the sets of ground truth plant centers and predicted plant centers, respectively. Consequently, $|X|$ and $|Y|$ represent the number of plant centers in the corresponding set. We use Euclidean distance for the function $d(\cdot, \cdot)$. For MAPE, MAE, and RMSE, C_i is the ground reference of the total number of plants in the i-th extracted image. $\hat{C}_\mathrm{i}$ is the estimated number of plants. $\mathrm{e_i} = \hat{C}_\mathrm{i} - C_\mathrm{i}$. N is the number of plant images. Precision, Recall, and F1 Score can indicate how close the estimated points are to the ground reference points. Multiple plant center detection on a single plant is possible even with a high F1 score. We add MAPE, MAE, and RMSE to account for multiple detections.

We compared the performance of the model between transfer learning and training from scratch. Both networks use the modified U-Net [101] depicted in Figure 2.5. As noted previously, the pre-trained network used in transfer learning is trained on 50,000 randomly cropped images with 15,208 distinct plant centers obtained from an orthomosaic [24] image of a sorghum field acquired on June 13, 2016. The learning rate is set to 10^{-5} for the transfer learning model and 10^{-4} for training from scratch. All training uses Adam [108] optimization with a batch size of 16. We evaluate the network performance based on the validation dataset for each epoch. The model with the lowest average Hausdorff distance on the validation dataset is saved as the best model. Figure 2.9 shows an example of detected plant location. The evaluation metrics are shown in Table 2.1.

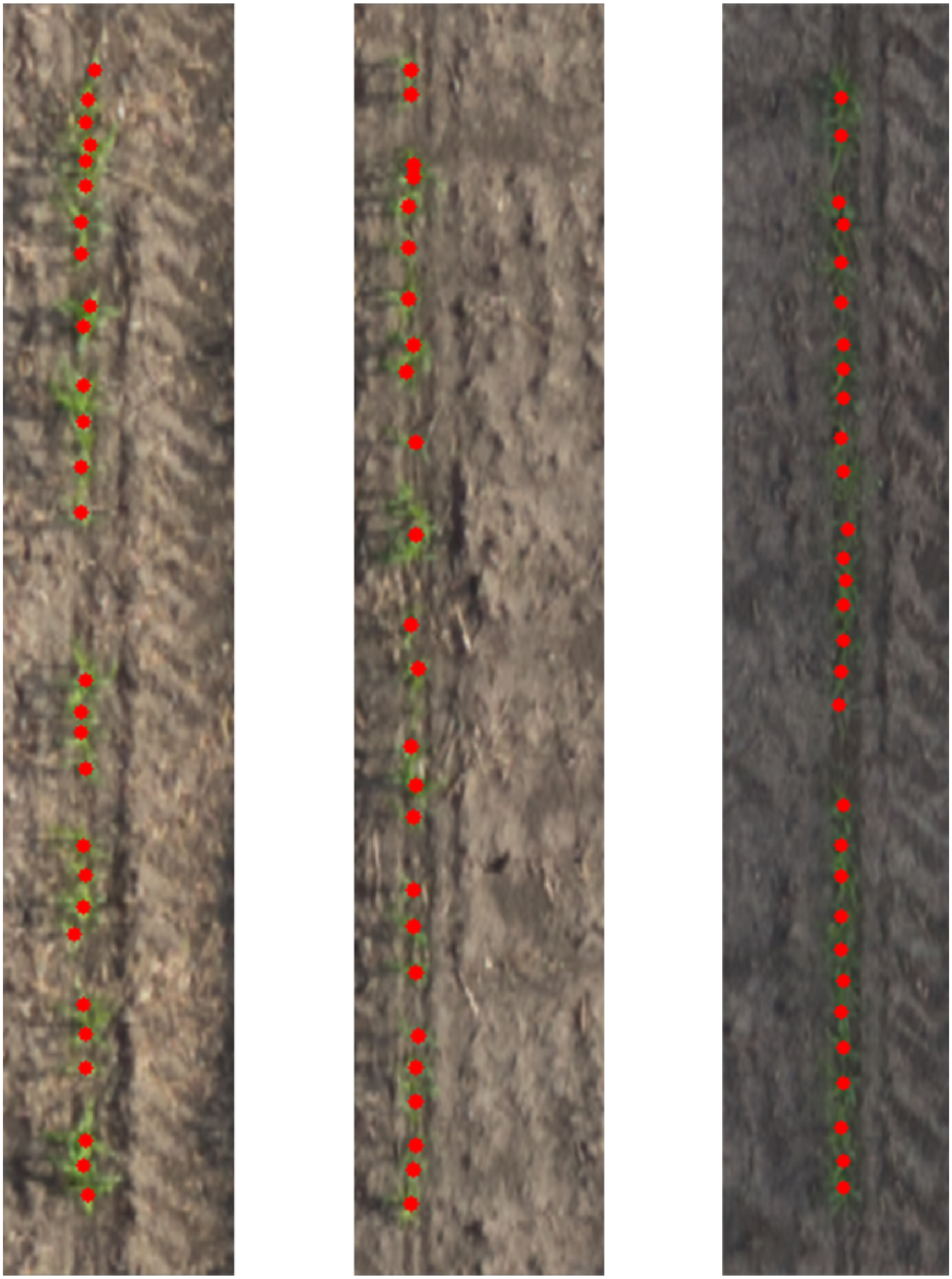


Figure 2.9. Example of plant localization result.

Table 2.1. Results of modified U-Net [101], using $r = 5$.

Metric	Pretrained Network with ImageNet	Non-Pretrained	Fine-Tuning On Pretrained Network
Precision	14.1%	55.1%	**82.6%**
Recall	0.49%	98.5%	**98.9%**
F1 Score	0.94%	70.7%	**90.0%**
MAHD	224.8	8.1	**7.1**
MAPE	100%	125.9%	**8.6%**
MAE	28.9	36.2	**3.9**
RMSE	29.0	36.7	**5.8**

We directly apply the pre-trained network on the maize dataset to evaluate the base performance without any fine-tuning. Note that the sorghum dataset has a dark soil background, while the maize dataset has a light soil background due to drier conditions with plants at a much earlier growth stage. The pre-trained network only has a 0.94% F1 Score. After training (fine-tuning) on 2,000 maize images, the pre-trained network outperforms the network without pre-training on a 90% F1 Score and fewer multiple detections.

We also evaluated the effectiveness of different pre-trained networks in transfer learning. We compared the performance of a model pre-trained on ImageNet [92] with that of a model pre-trained on plant images. The modified U-Net [101] structure does not have a readily-available encoder pre-trained on ImageNet [92]. While we could train an encoder ourselves, training the model on ImageNet [92] with over 1 million images would consume significant resources. There is no guarantee that the resulting network would perform on par with publicly available pre-trained networks, despite the resources invested. Thus, we decided to use a ResNet-50 [72] as the encoder for the modified U-Net [101] in this comparison experiment since ResNet-50 [72] has a publicly available model pre-trained on ImageNet [92]. The learning rate is set to 10^{-5} for both networks. We use Adam [108] optimization with a batch size of 16. The results are shown in Table 2.2. The ImageNet [92] pre-trained network performs better than the network without pre-training. The ImageNet [92] pre-trained network did worse than the plant image pre-trained network because the source

Table 2.2. Results of ResNet [72] encoder modified U-Net [101], using $r = 5$.

Metric	**Pretrained with ImageNet** [92]	**Pretrained with Plant Images**
Precision	55.4%	**84.3%**
Recall	95.3%	**98.8%**
F1 Score	70.0%	**91.0%**
MAHD	7.8	**6.6**
MAPE	92.4%	**9.1%**
MAE	26.5	**4.3**
RMSE	26.9	**5.8**

domain is too different from the target domain (the more general ImageNet [92] vs. UAV plant images).

We also investigate the effect of the size of the training dataset on the transfer learning result. In addition to the $2,000$ maize images training dataset, we randomly cropped 500, $1,000$, $3,000$, $4,000$, and $5,000$ images from the ground reference region. We use 2 NVIDIA GeForce 1080 Ti GPUs for training. Training with 500 images has the least training time, around 4 hours. Training with 5,000 images has the most training time of 12 hours, as the training time linearly increases with the number of training images. The results are shown in Figure 2.10. The dataset with 2,000 images results in a model that balances performance and training time.

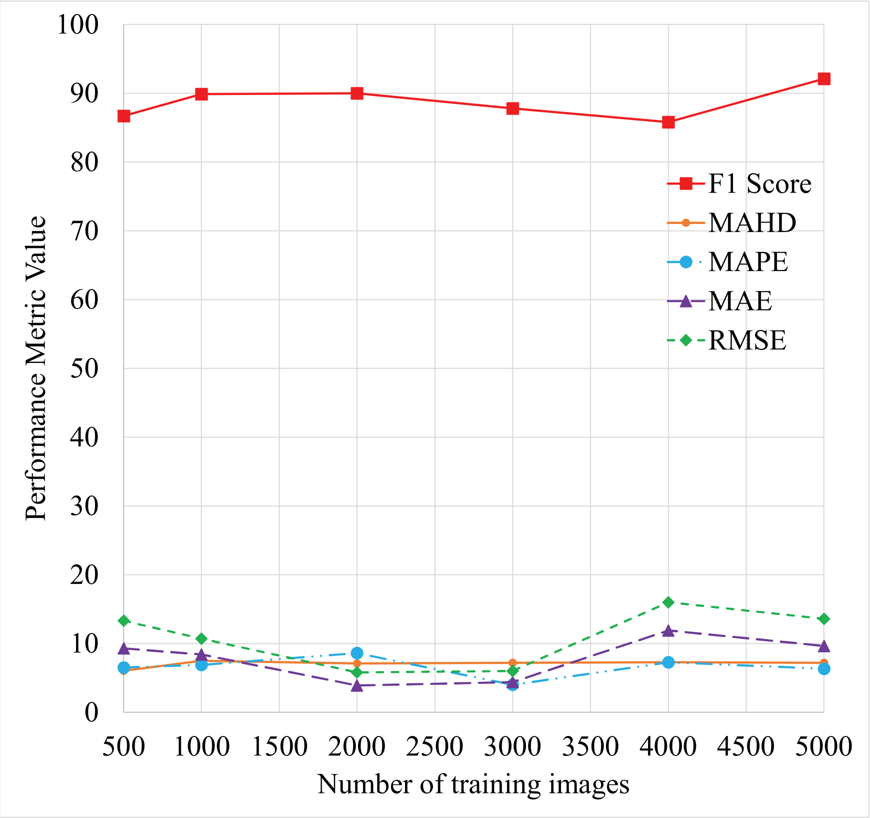


Figure 2.10. Testing results with 500, 1000, 2000, 3000, 4000, and 5000 images in the training dataset. All are trained on modified U-Net [101] structure.

3. SORGHUM FLOWERING TIME ESTIMATION

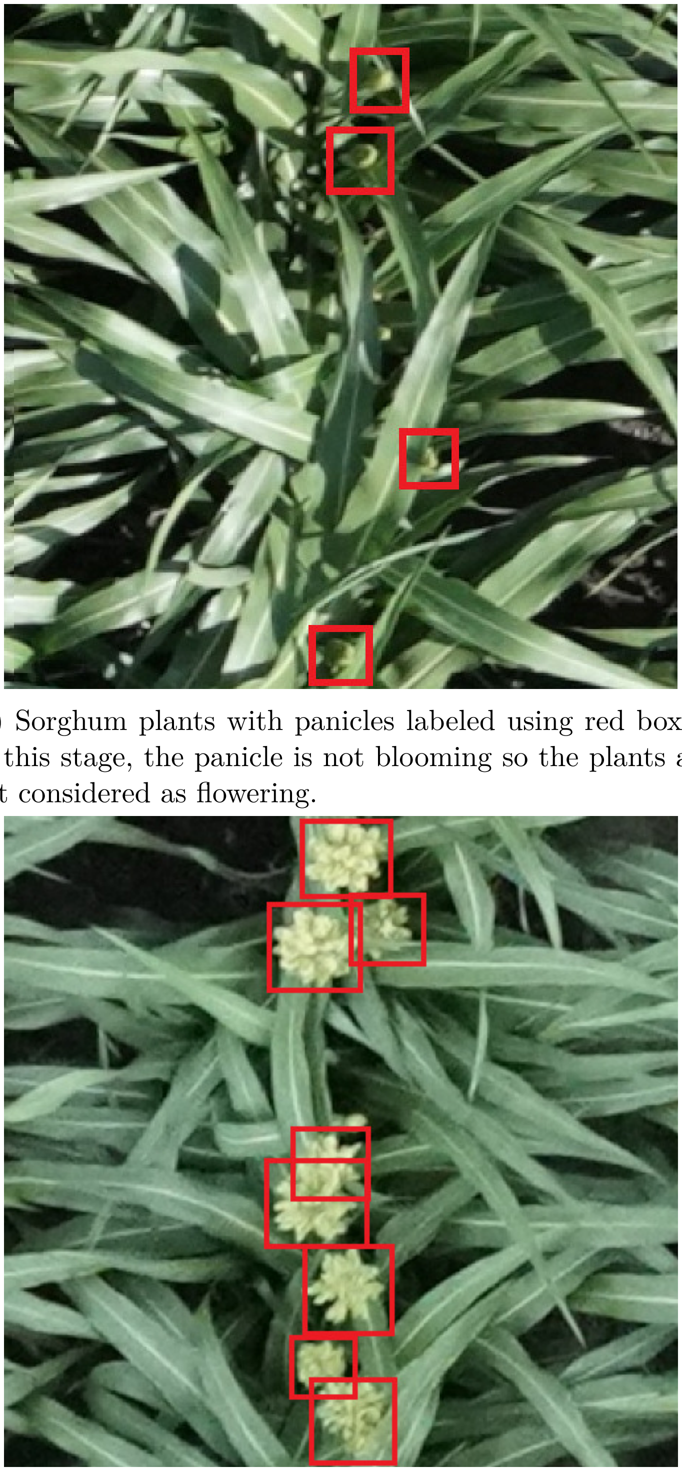
3.1 Introduction

We can evaluate flowering in a sorghum plant by observing its panicles as shown in Figure 3.1. While we are unable to determine the state of flowering of individual panicles due to the resolution of most imagery, we can consider counting across temporal data as a potential surrogate measure, as the capability to detect panicles increases when the flowers emerge from the tight panicle. In this chapter, we investigate the panicle detection performance of multiple networks and use the counts of the best network for flowering time estimation.

3.2 Overview Of Related Work

Deep neural networks have shown some progress in detecting and counting plant traits [109]–[116]. In [110], a weakly supervised deep learning framework with RetinaNet [117] is developed to detect and count sorghum panicles. Inside the framework, a semi-trained CNN model is used to perform synthetic annotation. The weakly supervised strategy is used to reduce the human labeling cost. The network was trained with a single image. After training, labels are generated from a randomly selected image using the network. The generated labels are corrected by a human and then feed into the training system for another training. Figure 3.2 shows the weakly supervised process. Similarly, in [111], an active learning method with Faster-RCNN [99] is used for panicle detection in cereal crops as shown in Figure 3.3. A novel query system is designed for the active learning process with point supervision. The model interacts with a human by querying the labels from the most informative images instead of all images in the dataset.

Weakly supervised learning and active learning can reduce human effort in data labeling. However, they still require human interaction during the training process. In chapter 5, we introduce methods that can auto-label unlabeled data into training without human interaction.



(a) Sorghum plants with panicles labeled using red boxes. In this stage, the panicle is not blooming so the plants are not considered as flowering.

(b) Flowering sorghum plants with blooming panicles labeled using red boxes.

Figure 3.1. An example of sorghum panicles.

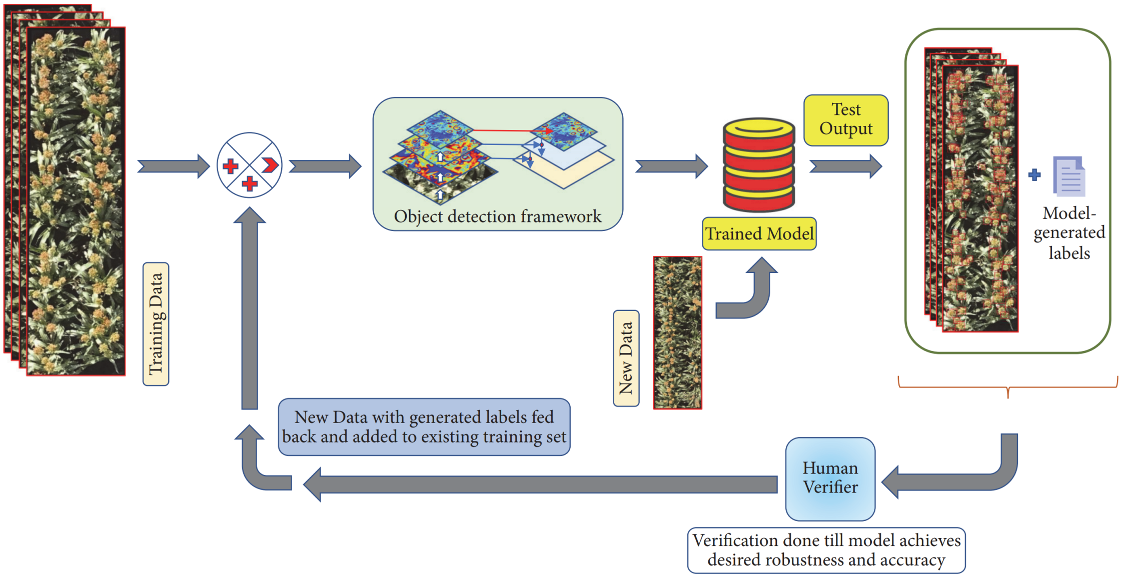

Training Data
Object detection framework
Test Output
Trained Model
Model-generated labels
New Data
New Data with generated labels fed back and added to existing training set
Human Verifier
Verification done till model achieves desired robustness and accuracy

Figure 3.2. Weakly supervised sorghum detection framework from [110]

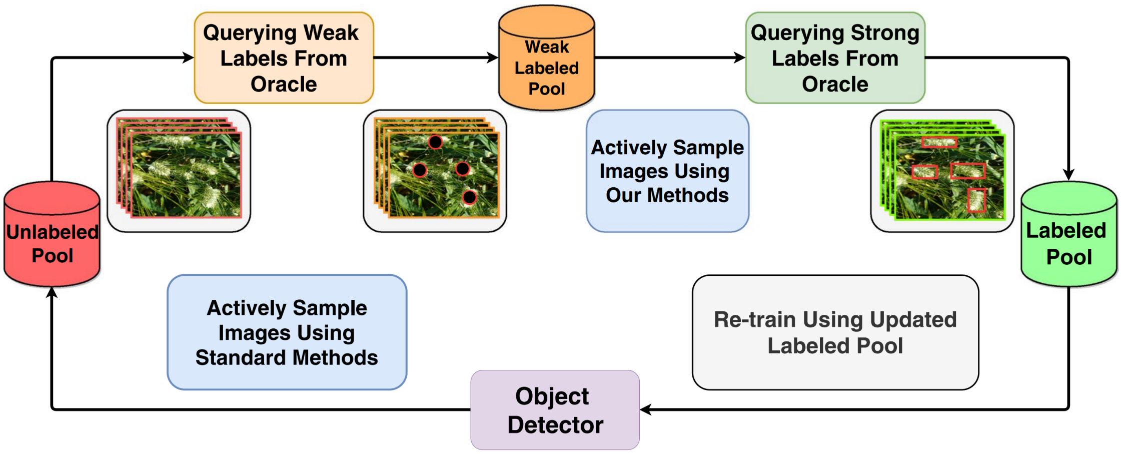

Querying Weak Labels From Oracle
Weak Labeled Pool
Querying Strong Labels From Oracle
Unlabeled Pool
Actively Sample Images Using Our Methods
Labeled Pool
Actively Sample Images Using Standard Methods
Re-train Using Updated Labeled Pool
Object Detector

Figure 3.3. Active learning for panicle detection from [111]

3.3 Flowering Time Estimation

Our method consists of multi-temporal panicle detection and flowering time series estimation, as shown in Figure 3.4. The orthorectified images are cropped into smaller segments for panicle detection and counting. Polynomial regression is applied to the counting results from multiple dates to estimate the flowering time.

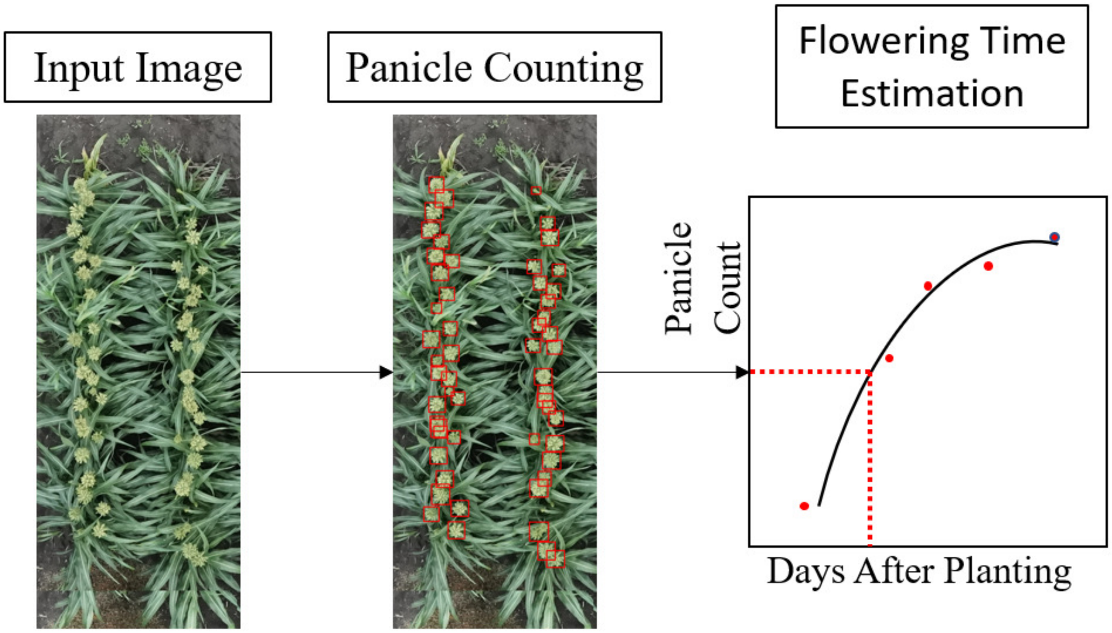


Figure 3.4. Our approach to flowering time estimation.

We chose the deep networks for panicle detection based on their performance on a general object detection dataset such as COCO [118]. We selected three detection-based deep networks for panicle detection. For early dates in the time sequence, some panicles that did not bloom can still be detected by the network. We set a threshold for the bounding box size to remove them. We then fit a third-degree polynomial to the estimated counting data to obtain the panicle count time series as shown in Figure 3.8 with the counts in Table 3.3. The estimated flowering time is the intersection between the line associated with half of the ultimate number of panicles counted and the flowering curve.

3.3.1 Network Architecture

RetinaNet. RetinaNet [117] is a one-stage detection-based network with focal loss as the loss function as shown in Figure 3.5. It uses ResNet [72] and feature pyramid network (FPN) [119] as backbone networks. Each level of the FPN is connected with a sub-network for bounding box regression and object classification. The focal loss is used in the classification sub-network:

$$\mathrm{FL}(p_t) = -(1-p_t)^{\gamma} log(p_t) \tag{3.1}$$

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{otherwise,} \end{cases} \tag{3.2}$$

where p is the estimated probability of the model and $\gamma \geq 0$ is the focusing parameter. In our experiments, we choose ResNet-101 with FPN as the backbone for RetinaNet.

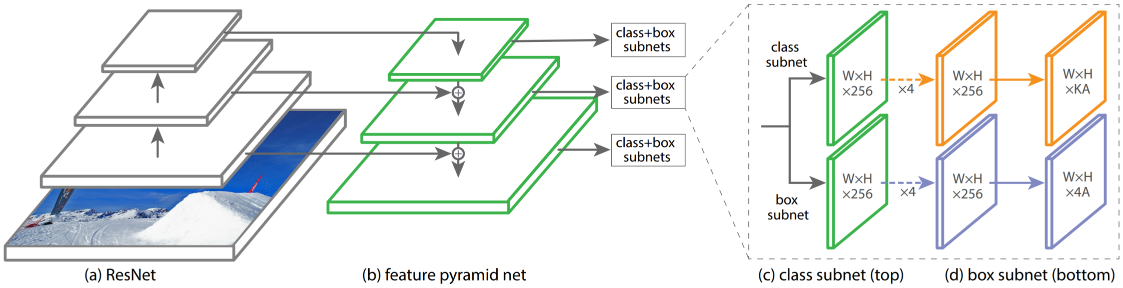


Figure 3.5. RetinaNet structure [117].

YOLOv5. YOLOv5 [120] is a one-stage detection-based network. The general structure of YOLOv5 consists of backbone, neck, and prediction as shown in Figure 3.6. YOLOv5 uses CSPNet [121] as backbone architecture. FPN [119] and Path Aggregation Network (PANet) [122] are used for the neck of YOLOv5. There are four different versions of YOLOv5. The main differences between the versions are the depth and width. We chose the YOLOv5x model for our experiments since it has the best accuracy across the different versions.

Faster-RCNN. Faster-RCNN [99] is a two-stage detection-based network consisting of a feature map extractor, regional proposal network (RPN), and Region of Interest (ROI) pooling and classification network as shown in Figure 2.3. The main idea of Faster-RCNN is to use RPN to generate bounding boxes. We use the ResNet-101 with FPN as the feature map extractor in the Faster-RCNN model.

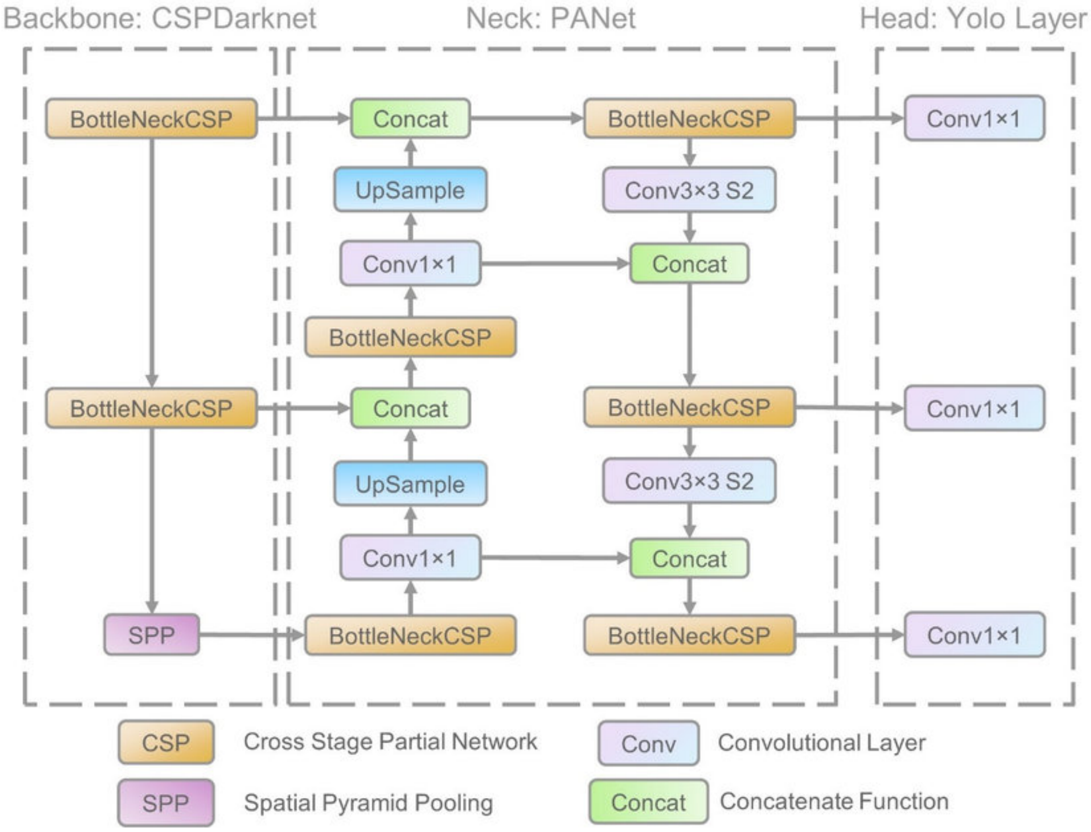


Figure 3.6. YOLOv5 structure from [123].

3.4 Experimental Results

For panicle detection training and testing, we use an RGB orthomosaic [24] photo of a sorghum field in West Lafayette, Indiana, USA acquired by a Sony ILCE-7RM3 camera mounted on a DJI Matrice 600 Pro platform on July 22, 2020 at 20m altitude. The orthomosaic photo is cropped into individual images of 2 row segments of plants. Each cropped image is horizontally divided into two sub-images. The images are further separated for training, validation, and testing. We manually ground truth the images by labeling each panicle with a bounding box. In total, we have 500 images for training, validation, and testing. The images have dimensions of 800 Œ 600 pixels which are resized to 512 Œ 512 pixels during training. Flowering time was estimated for a field of sorghum test plots (~200,000 plants/hectare), comprised of two replicates of 80 varieties in a randomized block design (plot size: 7.6m Œ 3.8m), 10 rows per plot. In practice, the flowering time varies for different genotypes of sorghum. For this specific hybrid genotype, with a planting date of May 13, 2020, we select the multi-temporal RGB images from 65, 68, 70, 76, 79, and 83 days after planting. Each image is cropped from the associated orthomosaic photo with a size of 3000 Œ 1200 pixels. The cropped image has 8 row segments of plants because 2 rows in the middle were destructively sampled for biomass. The ground truth data is obtained by manually counting panicles in these cropped images.

We split the 500 images into training (80%), validation (10%), and testing (10%). For all three networks, we start with models pre-trained on the COCO dataset, as this reduces training time. The learning rate is set to 0.00001 for three networks. The training time for each network is around 30 minutes using 4 NVIDIA GTX 1080 Ti graphics cards. Validation is performed every 10 epochs.

We use Average Precision (AP) with Intersection over Union (IoU) set to 0.5 for panicle detection. We use Mean Absolute Percent Error (MAPE) [107], Mean Absolute Error (MAE) [107], and Root Mean Squared Error (RMSE) [107] for panicle counting.

We evaluate the performance of the three networks with the validation and testing datasets. The results are shown in Table 3.1 and Table 3.2. Faster-RCNN and YOLOv5 are better than RetinaNet based on four metrics. YOLOv5 has a similar AP and better MAPE, MAE, and RMSE compared to Faster-RCNN. Based on these results, we use YOLOv5 as the network architecture for flowering time estimation.

Table 3.1. Evaluation of validation dataset.

Metric	**RetinaNet**	**YOLOv5**	**Faster-RCNN**
AP	86.6	89.1	**89.8**
MAPE	0.3	**0.2**	0.3
MAE	1.8	**1.2**	1.5
RMSE	2.5	**1.8**	2.2

The shape and color of panicles varied for each variety of sorghum. We select the variety based on the similarity of our training data. We use our panicle counting deep network to estimate the counts for each test image without resizing. Figure 3.7 shows an example of

Table 3.2. Evaluation of testing dataset.

Metric	RetinaNet	YOLOv5	Faster-RCNN
AP	83.8	**86.2**	86.1
MAPE	0.2	**0.1**	0.2
MAE	3.1	**1.5**	2.6
RMSE	4.0	**2.0**	3.2

Table 3.3. Flowering time estimation.

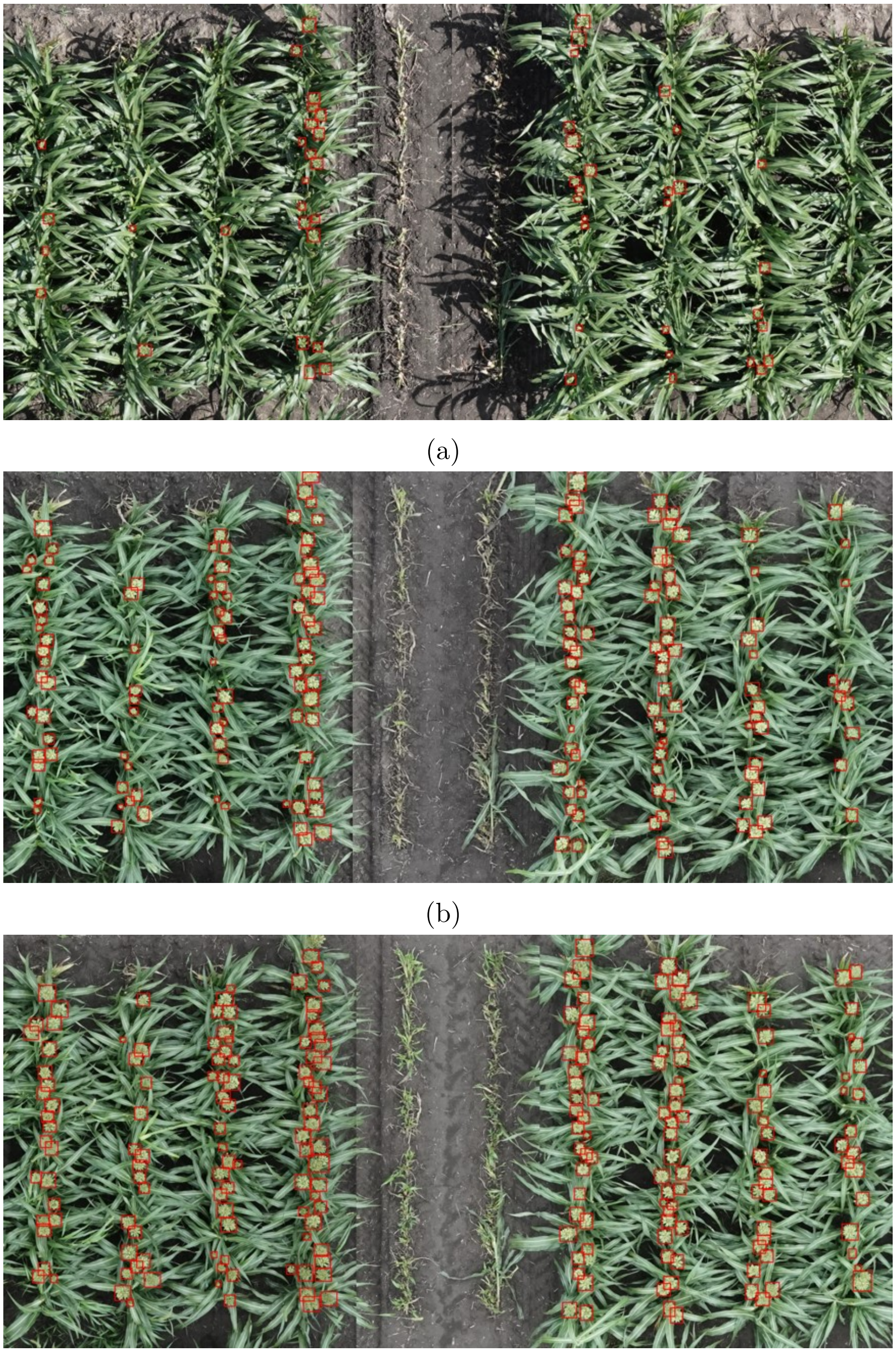
Days After Planting	Manual Count	Estimated Count
65	35	34
68 **(Est. Flowering Time)**	151	157
70	198	202
76	259	253
79	278	276
83	280	278

Table 3.4. Flowering time estimation metrics.

MAPE	MAE	RMSE
0.03	2.43	3.05

multi-temporal panicle detection results. For this specific plot, our estimated flowering time is 68 days after planting which is nearly identical to the result from the manual counts.

We also test the methods on all 160 plots in the same panel. The results are shown in Figure 3.9 and Table 3.4. From the results, there is a 2 to 3 days bias towards the reference dates. We only count the number of panicles, not the number of flowering panicles so the estimated dates are slightly off from the reference dates.



(a)

(b)

(c)

Figure 3.7. a) Panicle detection result on 2020-07-17. b) Panicle detection result on 2020-07-20. c) Panicle detection result on 2020-07-22.

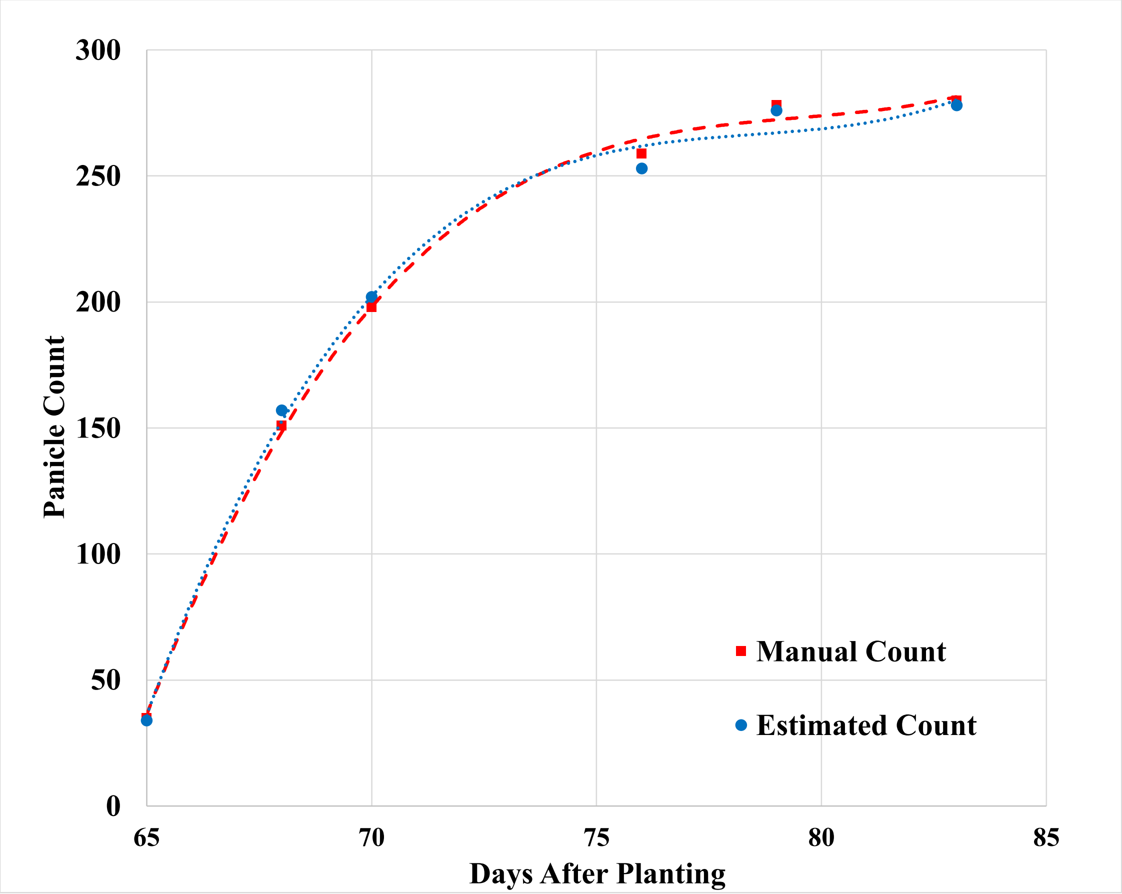


Figure 3.8. Panicle count time series.

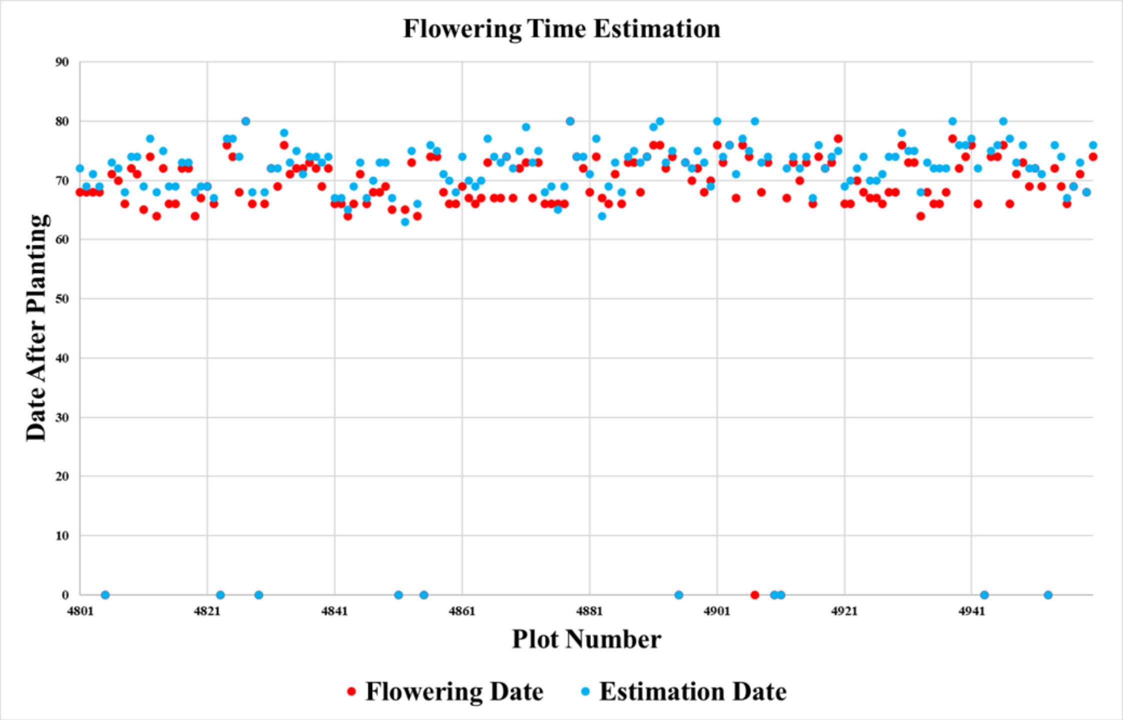


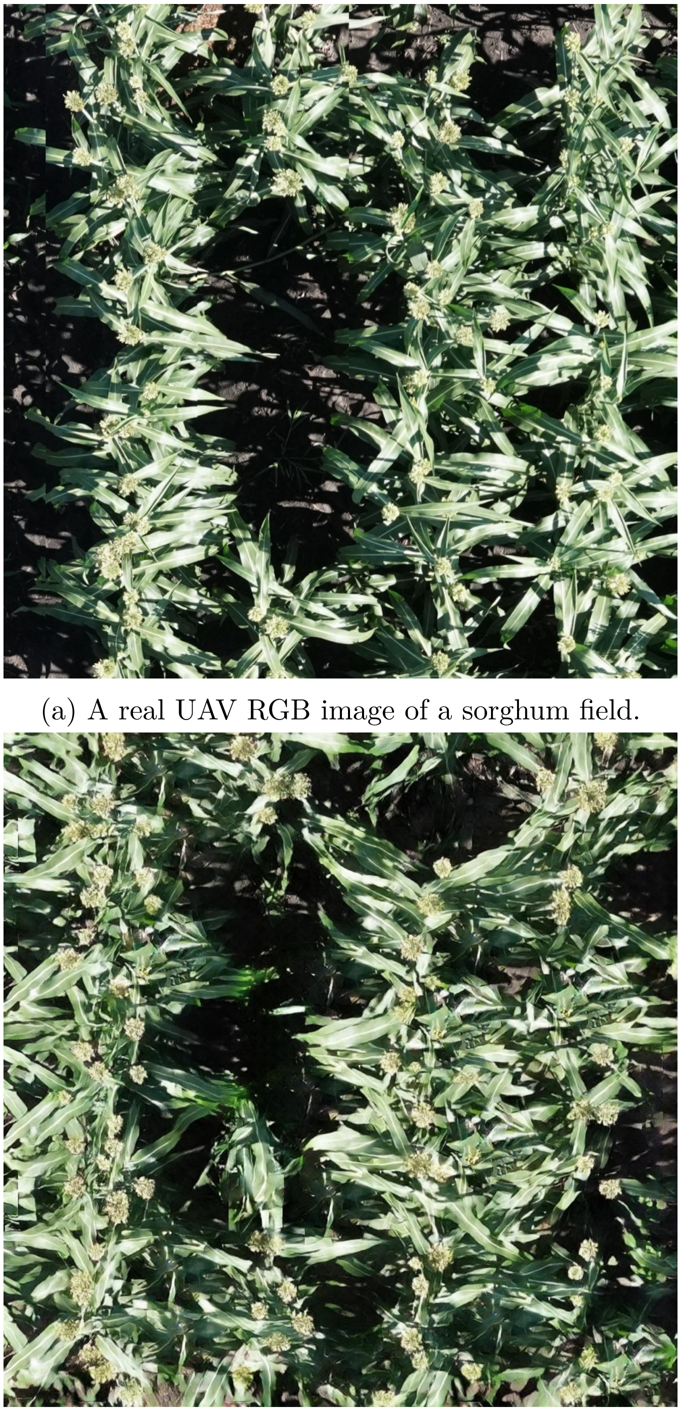
Figure 3.9. Flowering time estimation results of 160 plots.

4. IMPROVING SORGHUM PANICLE DETECTION

4.1 Introduction

Deep convolutional neural networks (CNN) [124] have achieved outstanding results in image classification, object detection, and segmentation [71], [99], [100]. The high accuracy for identifying objects in images with complicated backgrounds makes CNNs attractive for estimating phenotypic traits. For deep neural networks, a large quantity of training data is often required to prevent overfitting. Labeling a large amount of training data is time-consuming and tedious due to the high spatial resolution and density of panicles in plant images. The lack of training data is a major bottleneck that affects the performance of the deep neural network in plant trait estimation. Data augmentation [125] is a technique to reduce overfitting by creating more training data samples such as rotation, flip, and crop on the existing dataset without additional ground truth. Adding synthetic images has become popular for data augmentation since the introduction of generative adversarial networks (GANs) [126], which can generate realistic images. GAN-based methods have a positive impact on enhancing the performance of classification tasks for plant images [127], [128]. Most GAN-based approaches focus on classification by generating images containing a single plant instance. In plant trait estimation such as panicle detection, there are multiple instances of plants in an image, so the ability to generate images containing multiple objects with labels is necessary.

In this chapter, we propose a method for generating high-resolution synthetic sorghum UAV RGB images with bounding box labels using two image-to-image translation GANs. We create a label map (or label images) that contains bounding boxes where we want the panicle to be located in the synthetic image generated by the GAN. We shall denote each bounding box in the label map as a "mask". The GAN then generates a synthetic image with the panicle located as proscribed in the label map. We utilize a small amount of ground truth real images to train the image-to-image translation GAN with the label map. The synthetic images are combined with real images to train the object detection network. We compare the performance of the two image-to-image translation GANs. The results show performance



(a) A real UAV RGB image of a sorghum field.

(b) A synthetic UAV RGB image of a sorghum field generated from our GAN.

Figure 4.1. An example of real and synthetic UAV RGB images of a sorghum field.

improvement in both panicle detection and counting. Figure 4.1 shows an example of real and synthetic UAV RGB images of sorghum.

4.2 Overview Of Related Work

4.2.1 Generative Adversarial Network (GAN)

The basic idea of a GAN [126] is to generate images that are indistinguishable from real images. A GAN consists of two models: the generator model for image generation and the discriminator model for classification of real and synthetic images. It has the following objective function:

$$\min_{G}\max_{D} V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))] \tag{4.1}$$

where $V(D,G)$ is the value function, $p_{data}(x)$ is the distribution over data x, $p_z(z)$ is the input noise variable, D is discriminator and G is generator. The generator often uses an inside-out autoencoder [129] structure that takes random noise as input and generates synthetic images. The discriminator uses a CNN and multilayer perceptron (MLP) [130] as a classifier to distinguish the real and synthetic data. During the training of the GAN, the objective of the discriminator is to maximize the cost value, while the objective of the generator is to minimize the cost value. Developing a stable training approach is the main challenge in recent GAN research [131]. Since the GAN has two models (Generator and Discriminator), it is difficult to train both models simultaneously while ensuring convergence. There are a variety of GAN variants that focus on improving training stability. For example, Arjovsky *et al.* proposed the Wasserstein GAN [132] with a new loss function based on the Earth Mover's Distance [133] to enhance training stability. Radford *et al.*developed a new model DCGAN [134] using a novel CNN architecture for both the generator and discriminator. The conditional GAN has the following objective function:

$$\min_{G}\max_{D} V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x \mid y)] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z \mid y)))] \tag{4.2}$$

The function is similar to the function from the original GAN with an additional condition y. Figure 4.2 shows a simple example of conditional GAN structure.

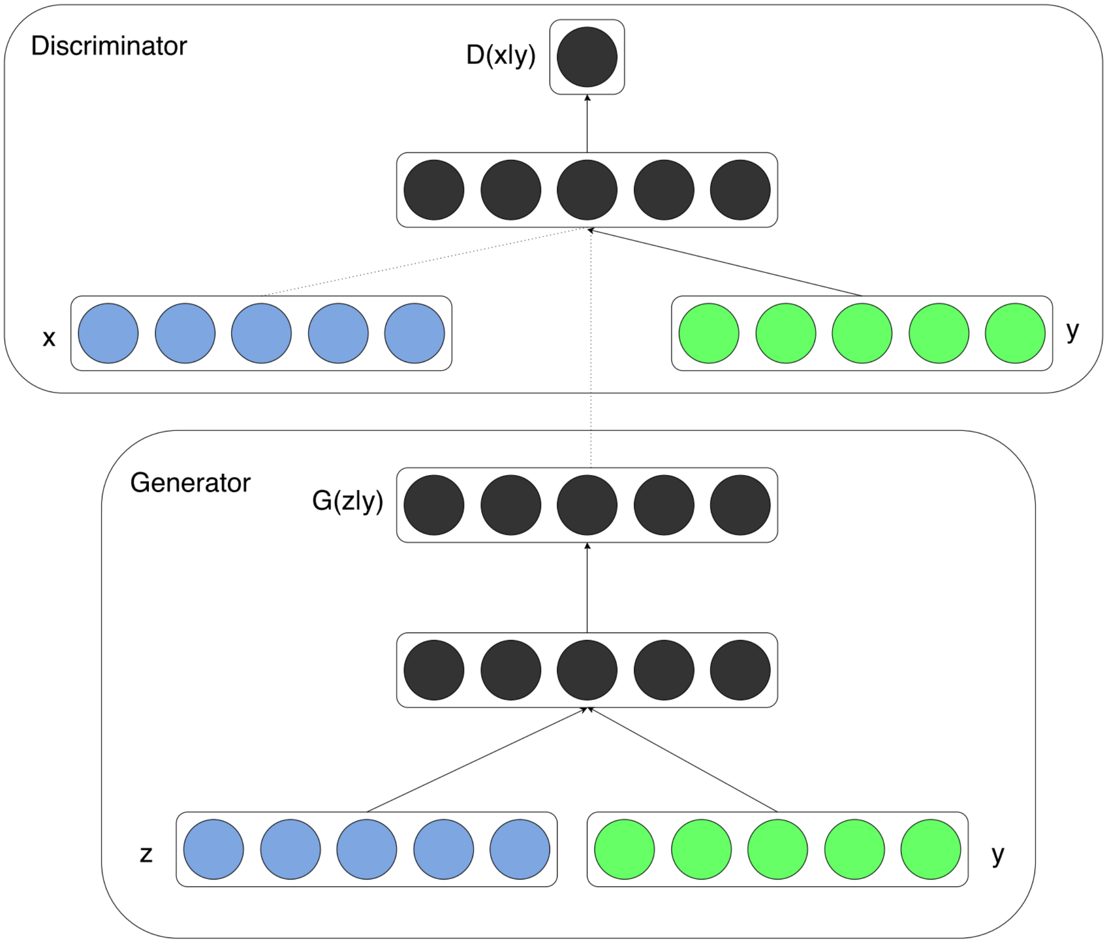
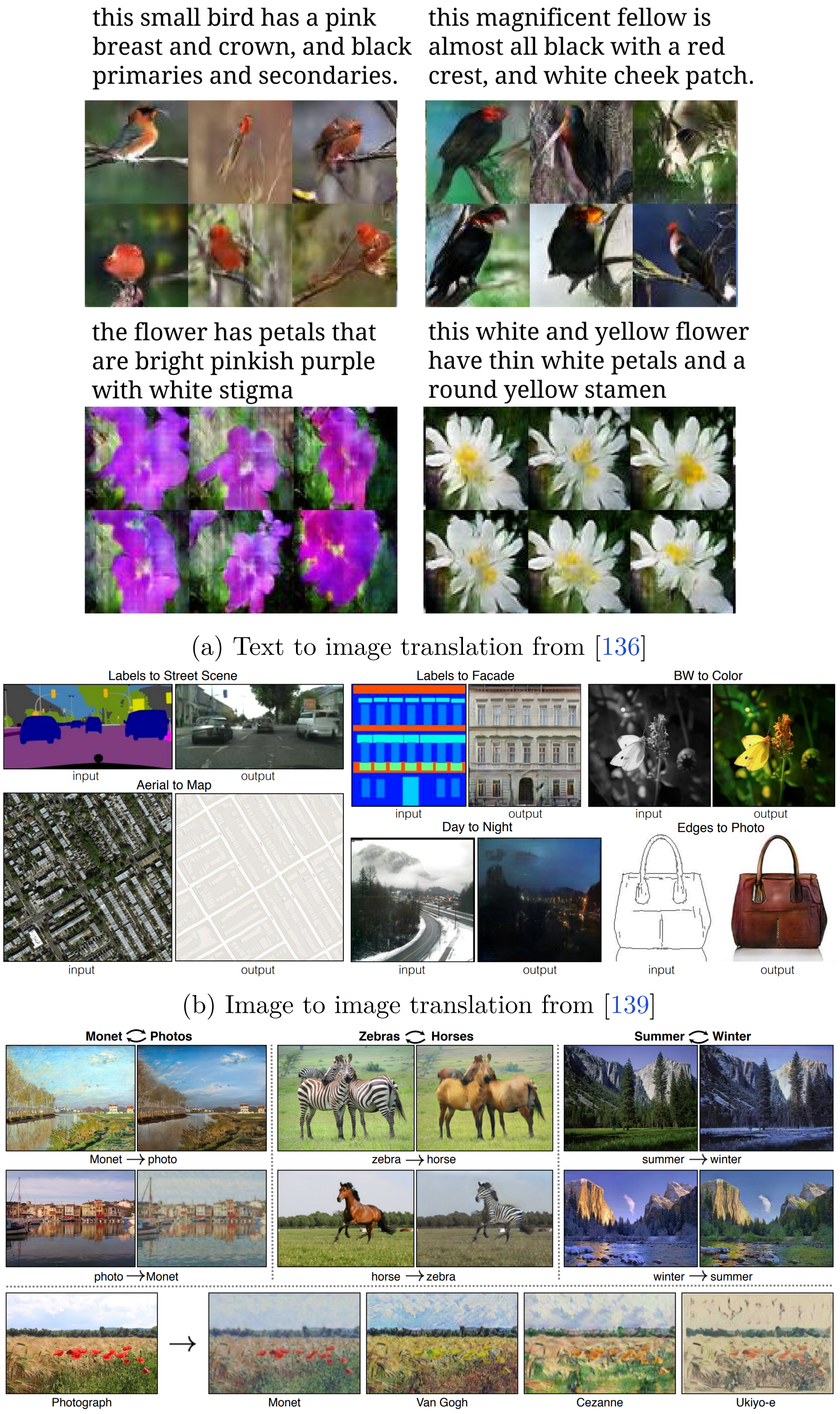


Figure 4.2. Conditional GAN structure from [135].

The Conditional GAN (cGAN) [135] is a variant of a GAN in which auxiliary inputs, such as labels or text, are added as inputs to both the generator and discriminator so that the additional information can control the output. The Conditional GAN has wide applications such as text-to-image translation [136], image-to-image translation [137], and style transfer [138]. Examples are shown in Figure 4.3.



(a) Text to image translation from [136]

(b) Image to image translation from [139]

(c) Style transfer from [140]

Figure 4.3. Applications of GANs.

4.2.2 Data Augmentation Using Synthetic Images

The purpose of data augmentation is to create or modify an existing training dataset to improve the performance of machine learning models [125]. Figure 4.4 show a taxonomy of image data augmentation techniques.

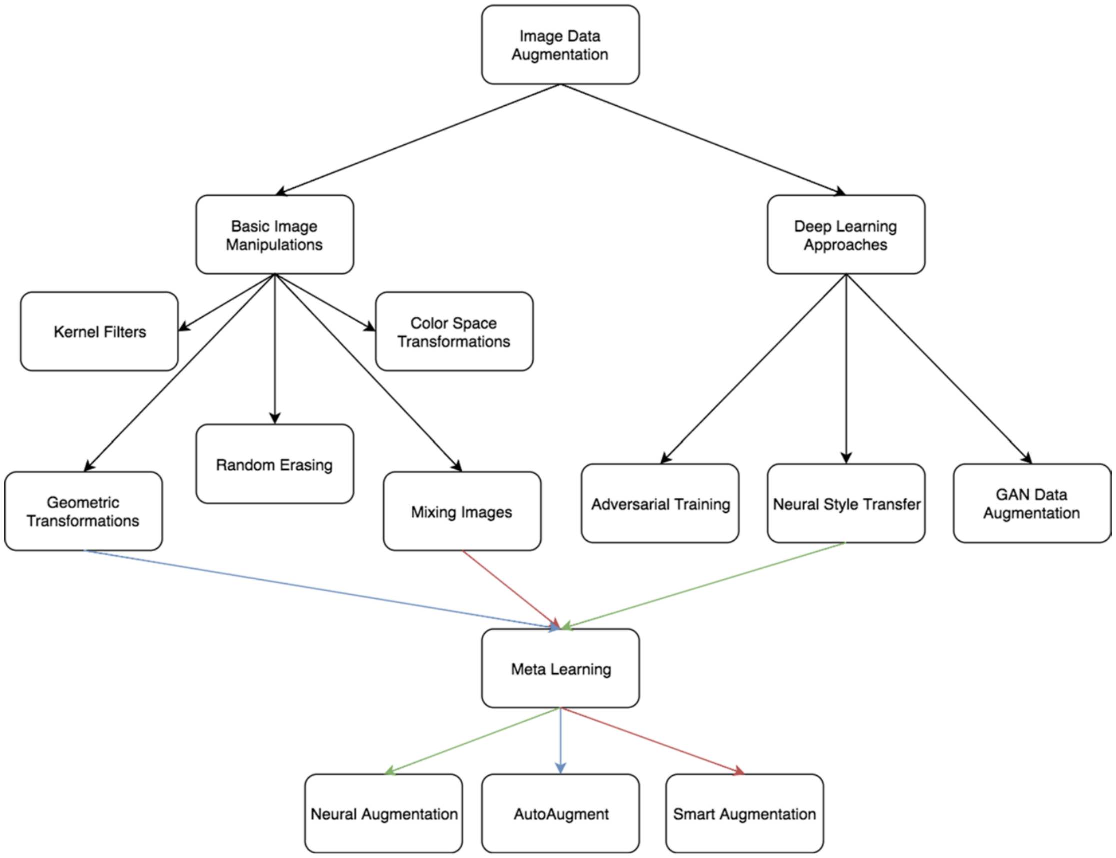


Figure 4.4. A taxonomy of data augmentation from [125].

Generating synthetic images from GANs for data augmentation is widely used to enhance the performance of deep neural networks for various tasks where there is insufficient training data [125]. The synthetic data is combined with real ground truth images to train a target network. Most GAN-based data augmentation methods are used to improve classification that identifies a single object from images. For example, in [127], Bi *et al.* developed a method using images generated by a Wasserstein GAN variant to improve plant disease classification. In [128], Madsen *et al.* improved plant seeds classification by using synthetic plant seed images from GAN. The Conditional GAN is often used in situations that require additional annotations for the synthetic images, such as data augmentation for detection and segmentation tasks. For example, in [141], Milz *et al.* proposed a conditional GAN to generate images with ground truth data for aerial object detection and segmentation. In [142], Sandfort *et al.* improved CT segmentation by using synthetic images from CycleGAN [140].

4.3 Approach

Our method consists of two parts: training a conditional GAN with a small amount of ground truth UAV RGB images (label maps) of sorghum and then generating synthetic images with random locations of the panicles which we use for data augmentation. Figure 4.5 illustrates our proposed approach. In the top part of Figure 4.5, label maps and images are used for training the GAN. The label map is a grayscale mask acquired by converting existing ground truth bounding box labels. In the bottom part of Figure 4.5, we use random locations of the panicles (random label map) as an input to the GAN to generate synthetic UAV RGB images.

The synthetic images are then added to the ground truth image dataset to form the training data we use to train the panicle detection network. The requirements for the GAN are that it must be capable of generating high-resolution images constrained by the inputs. We choose two image-to-image translation GANs because the real ground image bounding box labels can be converted directly into label masks. The two GANS we use are pix2pixHD [143] and SPADE [144] because of their reported performance on public datasets such as ADE20K [145] and Cityscapes [146].

4.4 Network Structures

4.4.1 pix2pixHD

The pix2pixHD [143] architecture is an image-to-image translation GAN that can generate high-resolution images from labeled map inputs. The concept of pix2pixHD comes from pix2pix [139], a conditional GAN for image-to-image translation. The original pix2pix has

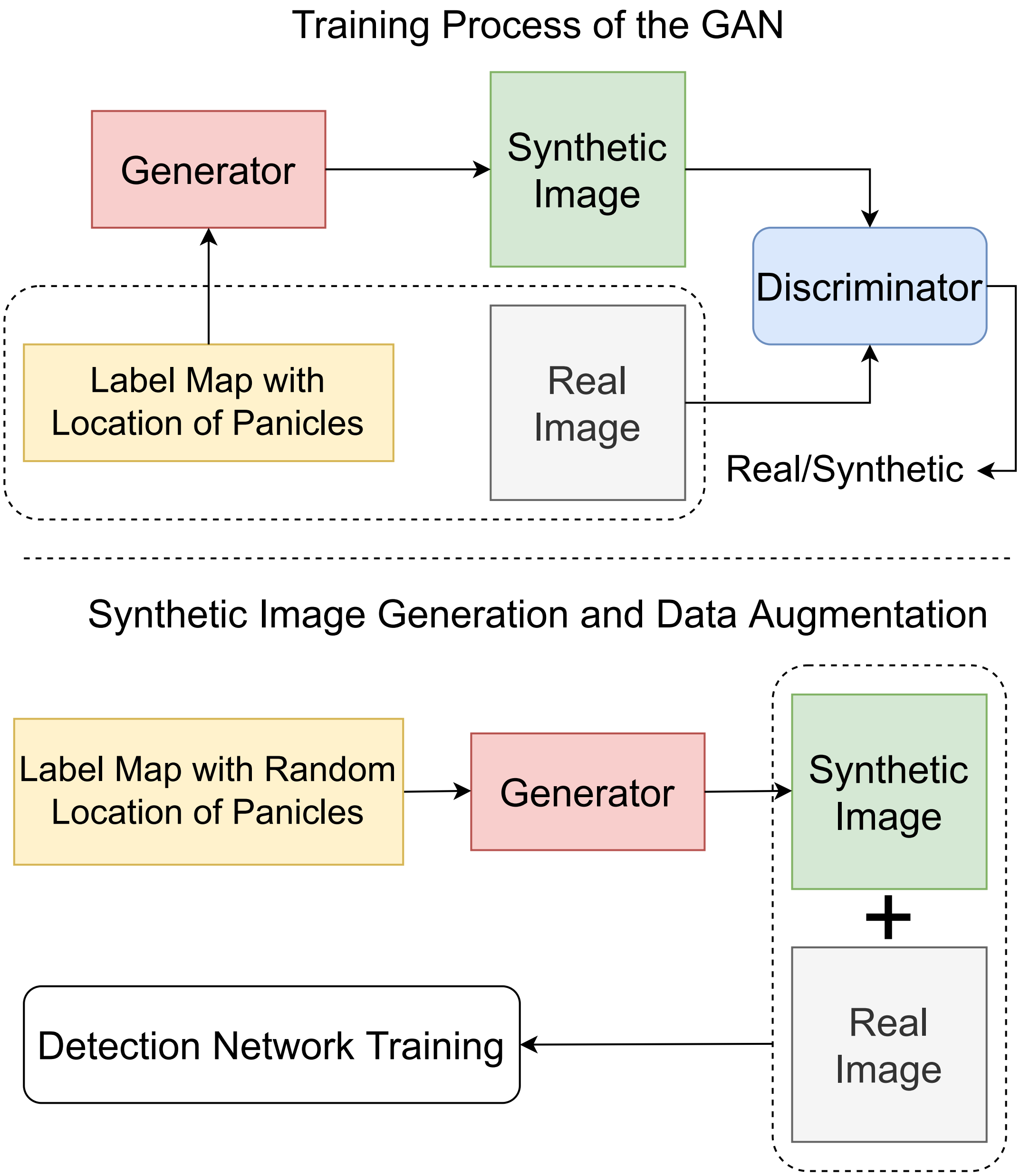

Training Process of the GAN
Generator
Synthetic Image
Discriminator
Label Map with Location of Panicles
Real Image
Real/Synthetic
Synthetic Image Generation and Data Augmentation
Label Map with Random Location of Panicles
Generator
Synthetic Image
+
Real Image
Detection Network Training

Figure 4.5. Block diagram of our proposed approach.

the structure shown in Figure 4.6. The pix2pix structure is comprised of a generator G and a discriminator D. It has the following objective function:

$$\mathcal{L}_{\text{GAN}}(G, D) \equiv \mathbb{E}_{(\mathbf{s},\mathbf{x})}[\log D(\mathbf{s}, \mathbf{x})] + \mathbb{E}_{\mathbf{s}}[\log(1 - D(\mathbf{s}, G(\mathbf{s}))] \tag{4.3}$$

where $(s_{\text{i}}, x_{\text{i}})$ is the i-th sample pair of a label and its corresponding nature image. We denote

$$\mathbb{E}_{(\mathbf{s},\mathbf{x})} \triangleq \mathbb{E}_{(\mathbf{s},\mathbf{x})} \sim p_{\text{data}}\,(\mathbf{s}, \mathbf{x}) \tag{4.4}$$

. The generator and discriminator of pix2pix are derived from the DCGAN [134] which uses a CNN as its main structure. The generator of pix2pix uses skip connections from U-Net [91] to share information between input and output. The discriminator of pix2pix adapts the PatchGAN [139] structure. The PatchGAN is a patch-based CNN that requires fewer parameters to train compared to fully connected structures of other discriminators.

pix2pix can only generate images with a maximum resolution of 256 × 256 pixels. If we train pix2pix with images larger than 256 × 256 pixels, the training process becomes unstable and the image quality deteriorates. pix2pixHD [143] addresses the limitation of pix2pix by introducing multi-scale generators and discriminators. The structure of pix2pixHD is shown in Figure 4.7. It has the following objective function:

$$\min_{G}\left(\left(\max_{D_1,D_2,D_3}\sum_{k=1,2,3}\mathcal{L}_{\text{GAN}}(G, D_k)\right) + \lambda\sum_{k=1,2,3}\mathcal{L}_{\text{FM}}(G, D_k)\right) \tag{4.5}$$

where λ controls the importance of a GAN loss $\mathcal{L}_{\text{GAN}}(G, D_k)$ and a feature matching loss $\mathcal{L}_{\text{FM}}(G, D_k)$. The k-th feature matching loss $\mathcal{L}_{\text{FM}}(G, D_k)$ for the discriminator is incorporated in pix2pixHD to stabilize training:

$$\mathcal{L}_{\text{FM}}(G, D_k) \equiv \mathbb{E}_{(\mathbf{s},\mathbf{x})}\sum_{\text{i}=1}^{T}\frac{1}{N_{\text{i}}}\left\|D_k^{(\text{i})}(\mathbf{s}, \mathbf{x}) - D_k^{(\text{i})}(\mathbf{s}, G(\mathbf{s}))\right\|_1 \tag{4.6}$$

where $D_k^{(\text{i})}$ is the feature extractor of discriminator D_k for i-th layer, N_{i} is denotes the number of elements in the i-th layer and T denotes the total number of layers.

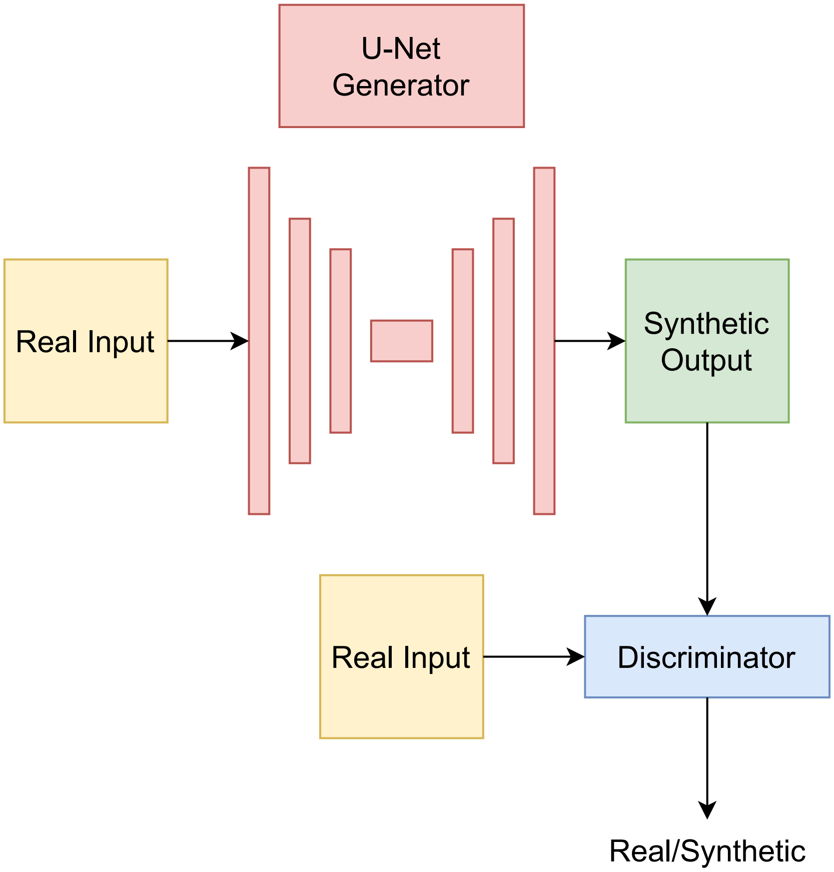


Figure 4.6. The pix2pix structure.

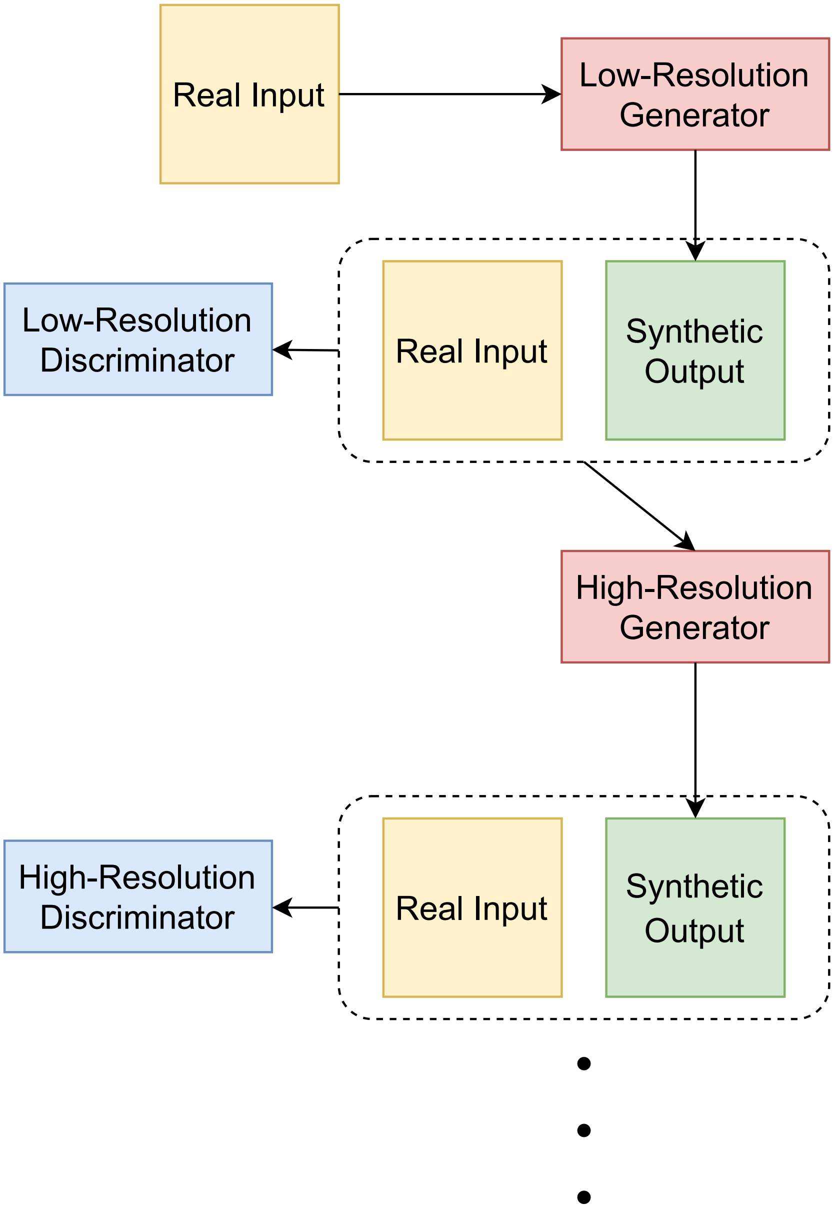


Figure 4.7. The pix2pixHD structure.

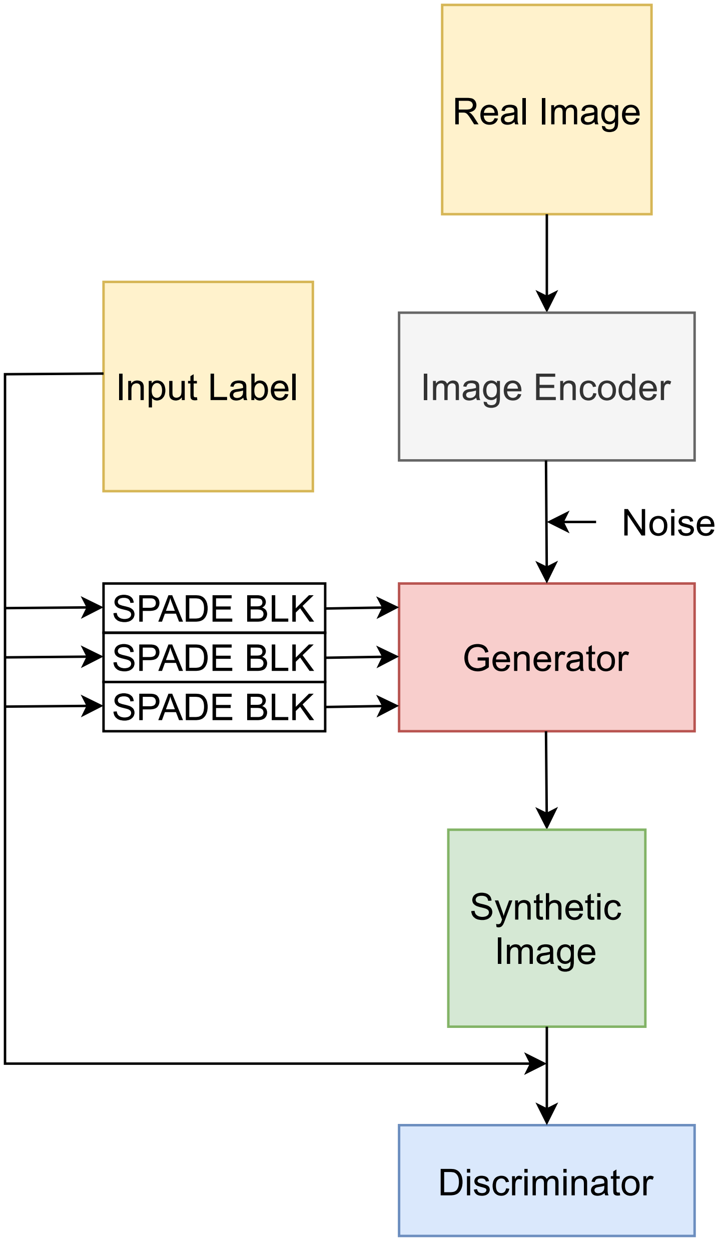


Figure 4.8. The SPADE structure.

The multiscale generators and discriminators form pyramid structures for synthesizing images with different resolutions. The generator structures of pix2pixHD consist of a global generator G_1 and a local enhancer network G_2 as shown in Figure 4.9.

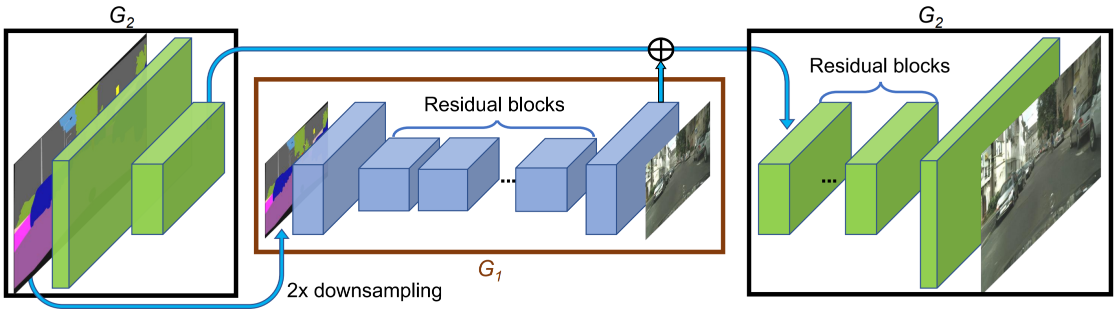


Figure 4.9. pix2pixHD generator structure from [143].

During training, G_1 is first trained on low-resolution images. G_2 is appended to the last layer of G_1 and trained jointly on high-resolution images. The discriminator structures of pix2pixHD are paired with multiscale generators for low-resolution and high-resolution images. The input of its discriminator is real images and synthetic images that are downsampled at multiple scales that match the pyramid structures. The high-resolution discriminator can encourage generating images with finer details. The low-resolution discriminator guides the generator to generate globally consistent images. pix2pixHD also uses an optional instance map as an auxiliary input. Each individual object inside the instance map has a unique ID. The instance map provides the object boundaries of the same class. However, for our application creating panicle segmentation masks for small objects is very time-consuming, which conflicts with our goal of saving time in ground truthing. We decide not to use instance maps for generating sorghum images.

4.4.2 SPADE

The regular normalization layers from the generator of pix2pixHD tend to lose semantic information. The diminished semantic information affects the quality of the synthetic images. Spatially-adaptive denormalization (SPADE) [144] preserves the semantic information compared to regular normalization by incorporating residual blocks with SPADE [144] layers into the up-sampling part of the generators as shown in Figure 4.10. The SPADE layer consists of a convolutional layer and two modulation layers as shown in Figure 4.10.

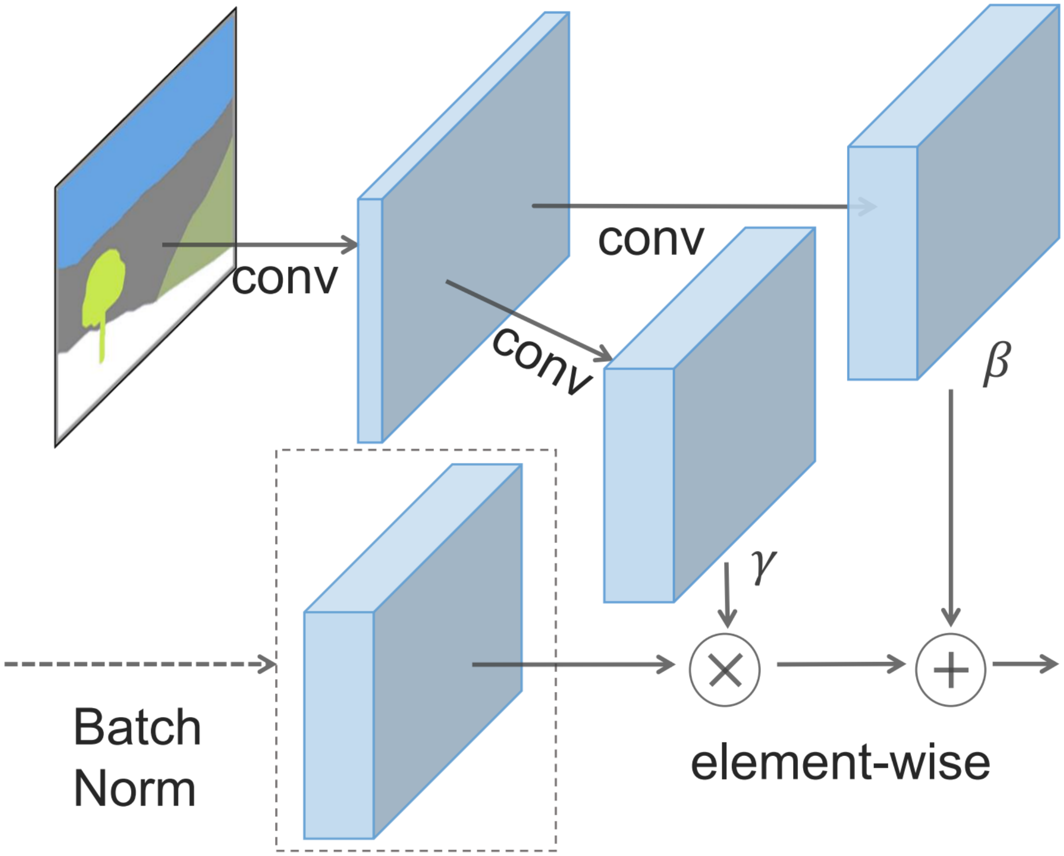


Figure 4.10. SPADE block design from [144].

The convolutional layer during training uses the ground truth label mask as input. Two modulation layers take the output of the convolutional layer and output two modulation parameters γ and β. The γ and β are multiplied and added to the normalized activation as output.

The generator of SPADE discards the encoder structures from pix2pixHD since the label maps are directly fed into the residual blocks within the generators. A new image encoder is used to encode the real image into random vectors before sending it into the up-sampling generator. The discriminator of SPADE follows the multi-scale discriminators of pix2pixHD with SPADE layers as normalization. The discriminator takes the concatenation of the input ground truth mask and the synthetic images and classifies whether the generated image is synthetic.

Compared to pix2pixHD, SPADE has better performance on public datasets such as ADE20K [145] and Cityscapes [146] due to the added SPADE blocks. All of the public datasets have multiple classes and well-defined segmentation masks that contain a large amount of semantic information. We only have one class in our dataset (panicles) with minimal semantic information so the added SPADE blocks for preserving semantic information might not be beneficial. We use both methods, pix2pixHD, and SPADE, and compare their performance for synthetic panicle image generation.

4.5 Experimental Results

4.5.1 Dataset

The real image dataset consists of a set of cropped images from 4 orthomosaics of sorghum field in acquired in 2020 as shown in Figure 4.11. The image is acquired by RGB cameras mounted on UAVs flying at an altitude of 20m. The spatial resolution of the orthomosaics is 0.25cm/pixel. There are 4 orthomosaics from multiple dates that represent different growing stages of sorghum. The resolution of the cropped image is 1024 × 1024 pixels. Each of the cropped images contains 4 row segments of sorghum plants due to the input image of the GANs has to be a square shape. Each panicle in the images is manually labeled with a bounding box. The bounding box labels are converted to the ground truth label maps for training the GANs. The label map is a single-channel grayscale image that has the same resolution as the image. The conversion is achieved by the following equation:

$$p_{map} = \begin{cases} 1, \ p_{map} \in R \\ 0, p_{map} \notin R \end{cases} \tag{4.7}$$

where p_{map} is the pixel in the label map. R is the region inside the bounding box. The pixel in the label map has a value of 1 if the pixel is inside the bounding box. The rest of the pixel has a value of 0. In total, we have 500 manually labeled, cropped images from 4 orthomosaics as our experimental dataset. We use 400 images for training for the GANS and

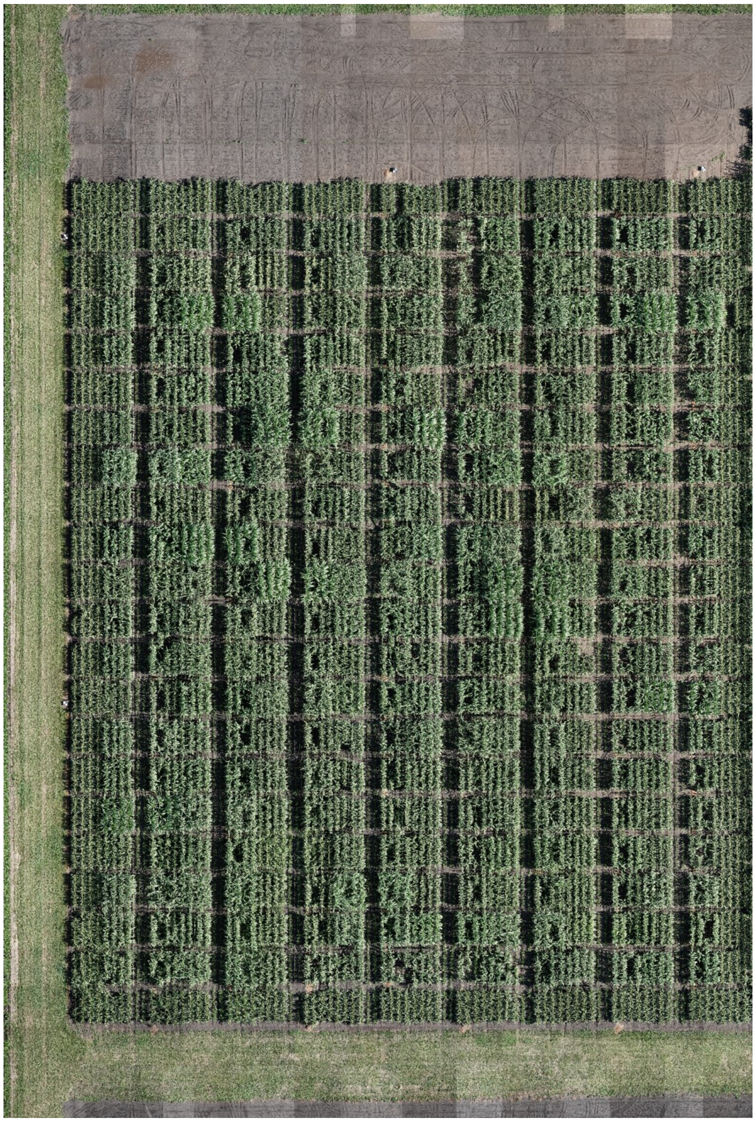


Figure 4.11. An orthomosaic of a sorghum field. The dataset used for the experiments is cropped from a set of orthomosaics. The cropped images do not contain the destructive sampling rows.

100 images for testing the panicle detection and counting system. The number of cropped images from each orthomosaic is evenly distributed.

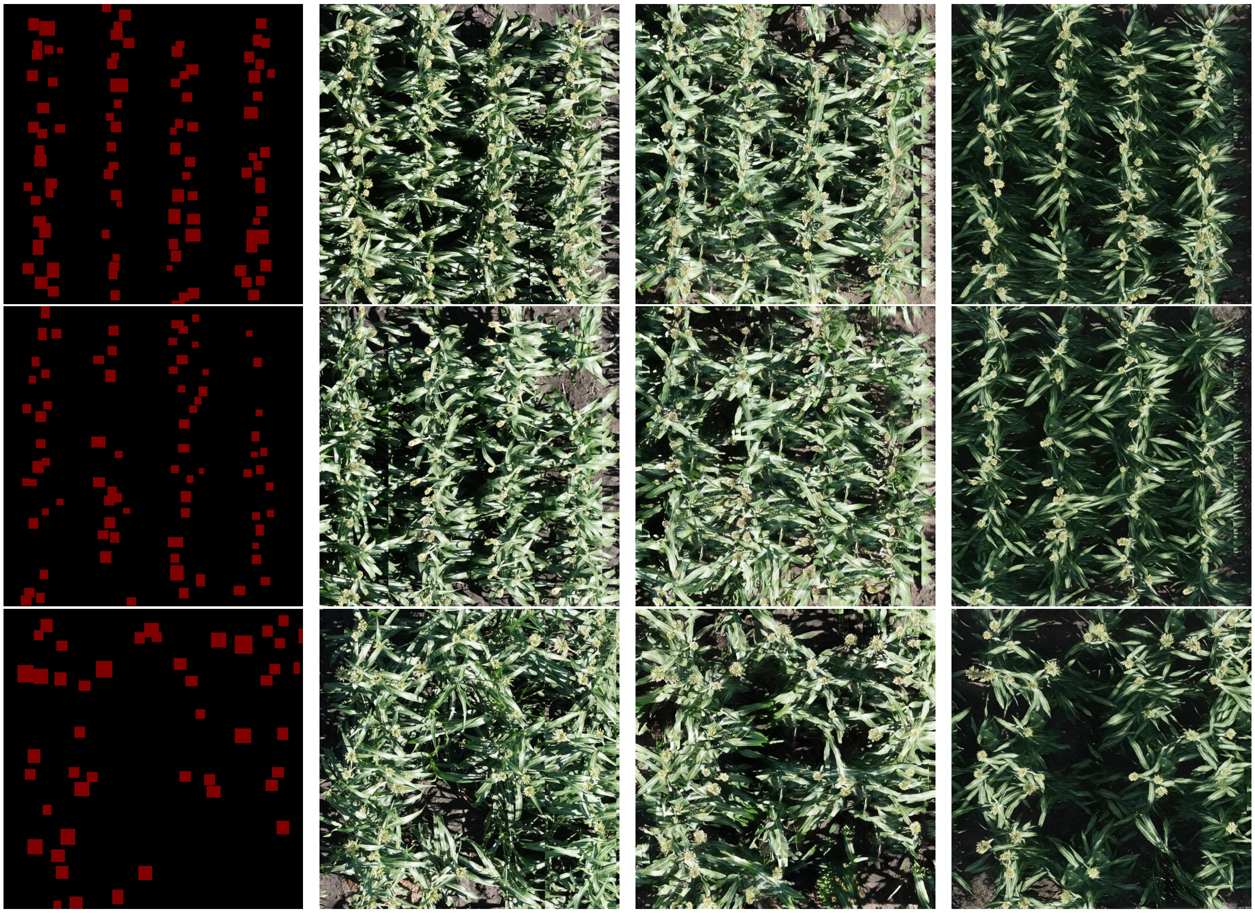
4.5.2 Synthetic Panicle Generation

We train pix2pixHD and SPADE with the training dataset described above. Both methods are trained on a single NVIDIA TITAN RTX with 24GB memory and 200 epochs in native resolution (1024 × 1024 pixels) without any resizing or cropping. Adam optimization [108] is used for both pix2pixHD and SPADE with an initial learning rate of 0.0002. The results are shown in Figure 4.12. The panicles generated by SPADE are more realistic than those generated by the pix2pixHD. The panicle edges from SPADE are clearer while the panicle edges of pix2pixHD are blurred. However, the pix2pixHD has a more natural background that resembles real sorghum images. The synthetic background, e.g. the highlights on leaves and the overall brightness of SPADE, is too dark. The straight line on the right side of the synthetic images is learned from the real images created by the process of orthorectification. Overall, the images generated by pix2pixHD are more realistic.

4.5.3 Panicle Detection

We generated 1000 synthetic images each from pix2pixHD and SPADE using random label maps (a total of 2000 GAN-generated images). Figure 4.13 shows some examples of synthetic images. The random label maps are constrained in the location and size of the bounding boxes such that the overall distribution is identical to real sorghum images in our dataset. The mean and variance of the panicle bounding box center and size are estimated for each row. This information is used during the random bounding box generation. A small overlap is allowed for each bounding box mask. In total, we have 1400 images for training our panicle detector from each of the GANs. This is 400 real ground truth images plus 1000 synthetic images for the particular GAN.

We use YOLOv5 [120] as our object detection network for panicles. We use the mean average precision (mAP) with Intersection over Union (IoU) from 0.5 to 0.95 (COCO mAP [118]) as the metric for detection, and Root Mean Squared Error (RMSE) [107], Mean



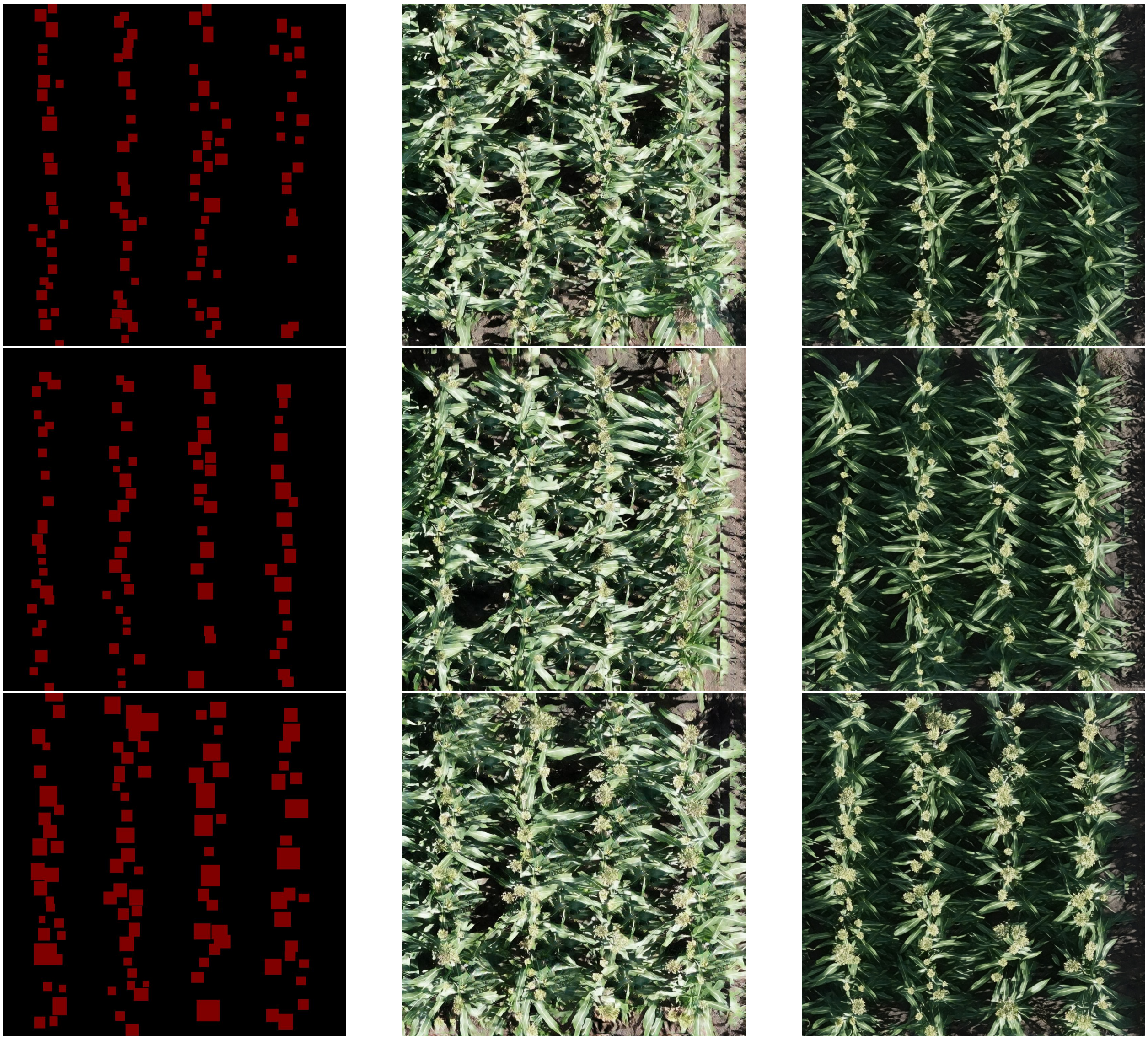
(a) Ground Truth Label map

(b) Real images

(c) pix2pixHD results

(d) SPADE results

Figure 4.12. Real panicle images and synthetic panicle images generated using pix2pixHD and SPADE.



(a) Random label map

(b) pix2pixHD results

(c) SPADE results

Figure 4.13. Synthetic panicle images generated using pix2pixHD and SPADE with random input labels. We adjusted the label generation parameters so the random labels have different shapes to simulate sorghum at multiple growing stages.

Absolute Error (MAE) [107] and Absolute Percent Error (MAPE) [107] for counting In our case, the mAP is equal to AP because we only have one class.

We evaluate the performance with the rest of the 100 real ground truth images not used for training. The results are shown in Table 4.1.

Table 4.1. Panicle detection and counting results. Each column represents a detection model trained on the corresponding dataset. "Real Image" is the dataset of 400 real images. "pix2pixHD" and "SPADE" are the datasets with 400 real images plus 1000 synthetic images from the corresponding GAN. Bold indicates the best performance.

Metric	**Real Image**	**pix2pixHD**	**SPADE**
mAP@[.5,.95]	72.4	78.9	**79**
MAPE	11.6	**7.2**	9.7
MAE	8.0	**4.6**	5.5
RMSE	9.6	**5.6**	6.5

Compared to training only on 400 real ground truth images, both GAN-based data augmentation methods achieve better mAP. For the counting metrics, the model trained on the augmented images (real plus synthetic) also performs better. Despite the different image styles generated by pix2pixHD and SPADE, their data augmentation performances are similar. The synthetic images of pix2pixHD have slightly better performance for the counting metrics due to the more realistic background. We found that for the pix2pixHD the weights tend to diminish around 200 epochs and the result becomes a completely black image. SPADE does not have this problem, possibly due to the added SPADE residual blocks in the generator. We feel both methods are suitable for GAN-based data augmentation for high-resolution UAV images.

5. SEMI-SUPERVISED OBJECT DETECTION FOR SORGHUM PANICLES

5.1 Introduction

From the previous chapter, we demonstrate using transfer learning to reduce the size of the training dataset for plant center localization. We also propose a data augmentation method to improve the performance of the sorghum panicle detection network. However, those methods still require at least a few hundred images as ground truth. For real applications, it is not feasible to annotate such a large quantity of images for each scenario. In this chapter, we investigate the approach to train a sorghum panicle detection deep neural network on a very small amount of RGB UAV images using semi-supervised learning.

5.2 Overview of Related Work

5.2.1 Semi-Supervised Classification

Semi-supervised classification approaches train the network with a small amount of labeled data and a large amount of unlabeled data to reduce the manual data labeling [147]–[150]. The use of pseudo-labels [151] is the key idea for semi-supervised approaches. The pseudo-labels are the data labels generated by the model pre-trained on the small dataset. The pseudo-labels are combined with the real labels to expand the training dataset. A semi-supervised loss is introduced for training on labeled and unlabeled data. Recent work focuses on regulating the loss function to maintain consistency during training. MixMatch [148] is an example of consistency regulation as shown in Figure 5.1.

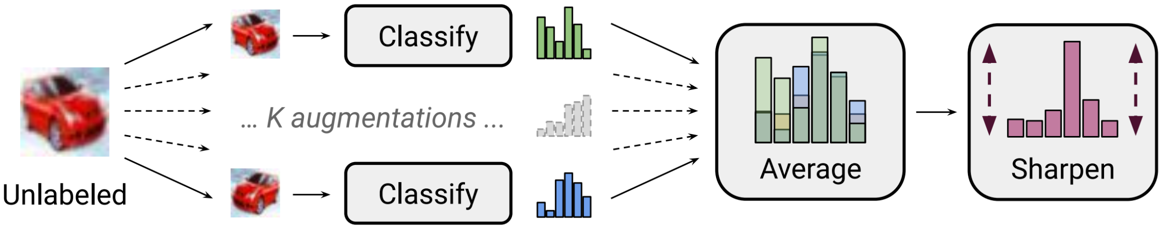


Figure 5.1. MixMatch process [148].

It uses data augmentation, label guessing, and MixUp on both labeled and unlabeled images. FixMatch [152] is another consistency-based method for semi-supervised classification as shown in Figure 5.2. It generates pseudo-labels from the prediction on unlabeled images. The pseudo-labels are used to train the model with an augmented version of the same image.

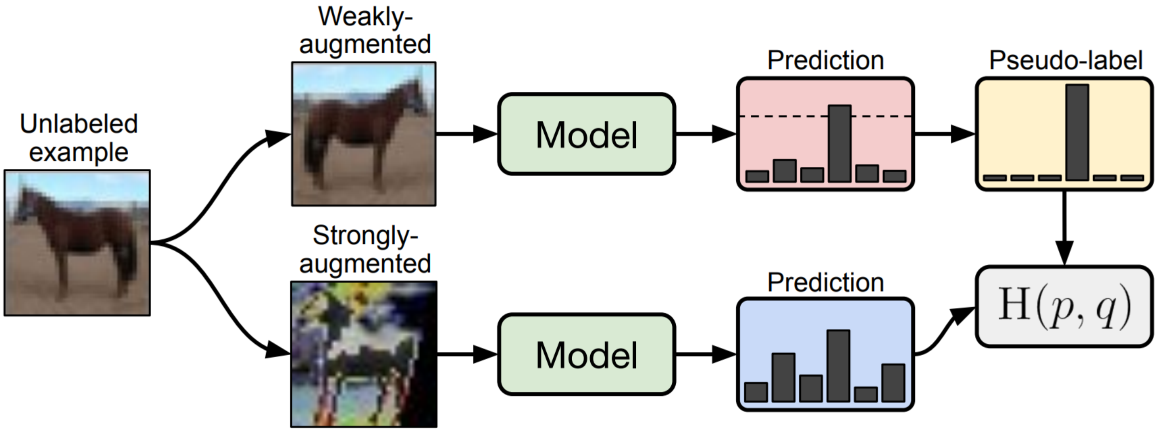


Figure 5.2. FixMatch process [152].

5.2.2 Semi-Supervised Object Detection

Similar to semi-supervised classification, pseudo-label-based approaches are used for semi-supervised object detection [151]. The approach consists of a teacher model and a student model. The teacher model is trained with a small amount of data at first. The teacher network then will generate the annotation from the unlabeled dataset to produce pseudo-labels for self-training. The pseudo-labeled data and labeled data are combined to train another neural network (student model). Semi-supervised object detection has recently shown progress in object detection tasks. In [153], Sohn *et al.* introduces a framework, STAC, to generate highly confident pseudo labels and update the models by enforcing consistency via strong augmentations. The overall framework of STAC is shown in Figure 5.3.

It generates pseudo-labels from unlabeled data using non-maximum suppression (NMS). The confidence-based method is used to filter pseudo-labels.

Unbiased Teacher [154] is another framework to jointly train the student and teacher networks in a mutually-beneficial manner as shown in Figure 5.4.

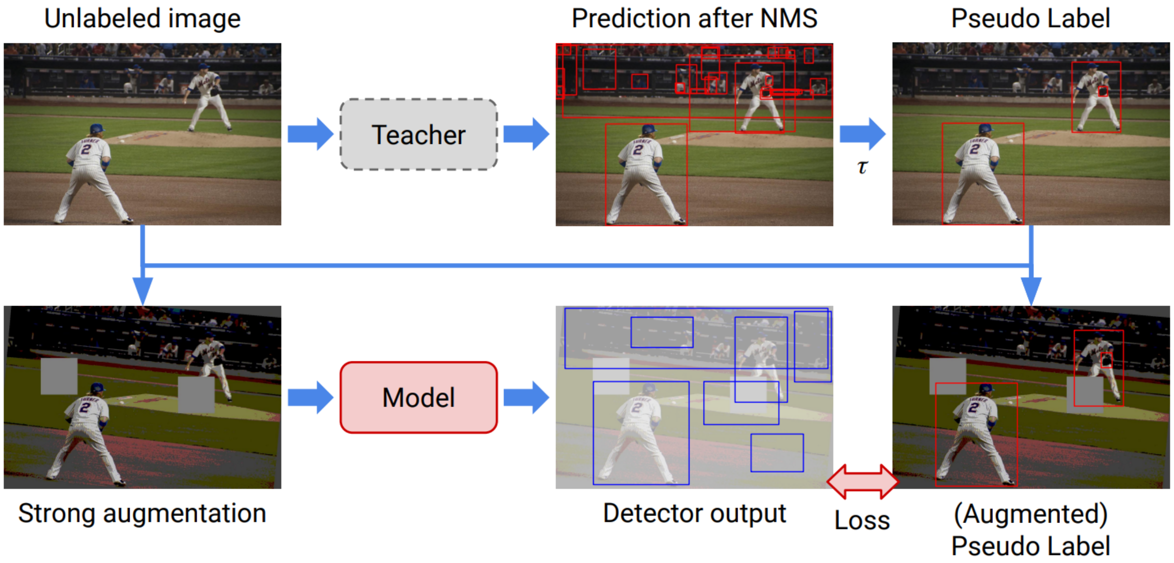


Figure 5.3. STAC Framework [153].

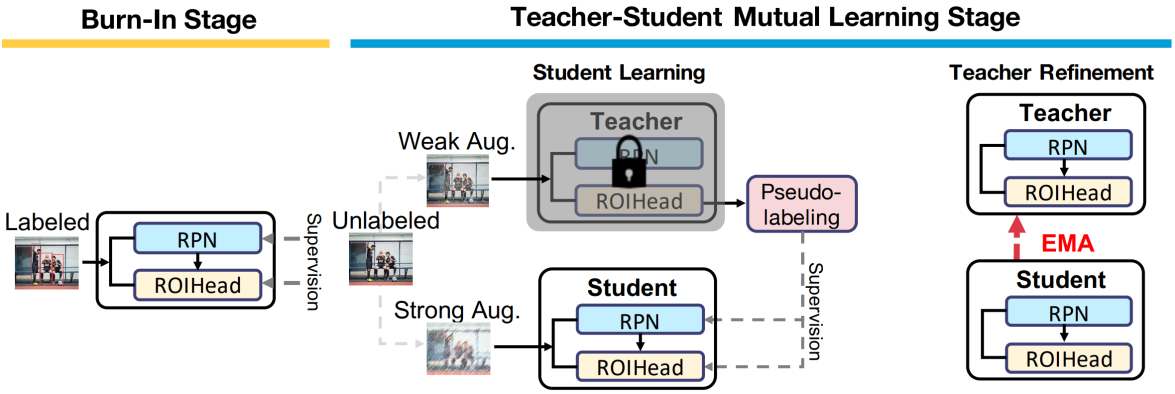


Figure 5.4. Unbiased Teacher Framework [154].

The teacher model is trained first with supervised only in the burn-in stage. After that, the teacher model will generate pseudo-labels to train the student model. The student model weight is transferred to the teacher model via exponential moving average (EMA).

5.3 Semi-Supervised Approach

We investigate semi-supervised learning for two-stage and one-stage object detection methods. For two-stage object detection, we use the Soft Teacher [155] framework with Faster-RCNN. For one-stage object detection, we choose the Efficient Teacher [156] framework with YOLOv5. The selection of the detection network is based on the performance of general detection datasets such as COCO [118]. Theoretically, both semi-supervised methods are interchangeable with the other object detection method. However, the performance is degraded if we simply apply one method to another due to the structure difference between one-stage networks and two-stage networks. In this case, we choose semi-supervised methods that have the best fit for each type of detection network as a fair comparison.

5.3.1 Soft Teacher

The Soft Teacher framework consists of a teacher model and a student model as shown in Figure 5.5. The teacher model is trained using a small batch of labeled data and performs pseudo-labeling on the unlabeled images. The student model is trained on both labeled and pseudo-labeled images. During the training process, the teacher model is continuously updated by the student model through the exponential moving average (EMA) strategy. The loss function of the soft teacher is a combined loss function from supervised and unsupervised loss:

$$\mathcal{L} = \mathcal{L}_s + \alpha \mathcal{L}_u \tag{5.1}$$

$$\mathcal{L}_s = \frac{1}{N_l}(\mathcal{L}_{cls}(I_l^{\mathrm{i}}) + \mathcal{L}_{reg}(I_l^{\mathrm{i}})) \tag{5.2}$$

$$\mathcal{L}_u = \frac{1}{N_u}(\mathcal{L}_{cls}(I_u^{\mathrm{i}}) + \mathcal{L}_{reg}(I_u^{\mathrm{i}})) \tag{5.3}$$

where $\mathcal{L}$ is the weighted sum of supervised loss $\mathcal{L}_s$ and unsupervised loss $\mathcal{L}_s$, α is the weight for unsupervised loss, $\mathcal{L}_{cls}$ is the classification loss, $\mathcal{L}_{reg}$ is box classification loss, I_u^{i} is the i-th unlabeled image, I_l^{i} is the i-th labeled image, N_u is the number of unlabeled image and N_l is the number of labeled image. During pseudo-labels generation, the framework uses NMS and FixMatch [152] strategy to remove duplicate bounding box candidates. The high threshold value is also used for pseudo-label generation to improve the quality of pseudo-labels. The process of pseudo-label generation will introduce an error as some foreground box candidates will be assigned as negatives. To compensate for this problem, the soft teacher framework introduces a loss function that uses more information from the teacher model:

$$\mathcal{L}_u^{cls} = \frac{1}{N_b^{fg}} \sum_{\mathrm{i}=1}^{N_b^{fg}} l_{cls}(b_{\mathrm{i}}^{fg}, \mathcal{G}_{cls}) + \sum_{\mathrm{j}=1}^{N_b^{bg}} w_{\mathrm{j}} l_{cls}(b_{\mathrm{j}}^{bg}, \mathcal{G}_{cls}) \tag{5.4}$$

$$w_{\mathrm{j}} = \frac{r_{\mathrm{j}}}{\sum_{k=1}^{N_b^{bg}} r_k} \tag{5.5}$$

where b_{i}^{fg} are foreground boxes, b_{j}^{bg} are background boxes, $\mathcal{G}_{cls}$ is the set of pseudo boxes, l_{cls} is the classification loss, r_k is the reliability score.

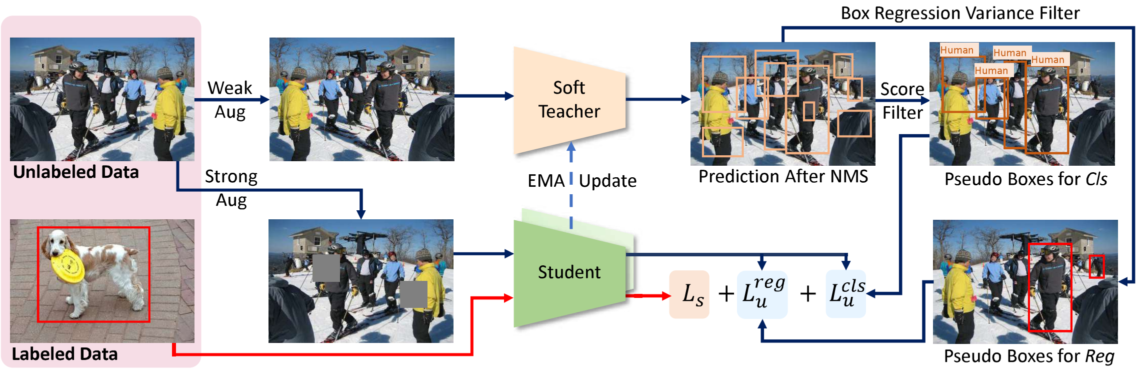


Figure 5.5. Soft Teacher semi-supervised framework from [123].

The original method from Soft Teacher is training the teacher model and student model at the same time with random weights at the beginning. In practice, we found the training is unstable due to the limited number of images in our dataset. We introduce another warm-up stage for the teacher model. During the warm-up stage, the teacher model will be trained only with labeled data. The trained weight will then be loaded into the co-training stage with the student model.

5.3.2 Efficient Teacher

One-stage object detection networks [117], [120], [157] generally have higher recall and faster training speed compared to two-stage object detection networks [99]. However, the semi-supervised learning approach from two-stage detection networks is facing challenges when directly applied to a one-stage detection network. The multi-anchor strategy used in the one-stage network magnifies the label imbalance problem from semi-supervised learning in the two-stage network, resulting in low-quality pseudo-labels and poor training results. The efficient teacher framework is a semi-supervised learning approach optimized for single-stage object detection networks. The overview of the framework is shown in Figure 5.6. To leverage the label inconsistency problem, the efficient teacher framework introduces a novel pseudo-label assigner to prevent interference from low-quality pseudo-labels. During training, each pseudo-label is assigned a pseudo-label score that represents the uncertainty of the label. Two threshold value of the score τ_1 and τ_2 is used. If a pseudo-label has a score between τ_1 and τ_2, the pseudo-label is categorized as an uncertain label. The loss of uncertain labels is filtered out to improve the performance. The efficient teacher framework uses an unlabeled data loss:

$$\mathcal{L}_u = \mathcal{L}_u^{cls} + \mathcal{L}_u^{reg} + \mathcal{L}_u^{obj} \tag{5.6}$$

where $\mathcal{L}_u^{cls}$ is the class loss, $\mathcal{L}_u^{reg}$ is the regression loss and $\mathcal{L}_u^{obj}$ is the objectness loss. For pseudo-label with high classification scores, only $\mathcal{L}_u^{obj}$ is used. For the rest of the pseudo-labels, $\mathcal{L}_u^{reg}$ is used when the objectness score is higher than 0.99.

The efficient teacher framework also introduces an epoch adaptor mechanism to stabilize and accelerate the training process. Epoch adaptor combines domain adaptation and distribution adaptation techniques. The domain adaptation has the following loss function:

$$\mathcal{L}_{da} = -\sum_{h,w}[Dlog(p_{h,w}) + (1-D)log(1-p_{h,w})] \tag{5.7}$$

where $p_{h,w}$ is the domain classifier output, $D = 0, 1$ for labeled and unlabeled data, respectively. Domain adaptation enables training on unlabeled and labeled data during the first Burn-In phase to prevent overfitting on labeled data. The distribution adaptation technique dynamically updates the thresholds τ_1 and τ_2 at each epoch to reduce overfitting.

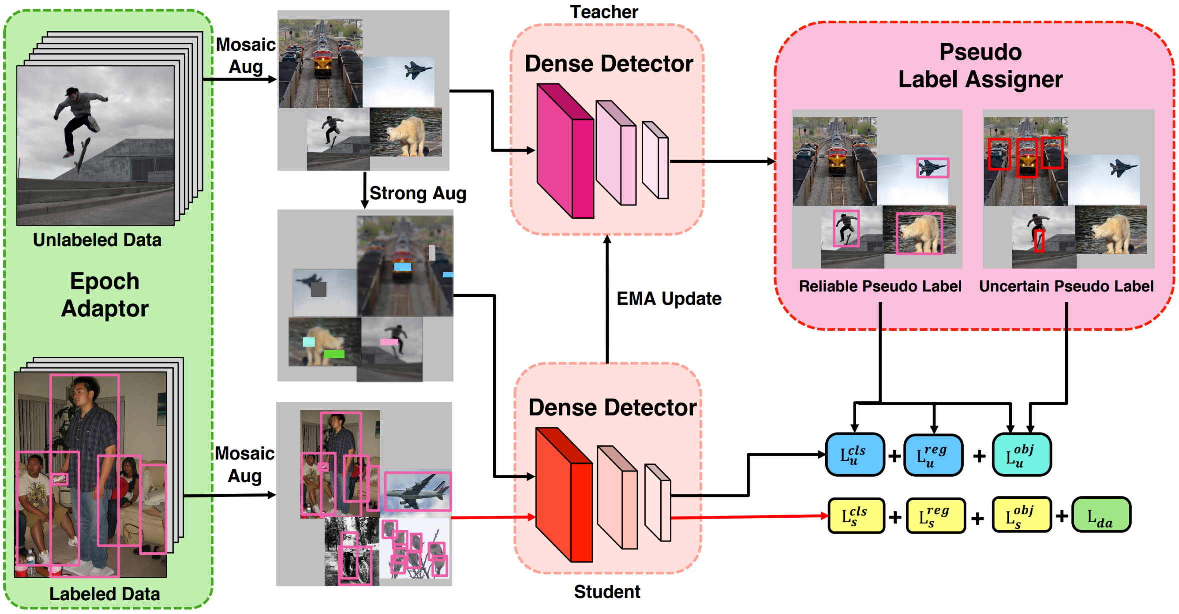


Figure 5.6. Efficient Teacher semi-supervised framework from [156].

5.4 Experiment and Results

5.4.1 Dataset

The sorghum panicle dataset is from an RGB orthomosaic [24] captured by UAVs in a sorghum field. We select a small region of the orthomosaic and crop it into small images for data labeling and training purposes as shown in Figure 5.7. We select the early sorghum growing stage for the experiments. Compared to the later stage, the early-stage sorghum panicles have more variation in shapes and sizes which brings more challenge to the methods.

Moreover, the fewer panicles in each image can further reduce the number of available labels for training. In total, we have 364 images for training, 90 images for validation, and 60 images for testing. Each image is resized to 640 ×640 resolution during training. These RGB images are used to form a supervised baseline to compare with semi-supervised learning. For semi-supervised learning, we randomly select 1%, 5%, and 10% within the training dataset to form a semi-supervised learning dataset. The labels of the rest of the training data are removed correspondingly to represent the unlabeled data. We have 3 labeled images for the 1% dataset, 18 labeled images for the 5% dataset, and 36 labeled images for the 10% dataset.

5.4.2 Parameter Tuning

Setting the appropriate training parameters is very important for semi-supervised learning on the limited dataset. The learning rate and NMS threshold for pseudo-labels have the most impact on training performance. In the supervised learning stage, the learning rate can have a greater learning rate for fast convergence. In the semi-supervised learning stage, the learning rate needed to be adjusted for a very small amount of labeled data. In practice, we found setting the learning rate of 0.001 for the supervised stage is good for warm-up the teacher model. The default semi-supervised learning rate from both methods is too large resulting in unstable training. We found to set the learning rate to 0.00005 is suitable for the semi-supervised stage in both methods. For the NMS threshold in the efficient teacher framework, we set the confidence threshold to 0.5 to reduce the false positive and the IoU threshold to 0.1 to reduce the duplicated bounding box.

5.4.3 Results

We evaluate the semi-supervised learning approach by using three different settings of the original training dataset: 1%, 5%, and 10% training data. In the warm-up stage, we first trained the network with only 1%, 5%, and 10% labeled data in a supervised manner to form a baseline. In the semi-supervised stage, the weights of the baseline model are loaded into the teacher model. The teacher model with pre-loaded weight is trained with the student model together. For the soft teacher framework, we use Faster-RCNN [99] with ResNet-50

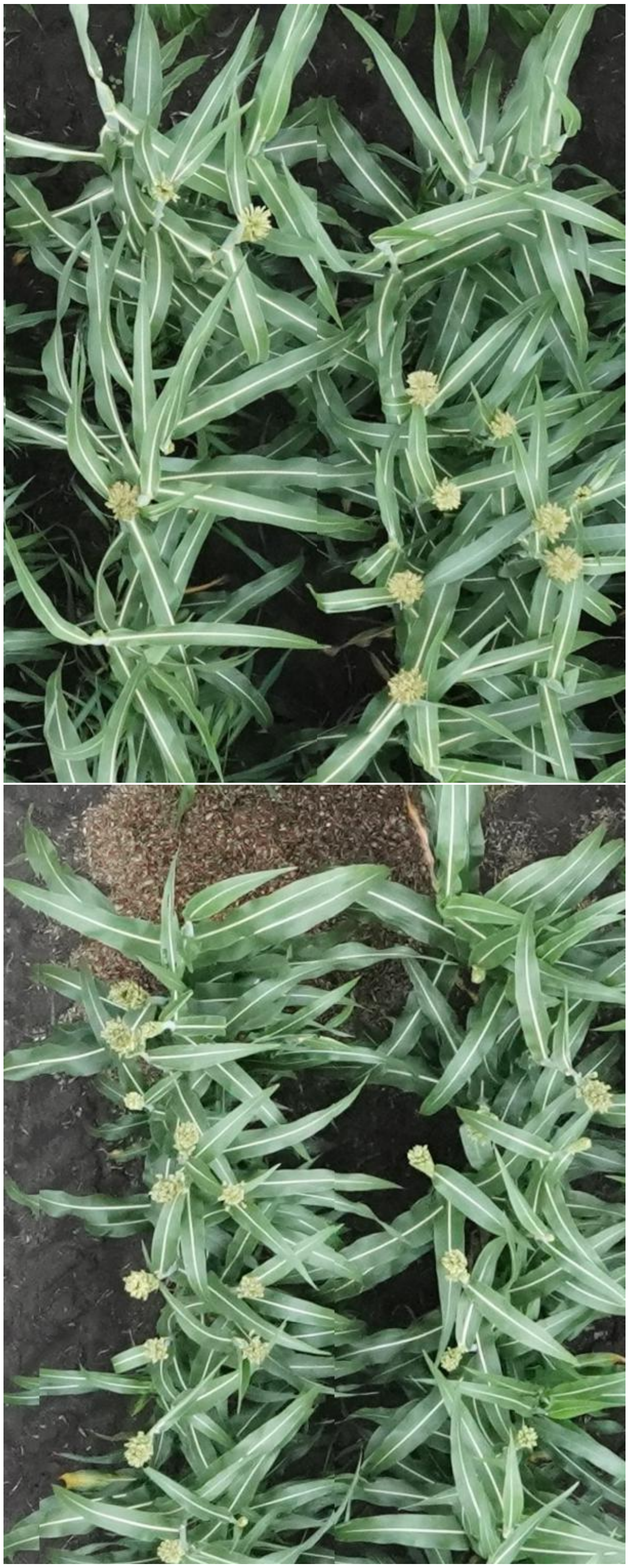


Figure 5.7. Sample images from training dataset for semi-supervised learning.

[72] backbone. For the efficient teacher framework, we use the YOLOv5l model. The result of the soft teacher method is shown in Figure 5.8, the semi-supervised learning increases the mAP by 5.6% in 1% labeled data, 2.5% in 5% labeled data and 3.7% in 10% labeled data. The result of the efficient teacher method is shown in Figure 5.9, the semi-supervised learning increases the mAP by 3.1% in 1% labeled data, 1.7% in 5% labeled data and 1.7% in 10% labeled data. The efficient teacher method achieves the highest mAP due to the better YOLOv5 model in the baseline. However, the soft teacher model has the highest mAP increases in three scenarios. The training is done on a single NVIDIA RTX A40 GPU. The soft teacher took 7 hours to finish training while the efficient teacher only took one hour to finish. Compared to supervised learning using fully labeled data (364 images), we can achieve comparable results with only 10% of the original amount (36 images). The method can be further expanded to other plant trait detection tasks to reduce the amount of time for data labeling.

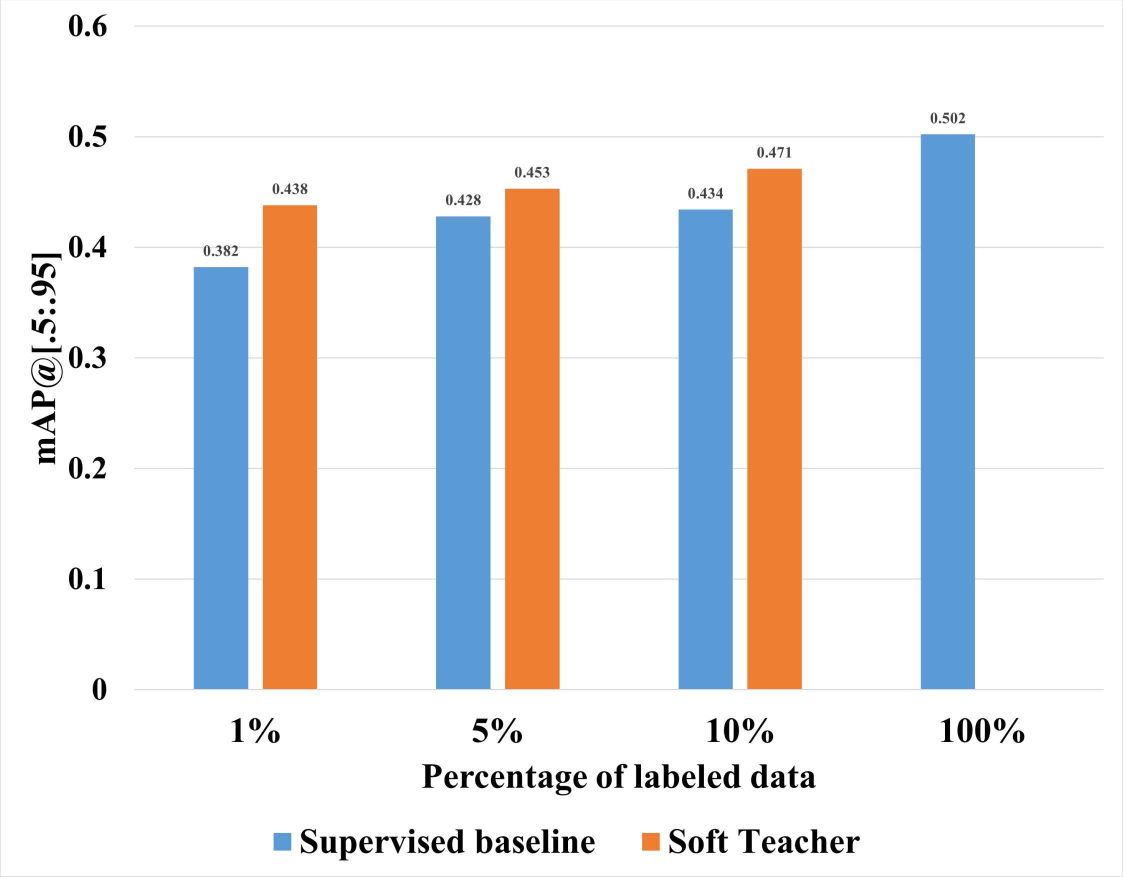


Figure 5.8. Result from the soft teacher.

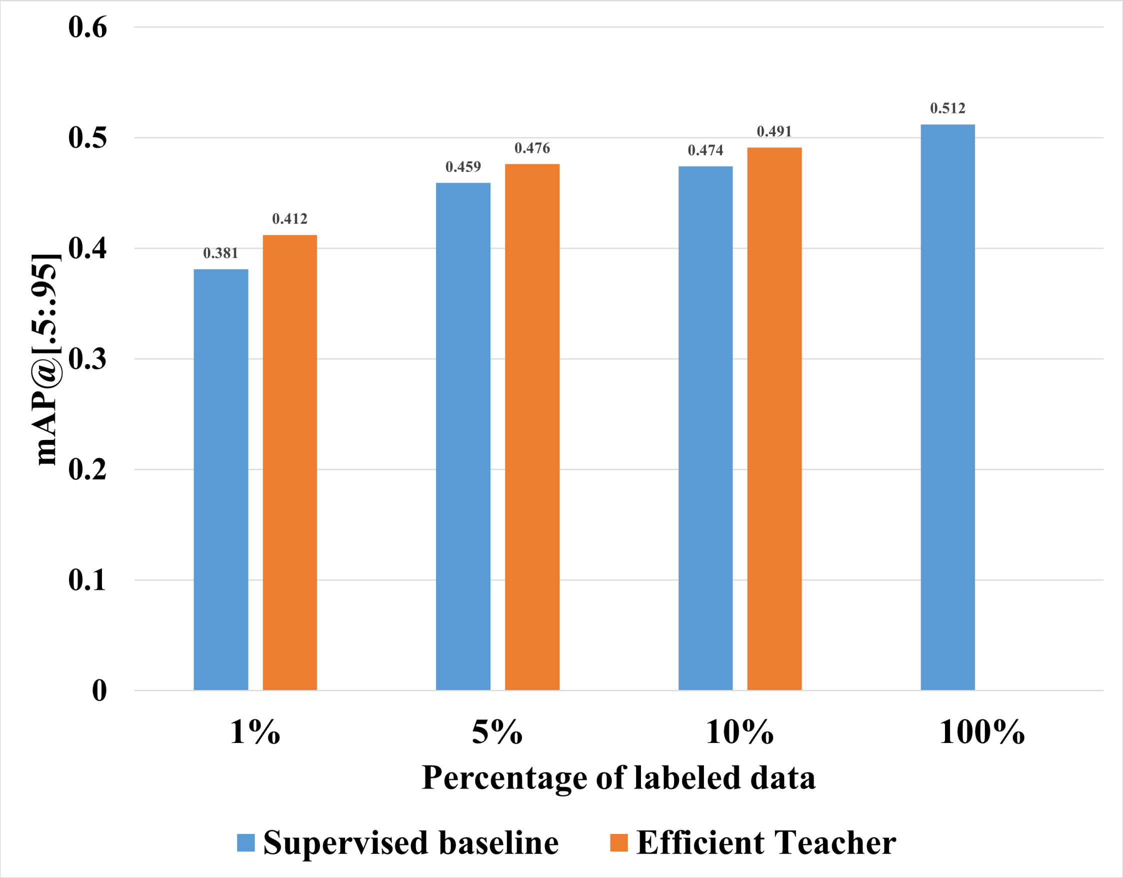


Figure 5.9. Result from the efficient teacher.

6. SYNTHETIC FIELD IMAGE GENERATION WITH STABLE DIFFUSION

6.1 Introduction

In the previous chapter, we introduce plant image synthesis using GANs. The biggest problem in training GANs is stability. We often observe that the output becomes a completely black image after a few hundred steps. In recent work, diffusion models show promising results for synthetic image generation. Compared to GANs, it does not require a large amount of training data and can transfer the text input into image output. Moreover, the training process is much more stable than GANs. In this chapter, we investigate generating crop images using stable diffusion.

6.2 Overview of Related Work

6.2.1 Diffusion Probabilistic Models

The diffusion probabilistic models [158] use a Markov chain with a diffusion process to convert one distribution into another. Given a data distribution $q(x)$, the forward diffusion process has the following equations:

$$\boldsymbol{\pi}(\mathbf{y}) = \int d\mathbf{y}^{'} T_{\pi}(\mathbf{y} \mid \mathbf{y}^{'}; \beta)\boldsymbol{\pi}(\mathbf{y}^{'}) \tag{6.1}$$

$$q(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t-1)}) = T_{\pi}(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t-1)}; \beta_t) \tag{6.2}$$

where $\boldsymbol{\pi}(\mathbf{y})$ is the distribution from repeated diffusion process, $T_{\pi}(\mathbf{y} \mid \mathbf{y}^{'}; \beta)$ is the diffusion kernel, β is the diffusion rate and $q(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t-1)})$ is the data distribution of the kernel with $\mathbf{x}$ data point at step t. The forward trajectory can be described as:

$$q(\mathbf{x}^{(0 \cdots T)}) = q(\mathbf{x}^{(0)}) \prod_{t=1}^{T} q(\mathbf{x}^{t} \mid \mathbf{x}^{t-1}) \tag{6.3}$$

where T is the total step for the diffusion process. For example, $q(\mathbf{x}^t \mid \mathbf{x}^{t-1})$ can represented as a Gaussian diffusion process. From the forward diffusion process, we can derive the reverse process:

$$p(\mathbf{x}^T) = \boldsymbol{\pi}(\mathbf{x}^T) \tag{6.4}$$

$$p(\mathbf{x}^{(0\cdots T)}) = p(\mathbf{x}^{(T)}) \prod_{t=1}^{T} p(\mathbf{x}^{(t-1)} \mid \mathbf{x}^t) \tag{6.5}$$

where $p(\mathbf{x}^{(t-1)} \mid \mathbf{x}^t)$ is the reverse distribution. The diffusion model can be combined with the U-Net backbone to generate synthetic images by reversing the diffusion process. Figure 6.1 shows an example of the reverse diffusion process from noise to image.

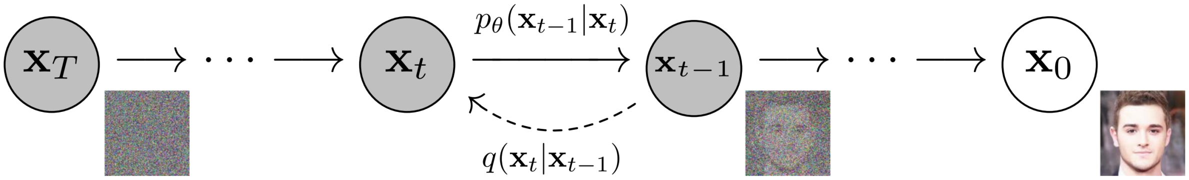

$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_{t-1}$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$\mathbf{x}_0$

Figure 6.1. The denoising reverse diffusion process from [159].

6.2.2 Transformer

The transformer [160] structure is an encoder-decoder architecture that uses self-attention mechanism. The overall structure is shown in Figure 6.2.

Given an input $x_1, \cdots, xn$, the encoder of the transformer converts it to a sequence $\mathbf{z} = (z_1, \cdots, z_n)$. The decoder of the transformer converts $\mathbf{z}$ to the output sequence $(y_1, \cdots, y_m)$. The attention mechanism is the mapping from a query and key-value pairs to an output as shown in Figure 6.3. In the scaled dot-product attention part, the dot-product is used for queries, keys, and values as the following:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{6.6}$$

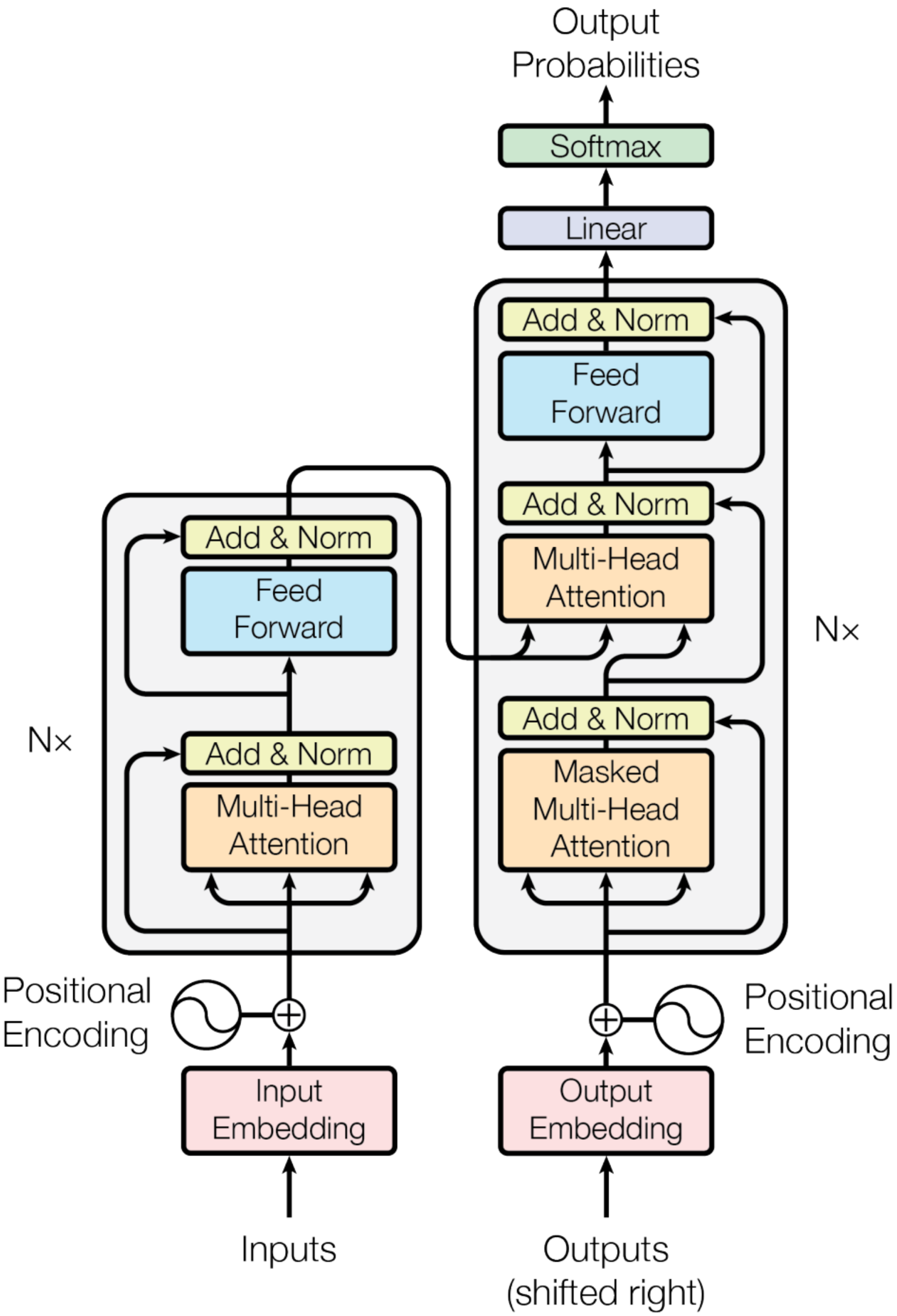


Figure 6.2. The transformer model architecture from [160].

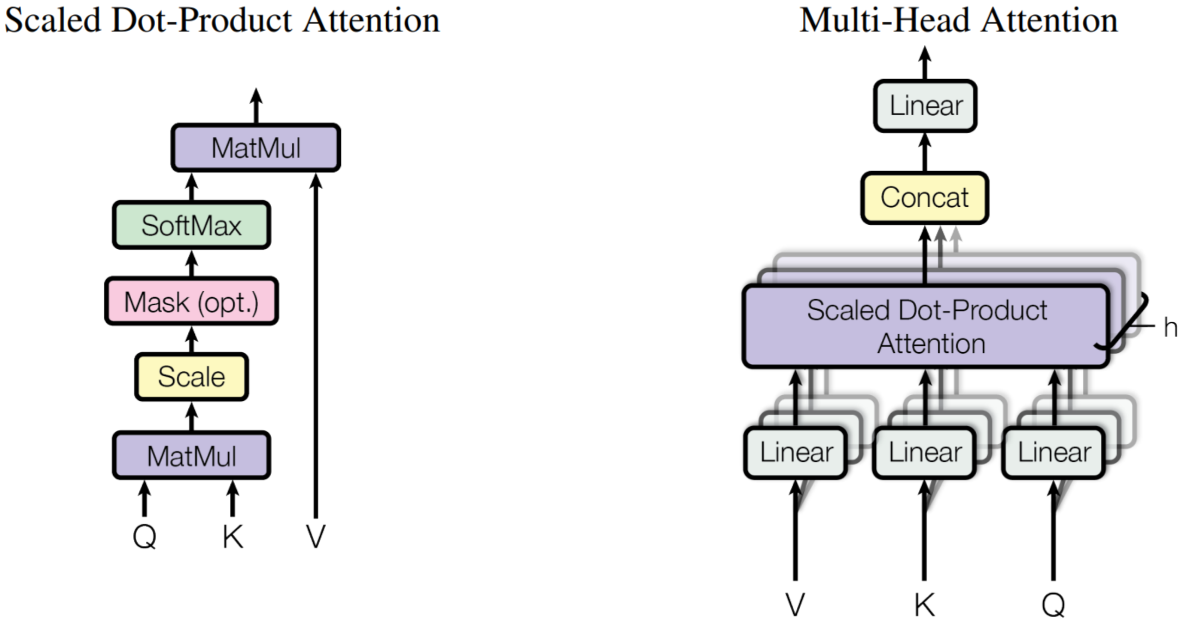


Figure 6.3. The attention function from [160].

where $\sqrt{d_k}$ is the key dimension, and Q, K, V are query, key, and value, respectively. The multi-head attention structure first projects the query, key, and value and applies the attention mechanism in parallel with concatenation:

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head}_1, \cdots, \mathrm{head}_h)W^O \tag{6.7}$$

$$\mathrm{head}_\mathrm{i} = \mathrm{Attention}(QW_\mathrm{i}^Q, KW_\mathrm{i}^K, VW_\mathrm{i}^V) \tag{6.8}$$

where W_i^Q, W_i^K, W_i^V and W^O are parameter matrices.

6.3 Stable Diffusion

The stable diffusion [161] is a latent diffusion model that generates latent representations with conditional inputs as shown in Figure 6.4.

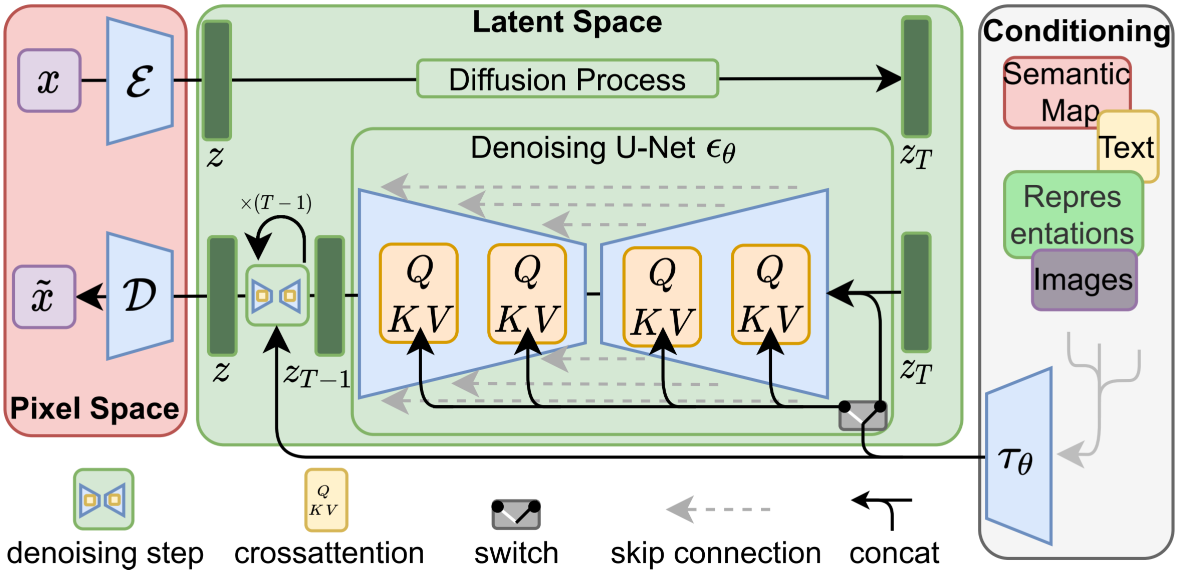


Figure 6.4. The Stable Diffusion architecture from [161].

Before feeding into the diffusion model, the input image is compressed to latent representations from an autoencoder $\mathcal{E}$. The generated latent representation is converted back to real images using autodecoder $\mathcal{D}$. The diffusion model only receives images in the latent space to reduce the computational cost. It uses denoising U-Net ϵ_0 as the backbone network for image synthesis. The U-Net architecture is modified with cross-attention mechanism [160] for guided image generation. The conditional input y (text, map, image) is projected by encoder τ_0 and mapped to the inner layers of the U-Net using cross-attention.

The training process of stable diffusion starts from a set of images with different levels of noise (steps) in latent space. The training data consists of the images and the embedded noise steps. The denoising U-Net is trained to predict the noise given training images with its step. During inference, a noise image and its noise step are fed into the denoising U-Net to predict the noise. The predicted noise is subtracted from the noise image to form a less noisy image. The image with less noise will continue to feed into U-Net repeatedly to subtract the noise as a Markov chain process.

6.4 Results

We use stable diffusion to generate synthetic sorghum and maize images. The training images are from a single orthomosaic. The orthomosaic is cropped into a square image with three rows of plants as shown in Figure 6.5. Each cropped image is resized to 512 ×512 resolution with text data. The resolution is aligned with the maximum resolution in stable diffusion. In total, there are 400 sorghum images and 400 maize images. The date of the orthomosaic is the later season where the sorghum panicles and maize tassels are shown in the images.

The stable diffusion v1-4 [161] pre-trained model is used for fine-tuning. The total training process is done on 8 NVIDIA A40 GPUs with 15000 training steps. The learning rate is set to 0.00001 for fine-tuning purposes. Results are shown in Figure 6.6 and Figure 6.7. From the results, the images from diffusion models do not look as real as the images from GANs. Compared to real images, the images from stable diffusion have blurry textures and colors similar to paint.

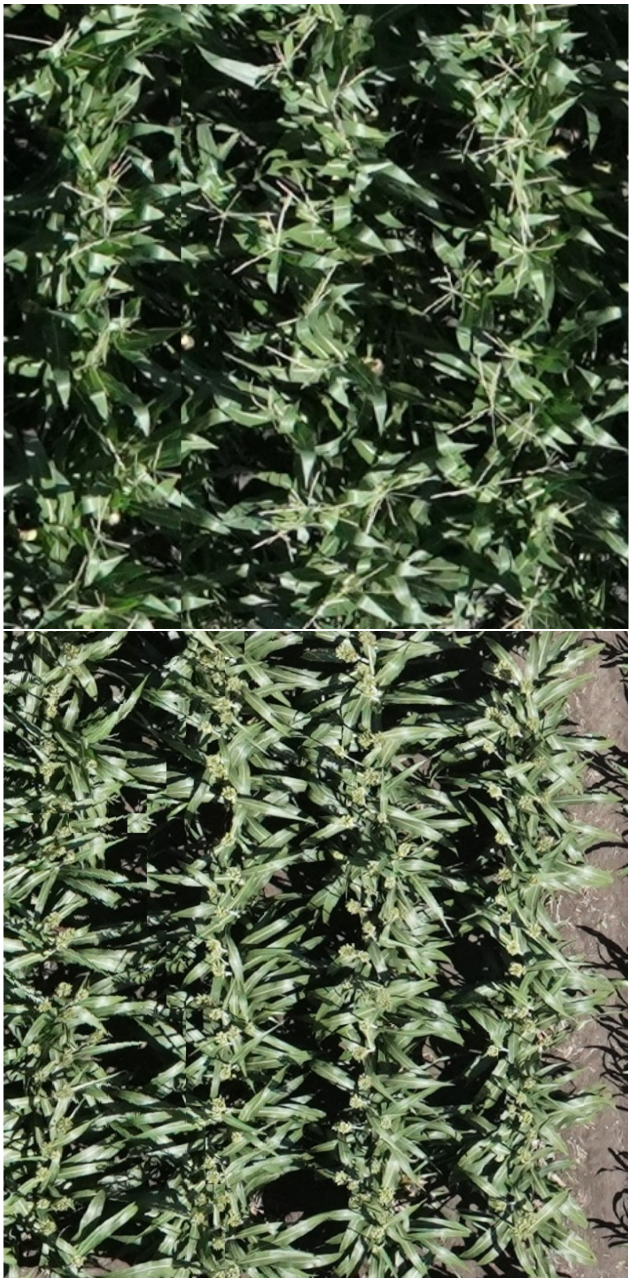
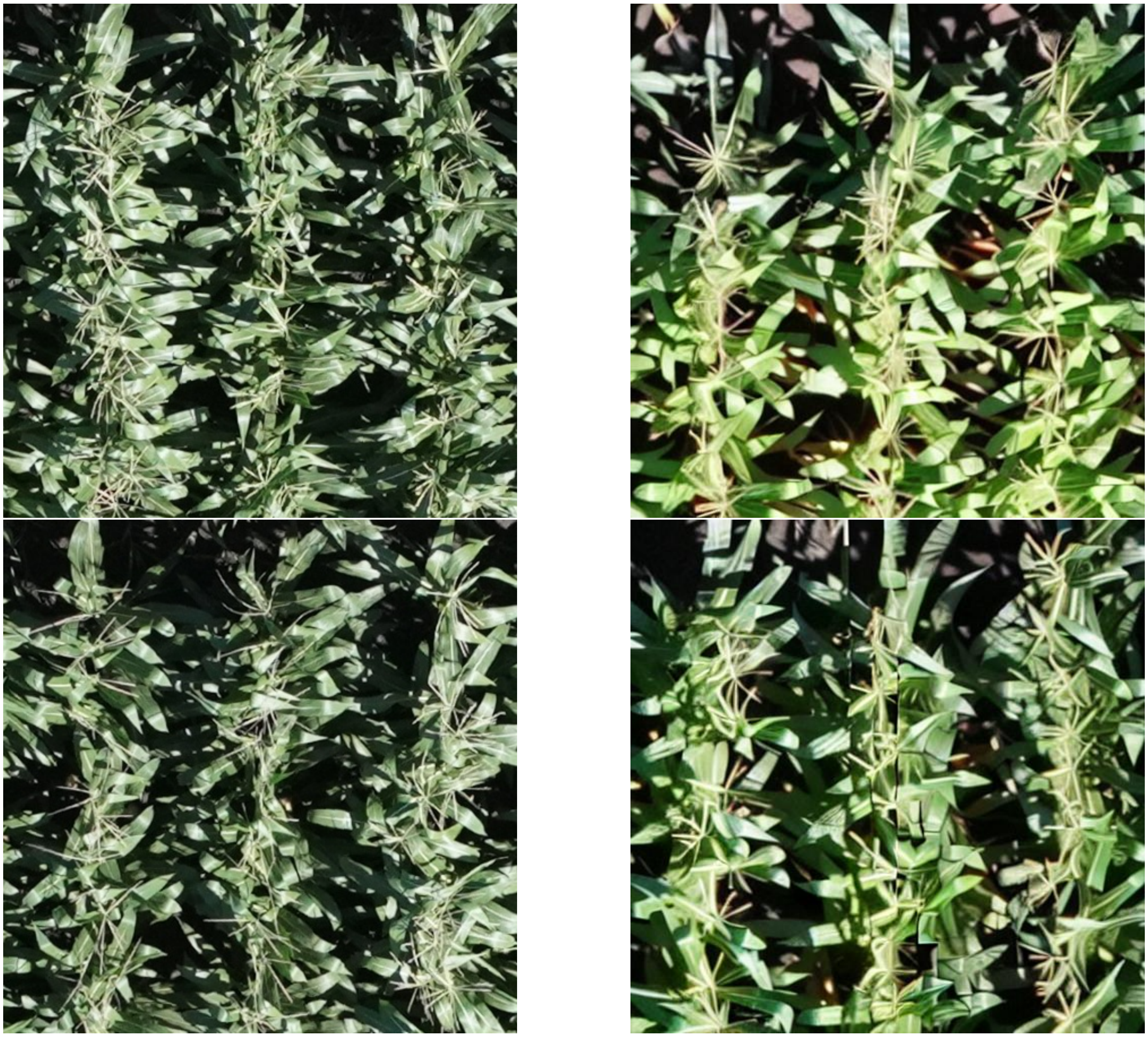


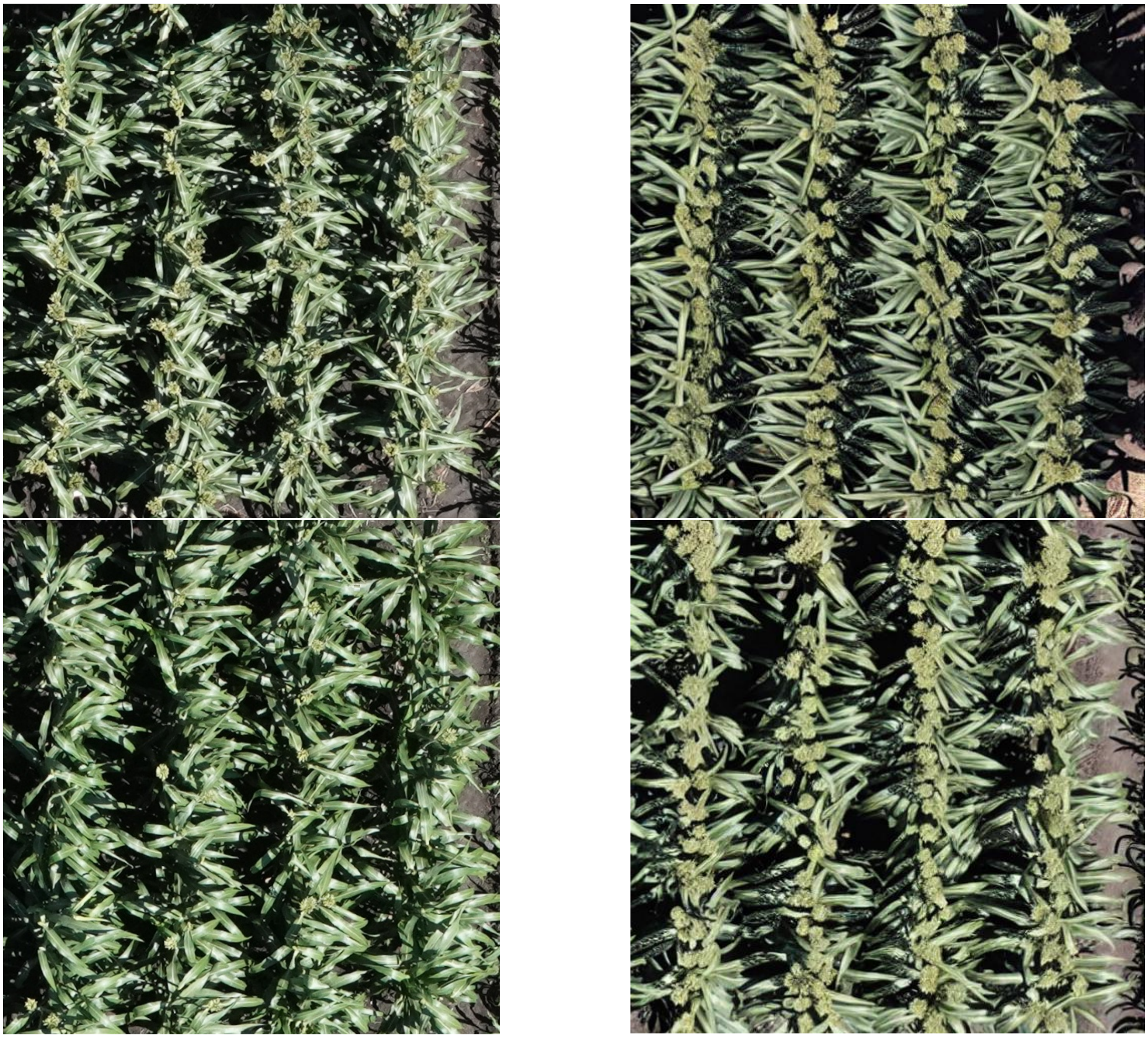
Figure 6.5. Sample images from training dataset for stable diffusion.



(a) Real maize images

(b) Synthetic maize images

Figure 6.6. Real maize images and synthetic maize images generated from stable diffusion.



(a) Real sorghum images

(b) Synthetic sorghum images

Figure 6.7. Real sorghum images and synthetic sorghum images generated from stable diffusion.

7. TOMATO WILTING CLASSIFICATION

7.1 Overview of Methods

Our method for tomato wilting classification is shown in Figure 7.1. We use convolutional neural networks for tomato wilting classification. The input images are preprocessed before sending to CNN for classification. The preprocess has three parts: color calibration, plant segmentation, and plant cropping. The preprocessed images feed into the neural network for classification. The classification results are the wilting stages in terms of number.

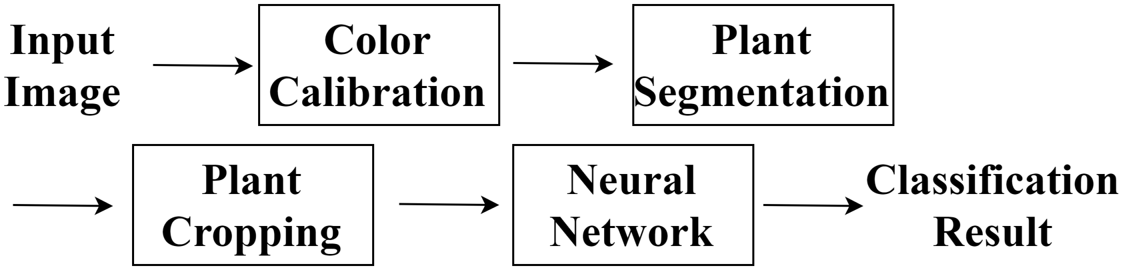


Figure 7.1. Block diagram of the proposed method

7.2 Related Work

Machine learning and deep learning provide promising results in image-based tomato disease detection. In [162], a CNN-based approach is used to detect tomato leaf disease. In [163], an attention module is added to the CNN structure to improve the performance of tomato leaf disease detection. In [164], a SqueezeNet [165] model is used to detect tomato plant disease by leaf images. Most image-based tomato disease detection methods are based on the leaf images of tomato plants. None of those methods use the image of entire tomato plants. This is partially due to the limitation of public datasets. The public datasets only have tomato leaf images without genetic information. In our method, we use the entire tomato images with specific tomato varieties. Thus, we focus more on the entire structures of tomato plants.

7.3 Plant Segmentation

7.3.1 Color Calibration

The goal of color calibration is to make plant segmentation consistent in varied lighting conditions and to ensure that the images used throughout the process are consistent in terms of pixel values. A color checkerboard is used for color calibration. In order to extract the color patches from each image, we first convert the image to HSV color space. Then, we threshold the image from S and V color channels in HSV color space to form a mask image for color checkerboard pattern extraction as shown in Figure 7.3. The mask image is obtained as:

$$P_{mask} = \begin{cases} 1, & C_{HSV} \geq T \\ 0, & C_{HSV} < T \end{cases} \tag{7.1}$$

where P_{mask} is the pixel value on the mask. C is the H, S, or V value. T is the threshold value. We use the Harris corner detector [166] to find the checkerboard corner and extract the RGB values from each color patch as shown in Figure 7.4. We measure the XYZ values from each section of the color checkerboard using a spectrometer as a color reference to the current image. The XYZ values from the spectrometer are converted to RGB for color correction.

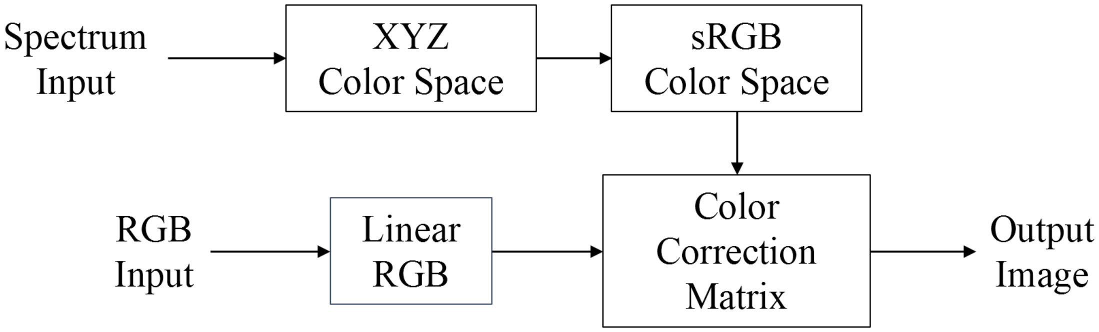
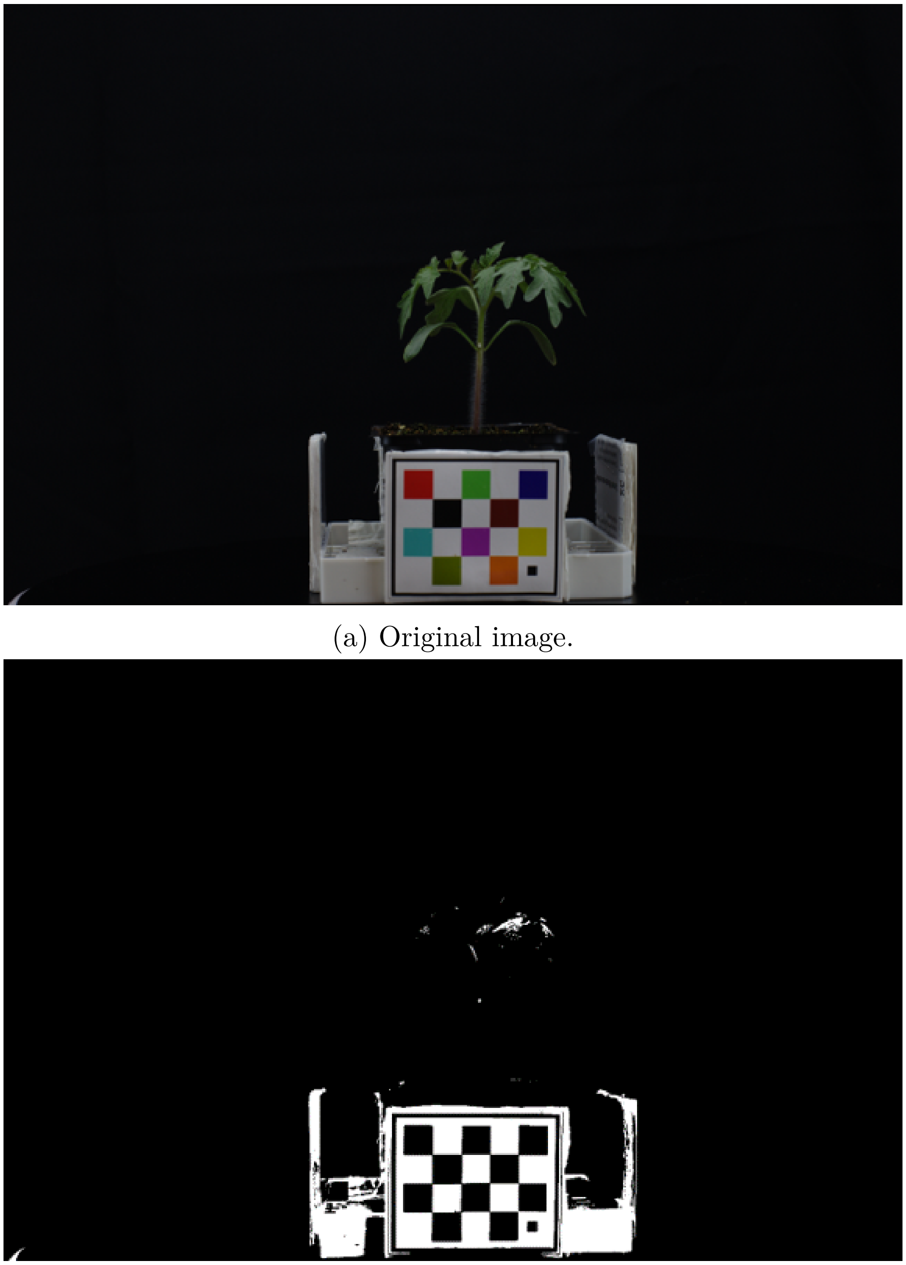


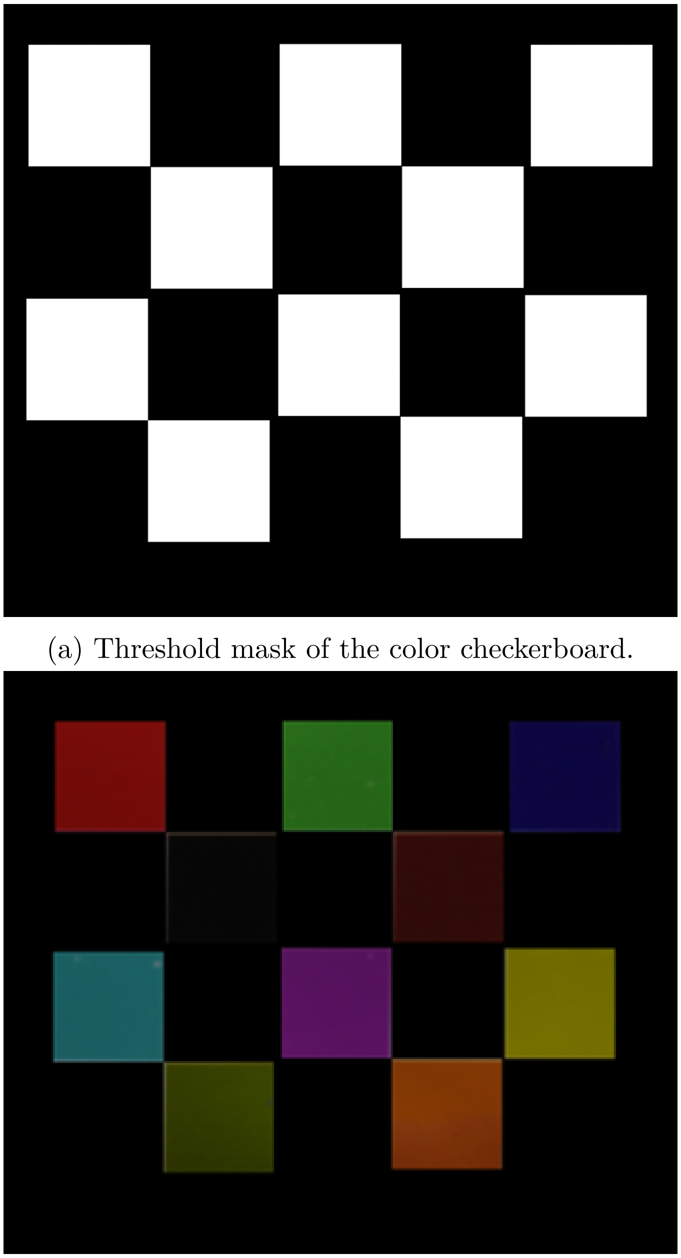
Figure 7.2. Block diagram of color correction method.



(a) Original image.

(b) Threshold image.

Figure 7.3. Output from the threshold.



(a) Threshold mask of the color checkerboard.

(b) Extracted color patches.

Figure 7.4. Color patch extraction.

The color correction diagram is shown in Figure 7.2. We use the measured RGB values and RGB values on the original image to perform color correction. We use the color correction matrix M to correct the color on RGB plant images.

$$M = P'(P^T P)^{-1} P^T, \tag{7.2}$$

where

$$P' = \begin{bmatrix} R_{ref} \\ G_{ref} \\ B_{ref} \end{bmatrix}, P = \begin{bmatrix} R_{tar} \\ G_{tar} \\ B_{tar} \end{bmatrix} \tag{7.3}$$

P' is the reference color matrix consisting of RGB values of each color patch from the XYZ values. P is the target color matrix consisting of RGB values of each color patch from the original images.

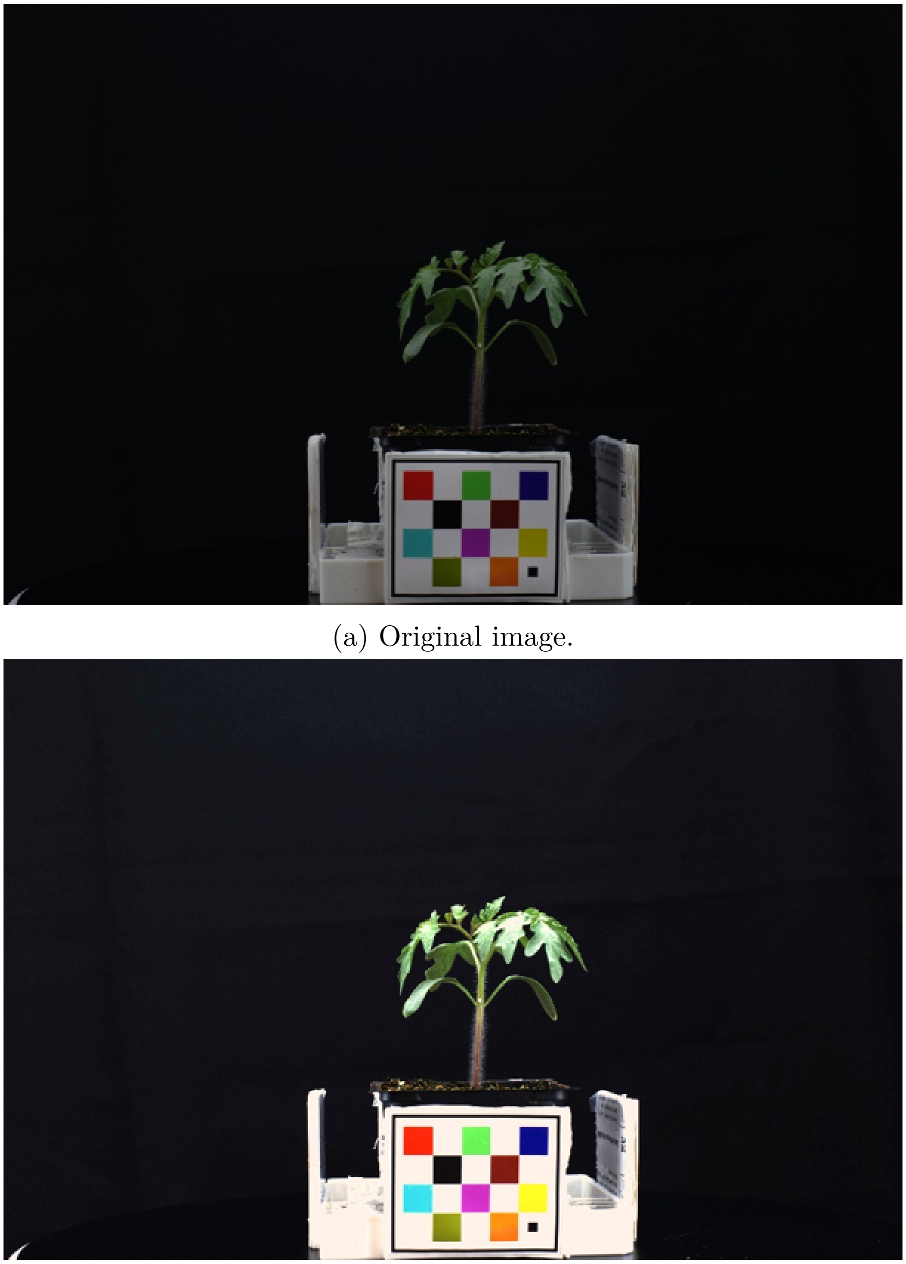
The original image is in standard RGB (sRGB) [167] color space. Because the original image is gamma-corrected, we convert the RGB values on the original images to linear RGB. We use the following equations for linear RGB conversion [167]:

$$C_{\text{linear}} = \begin{cases} \frac{C_{srgb}}{12.92}, & C_{srgb} \leq 0.04045 \\ (\frac{C_{srgb}+0.055}{1.055})^{2.4}, & C_{srgb} \geq 0.04045 \end{cases} \tag{7.4}$$

where C is the R, G, or B value. After estimating the matrix, we apply the correction matrix to the original image to acquire the color-corrected image.

7.3.2 Plant Segmentation

We segment the plants from the background for training the wilting detection network. By removing the excessive background, the neural network can focus more on classifying the plant itself. We use the color-corrected images as a reference and apply the threshold method to extract plants from the background. The color-corrected image is converted to HSV and CIELAB color space. We set a threshold for the V channel from HSV, a* channel, and b* channel from CIELAB. The threshold results are binary masks for each channel. A



(a) Original image.

(b) Color corrected image.

Figure 7.5. Color correction.

bitwise OR operation is applied among those masks to acquire a binary mask for the plant. After extracting the plants, we crop the image to remove the black margin.

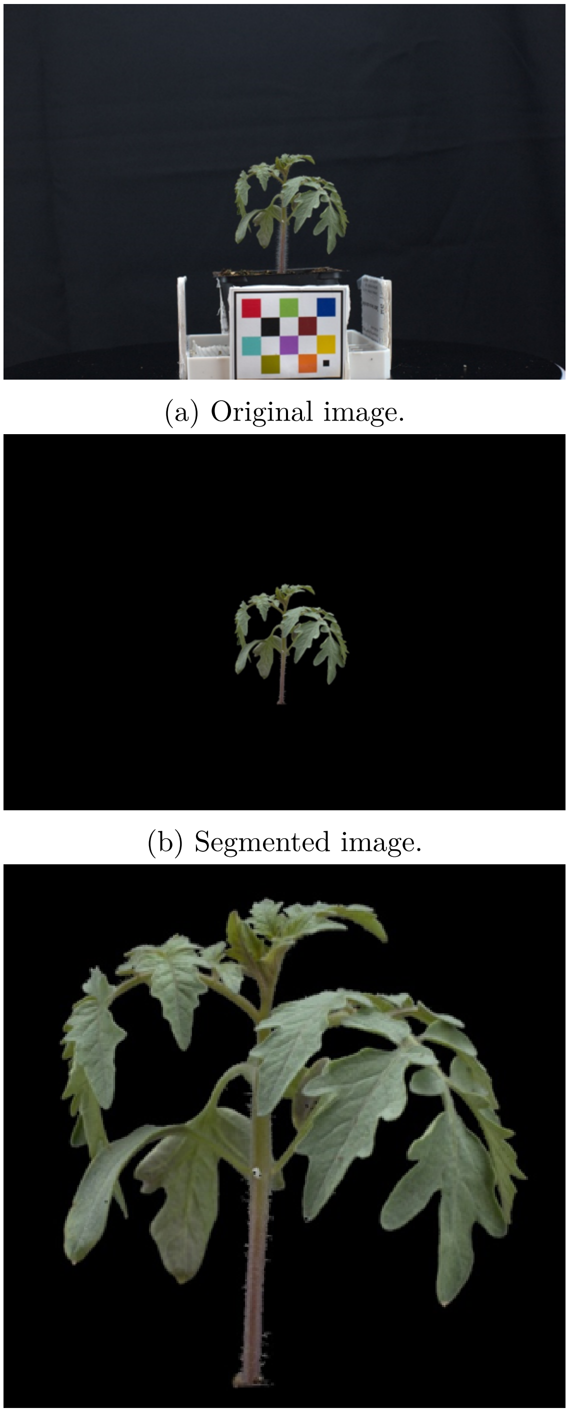
7.4 Wilting Classification Network

7.4.1 Training and Testing Datasets

The wilting tomato dataset contains 96 different tomato plants from three genotypes. Most of the plants start to wilt after the second day of treatment with bacteria so we only use the images from day 3 to day 5. Each plant image is captured horizontally on a rotating platform. Images are captured in 4 main directions (front, back, left and right) and the directions between each main direction so there are 8 images for one plant. In total, we use 1400 images for training and testing neural networks. The 20% of total images are split into the testing dataset. Each plant in the dataset has a wilting score from 0% to 100% to represent the wilting status of the plant. 0% means the plant is unwilted (healthy). 100% means the plant is totally wilted (died). We define plants that have a wilting score greater than 0% as wilted. We separate wilted plants into different wilted stages. For example, if 4 wilted stages are defined, the plants that have wilting scores between 1% and 25% will belong to stage 1, the plants that have wilting scores between 26% and 50% will belong to stage 2, the plants that have wilting scores between 51% and 75% will belong to stage 3, the rest of them will belong to stage 4.

7.4.2 Network Architecture

We use ResNet [72] in Figure 7.8 and VGG-16 [71] in Figure 7.7 as our wilting classification network architectures. For the specific ResNet models, we choose ResNet-18, ResNet-34, ResNet-50, and ResNet-101. Because of the size limitation of the dataset, transfer learning is used for training the dataset. The transfer learning models are pre-trained with ImageNet [92].



(a) Original image.

(b) Segmented image.

(c) Cropped image.

Figure 7.6. Plant segmentation.

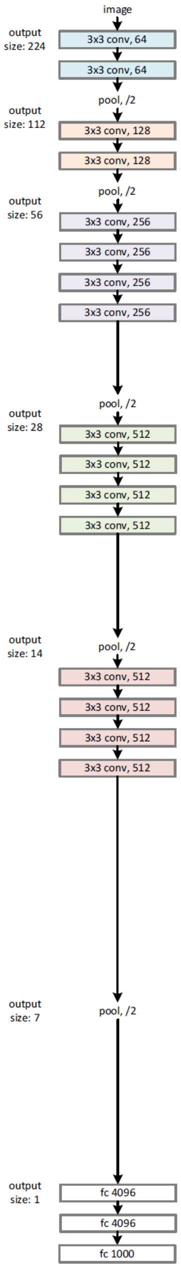


Figure 7.7. VGG-19 structure from [71]

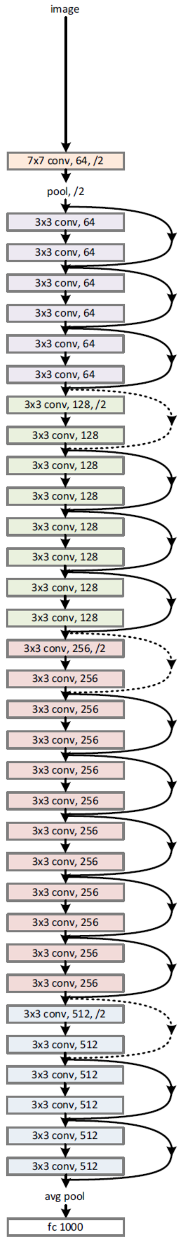


Figure 7.8. ResNet-34 structure from [72].

7.5 Experimental Result

7.5.1 Wilting Stage Classification

We compare the transfer learning method with the regular training method on the VGG-16 structure. We split the wilted plants into two stages for training and testing. We use accuracy as the metric for wilting classification.

$$ACC = 100\% \frac{S}{T} \tag{7.5}$$

T is the number of testing samples. S is the number of corrected outputs in testing samples T. The transfer learning method has better accuracy because it is pre-trained on large datasets. The weight of low-level feature extractors on the pre-trained network is highly optimized. Our plant datasets are very small compared to ImageNet. Therefore, we decide to use the transfer learning method for the rest of the experiments.

Table 7.1. Transfer learning and training accuracy.

Stages	Transfer Learning	Training
Healthy	80%	68%
Stage 1	42%	14%
Stage 2	78%	85%

We also test the effect on accuracy for the higher number of wilting stages. We compare the results of 4 wilting stage setups and the results of 2 wilting stage setups. The network performs better on fewer stages. As the number of stages increases, the boundary between each stage becomes unclear and the number of images in each stage is unbalanced due to the size of the datasets. We decide to use two stages for wilting classification.

Table 7.2. Wilted plants with 4 stages.

Stages	VGG
Healthy	92.5%
Stage 1	33%
Stage 2	0%
Stage 3	33%
Stage 4	100%

Table 7.3. Wilted plants with 2 stages.

Stages	VGG
Healthy	80%
Stage 1	42%
Stage 2	78%

We also test the difference between ResNet and VGG. The ResNet-18 has better results compared to VGG-16.

Table 7.4. VGG and ResNet accuracy.

Stages	ResNet	VGG
Healthy	70%	80%
Stage 1	78%	42%
Stage 2	82%	78%

7.5.2 Early Wilting Detection

We use the images from day 1 to day 2 as the training sets for early wilting detection. Although the plants are given 0% wilting scores, we label them based on the wilting scores on day five. All plants with wilting scores greater than 0% are labeled as inoculated. The rest of the plants are labeled as mocked. The ResNet-18, ResNet-34, ResNet-50, and ResNet-101 are the structures we used to train the early wilting detection network. Among those networks, ResNet-34, ResNet-50, and ResNet-101 have better performance than ResNet-18. ResNet-50 and ResNet-101 have the same performance during testing. ResNet-34 has slightly better results in mocked plants compared to ResNet-50 and ResNet-101.

Table 7.5. Early wilting detection result.

Type	ResNet-18	ResNet-34	ResNet-50	ResNet-101
Mocked	90%	85%	95%	95%
Inoculated	90%	95%	90%	90%

8. SUMMARY AND FUTURE WORK

8.1 Summary

In this thesis, we develop methods for plant center localization, sorghum flowering time estimation, and tomato wilting classification. The main contributions of this work are:

- We present a method for estimating plant centers for two types of crops and dates with a limited quantity of training data using a transfer learning approach. The method uses point annotation as training labels. We use a modified U-Net structure with the weighted Hausdorff distance as the network architecture for transfer learning. We slightly retrained a network pre-trained on sorghum to locate plant centers on maize crops with limited training data. We show that with proper pre-trained networks, transfer learning can improve the overall performance of the network with scarce training data. We also demonstrate that performing transfer learning with a pre-trained network is not effective if the distribution of the source domain is significantly different from the target domain.

- We propose a method for sorghum panicle detection and counting. The method uses bounding boxes as training labels. We select three different object detection networks for panicle detection and counting. Among those networks, YOLOv5 has the best performance in the evaluation. We demonstrate the counting results from the panicle detection method are suitable for flowering time estimation.

- We propose a method for flowering time estimation by counting panicles in UAV images. We describe the use of multi-temporal panicle counting for flowering time estimation. We use the polynomial regression on the multi-temporal counting data to estimate the sorghum flowering time. We test the flowering time estimation method on all the plots in a panel. Our result shows the estimated flowering times are very close to the results of manual counting.

- We presented the use of GAN-generated synthetic images to augment the training data for panicle detection and counting. We propose a method for generating high-resolution synthetic Sorghum UAV RGB images with bounding box labels using two image-to-image translation GANs. We examined two image-to-image translation GANs and showed that their use can improve the performance of panicle detection and counting.

- We demonstrate the capability of semi-supervised learning methods for reducing the amount of training data in sorghum panicle detection task. We examined two different types of semi-supervised learning approaches for sorghum panicle detection.

- We investigate crop image generation using diffusion models. We generate synthetic sorghum and crop images using a stable diffusion model with fine-tuning.

- We develop a method to segment the tomato plants from the background. We use the threshold method in HSV color space to segment the tomato plants. We apply color correction to tomato images to assist the tomato segmentation. The segmentation results can be used for tomato wilting classification.

- We present a method for tomato wilting classification. We train multiple CNN structures with the segmented tomato images for wilting classification. We compare the transfer learning method with the regular training method on CNN. We test the effect on accuracy for a higher number of wilting stages. We present a new approach to detect early plant wilting by labeling future wilting status on current plants.

8.2 Future Work

- Plant Location

  We use saliency-map-based methods for plant centers. The saliency-map-based method only works with point annotation. The available network structures are limited with point annotation. The saliency-map-based method is also sensitive to the ratio and the size of the image. The modified U-Net resizes the input image to 256x256, which is not suitable for high-resolution images. We can replace the methods with modern object-detection-based network architectures by converting the point annotation to bounding boxes. This will improve the accuracy and robustness of the system.

- Flowering Time Estimation

  The flowering time estimation method is an indirect method. It counts the number of panicles to approximate the time when 50% of panicle emerges. It does not detect the panicle with 50% blooming due to image resolution limits. There exists a bias due to the difference between panicle emerging time and flowering time. We could do more research on the relationship between the panicle emerging time and the flowering time to improve the estimation.

- Improving Sorghum Panicle Detection

  We did not use the temporal information available in our real Sorghum UAV dataset during training due to the limitation of the network structures. Future work includes developing multi-temporal methods that can generate synthetic plant images in a temporally consistent style. This will also us to estimate phenotypic traits as the plant grows. We will also examine our approach for estimating traits of other plants such as maize tassels.

- Semi-supervised Object Detection for Sorghum Panicles The semi-supervised network is strongly dependent on hyperparameter tuning to achieve the best performance. However, tuning the parameters itself is still time-consuming. We can develop an auto-tuning method on hyper-parameters. The method can also be extended to other plant traits.

- Synthetic Field Image Generation with Stable Diffusion The images from stable diffusion cannot be directly used for training due to the lack of ground truth information. We can replace the text encoding with panicle location encoding and guide the diffusion model to generate panicles based on the given location.

- Tomato wilting Classification

  The tomato wilting classification method receives the entire tomato image as input. Identifying wilting stages for the entire image of tomatoes is ambiguous even with human eyes due to the unclear boundaries, especially with a higher number of wilting stages. Classifying the wilting stages on plant sub-structures like stems and leaves might give more accurate results. We can develop a method that splits and classifies each part of tomato plants and generates results based on the relationship between each result.

8.3 Publications Resulting From This Thesis

1. **E. Cai**, J. Guo, C. Yang, and E. J. Delp, "Semi-Supervised Object Detection for Sorghum Panicles in UAV Imagery", *Proceedings of the IEEE International Symposium on Geoscience and Remote Sensing*, July 2023, Pasadena, CA.

2. **E. Cai**, Z. Luo, S. Baireddy, J. Guo, C. Yang, and E. J. Delp, "High-Resolution UAV Image Generation for Sorghum Panicle Detection", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), the 3rd International Workshop and Prize Challenge on Agriculture-Vision: Challenges & Opportunities for Computer Vision in Agriculture*, June 2022, New Orleans, LA.

3. **E. Cai**, S. Baireddy, C. Yang, M. Crawford, and E. J. Delp, "Panicle Counting in UAV Images For Estimating Flowering Time in Sorghum", *Proceedings of the IEEE International Symposium on Geoscience and Remote Sensing*, July 2021, Brussels, Belgium.

4. **E. Cai**, S. Baireddy, C. Yang, M. Crawford, and E. J. Delp, "Deep Transfer Learning For Plant Center Localization", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, the 1st International Workshop and Prize Challenge on Agriculture-Vision: Challenges & Opportunities for Computer Vision in Agriculture*, June 2020, Seattle, WA.

5. C. Yang, S. Baireddy, Y. Chen, **E. Cai**, D. Caldwell, V. Méline, A. S. Iyer-Pascuzzi, E. J. Delp, "Plant Stem Segmentation Using Fast Ground Truth Generation", *Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpretation*, March 2020, Santa Fe, NM.

8.4 Publications Not Resulting From This Thesis

1. J. Guo, C. Yang, **E. Cai**, and E. J. Delp, "Rotation Adaptive Plot Extraction from UAV RGB Images", *Proceedings of the IEEE International Symposium on Geoscience and Remote Sensing*, July 2023, Pasadena, CA.

2. **E. Cai**, R. Rossi, and C. Xiao, "Improving Learning-based Camera Pose Estimation for Image-based Augmented Reality Applications", *In Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems (CHI EA '23)*, April 2023, Hamburg, Germany.

3. C. Yang, S. Baireddy, **E. Cai**, V. Meline, D. Caldwell, A. S. Iyer-Pascuzzi and E. J. Delp, "Image-Based Plant Wilting Estimation", *arXiv preprint arXiv:2105.12926*, 2021.

4. C. Yang, S. Baireddy, **E. Cai**, M. Crawford, and E. J. Delp, "Field-Based Plot Extraction Using UAV RGB Images", *Proceedings of the IEEE International Conference on Computer Vision(ICCV), Workshop on Computer Vision in Plant Phenotyping and Agriculture(CVPPA)*, October 2021, Montreal, Canada.

5. Y. Chen, S. Baireddy, **E. Cai**, C. Yang, and E. J. Delp, "Leaf Segmentation by Functional Modeling", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Workshop on Computer Vision Problems in Plant Phenotyping*, June 2019, Long Beach, CA.

REFERENCES

[1] D. Houle, D. Govindaraju, and S. Omholt, "Phenomics: The next challenge," *Nature reviews. Genetics*, vol. 11, pp. 855–66, Dec. 2010. DOI: 10.1038/nrg2897.

[2] V. Orgogozo, B. Morizot, and A. Martin, "The differential view of genotypephenotype relationships," *Frontiers in Genetics*, vol. 6, p. 179, 2015. DOI: 10.3389/fgene.2015.00179.

[3] N. J. Schork, "Genetics of complex disease: Approaches, problems, and solutions," *American journal of respiratory and critical care medicine*, vol. 156 4 Pt 2, S103–9, 1997. DOI: 10.1164/ajrccm.156.4.12-tac-5.

[4] C. H. Schilling, J. S. Edwards, and B. O. Palsson, "Toward metabolic phenomics: Analysis of genomic data using flux balances," *Biotechnology Progress*, vol. 15, 1999. DOI: 10.1021/bp9900357.

[5] G. P. Wagner, "Characters, units and natural kinds: An introduction," in *The Character Concept in Evolutionary Biology*, G. P. Wagner, Ed., San Diego: Academic Press, 2001, pp. 1–10, ISBN: 978-0-12-730055-9. DOI: 10.1016/B978-012730055-9/50008-2.

[6] R. Bilder, F. Sabb, T. Cannon, E. London, J. Jentsch, D. Parker, R. Poldrack, C. Evans, and N. Freimer, "Phenomics: The systematic study of phenotypes on a genome-wide scale," *Neuroscience*, vol. 164, pp. 30–42, Feb. 2009. DOI: 10.1016/j.neuroscience.2009.01.027.

[7] N. Roll-Hansen, "Sources of wilhelm johannsen's genotype theory," *Journal of the History of Biology*, vol. 42, no. 3, pp. 457–493, 2009. DOI: 10.1007/s10739-008-9166-8.

[8] W. Johannsen, "The genotype conception of heredity," *International Journal of Epidemiology*, vol. 43, no. 4, pp. 989–1000, Mar. 2014, ISSN: 0300-5771. DOI: 10.1093/ije/dyu063.

[9] A. Walter, F. Liebisch, and A. Hund, "Plant phenotyping: From bean weighing to image analysis," *Plant Methods*, vol. 11, no. 1, pp. 1–11, 2015. DOI: 10.1186/s13007-015-0056-8.

[10] F. Fiorani and U. Schurr, "Future scenarios for plant phenotyping," *Annual Review of Plant Biology*, vol. 64, pp. 267–291, Feb. 2013. DOI: 10.1146/annurev-arplant-050312-120137.

[11] D. Kelly, A. Vatsa, W. Mayham, L. Ngô, A. Thompson, and T. KazicDerek, "An opinion on imaging challenges in phenotyping field crops," *Machine Vision and Applications*, pp. 1–14, Dec. 2015. DOI: 10.1007/s00138-015-0728-4.

[12] A. J. Millar, S. R. Short, N.-H. Chua, and S. A. Kay, "A novel circadian phenotype based on firefly luciferase expression in transgenic plants.," *The Plant cell*, vol. 4, pp. 1075–1087, 1992. DOI: 10.1105/tpc.4.9.1075.

[13] J. H. Ko, K. H. Han, S. Park, and J. Yang, "Plant body weight-induced secondary growth in arabidopsis and its transcription phenotype revealed by whole-transcriptome profiling," *Plant physiology*, vol. 135, pp. 1069–83, Jun. 2004. DOI: 10.1104/pp.104.038844.

[14] R. T. Furbank and M. Tester, "Phenomics-technologies to relieve the phenotyping bottleneck," *Trends in Plant Science*, vol. 16, no. 12, pp. 635–644, Nov. 2011. DOI: 10.1016/j.tplants.2011.09.005.

[15] S. C. Chapman, T. Merz, A. Chan, P. Jackway, S. Hrabar, M. F. Dreccer, E. Holland, B. Zheng, T. J. Ling, and J. Jimenez-Berni, "Pheno-copter: A low-altitude, autonomous remote-sensing robotic helicopter for high-throughput field-based phenotyping," *Agronomy*, vol. 4, no. 2, pp. 279–301, Jun. 2014. DOI: 10.3390/agronomy4020279.

[16] R. Makanza, M. Zaman-Allah, J. Cairns, C. Magorokosho, A. Tarekegne, M. Olsen, and B. Prasanna, "High-throughput phenotyping of canopy cover and senescence in maize field trials using aerial digital canopy imaging," *Remote Sensing*, vol. 10, no. 2, p. 330, Feb. 2018. DOI: 10.3390/rs10020330.

[17] A. Singh, B. Ganapathysubramanian, A. Singh, and S. Sarkar, "Machine learning for high-throughput stress phenotyping in plants," *Trends in Plant Science*, vol. 21, no. 2, pp. 110–124, Feb. 2016. DOI: 10.1016/j.tplants.2015.10.015.

[18] C. Costa, U. Schurr, F. Loreto, P. Menesatti, and S. Carpentier, "Plant phenotyping research trends, a science mapping approach," *Frontiers in Plant Science*, vol. 9, p. 1933, 2019. DOI: 10.3389/fpls.2018.01933.

[19] M. Muraya, J. Chu, Y. Zhao, A. Junker, C. Klukas, J. Reif, and T. Altmann, "Genetic variation of growth dynamics in maize (zea mays l.) revealed through automated non-invasive phenotyping," *The Plant Journal*, vol. 89, Jan. 2017. DOI: 10.1111/tpj.13390.

[20] Q. Xiao, X. Bai, C. Zhang, and Y. He, "Advanced high-throughput plant phenotyping techniques for genome-wide association studies: A review," *Journal of Advanced Research*, vol. 35, pp. 215–230, 2021. DOI: 10.1016/j.jare.2021.05.002.

[21] S. Sankaran, L. R. Khot, C. Z. Espinoza, S. Jarolmasjed, V. R. Sathuvalli, G. J. Vandemark, P. N. Miklas, A. H. Carter, M. O. Pumphrey, N. R. Knowles, and M. J. Pavek, "Low-altitude, high-resolution aerial imaging systems for row and field crop phenotyping: A review," *European Journal of Agronomy*, vol. 70, pp. 112–123, Oct. 2015. DOI: 10.1016/j.eja.2015.07.004.

[22] F. A. Vega, F. C. Ramírez, M. P. Saiz, and F. O. Rosúa, "Multi-temporal imaging using an unmanned aerial vehicle for monitoring a sunflower crop," *Biosystems Engineering*, vol. 132, pp. 19–27, Jan. 2015. DOI: 10.1016/j.biosystemseng.2015.01.008.

[23] J. Li, F. Zhang, X. Qian, Y. Zhu, and G. Shen, "Quantification of rice canopy nitrogen balance index with digital imagery from unmanned aerial vehicle," *Remote Sensing Letters*, vol. 6, no. 3, pp. 183–189, Mar. 2015. DOI: 10.1080/2150704X.2015.1021934.

[24] A. Habib, W. Xiong, F. He, H. L. Yang, and M. Crawford, "Improving orthorectification of UAV-Based push-broom scanner imagery using derived orthophotos from frame cameras," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, pp. 262–276, Jan. 2017. DOI: 10.1109/JSTARS.2016.2520929.

[25] J. Fargione, J. Hill, D. Tilman, S. Polasky, and P. Hawthorne, "Land clearing and the biofuel carbon debt," *Science*, vol. 319, no. 5867, pp. 1235–1238, 2008. DOI: 10.1126/science.1152747.

[26] B. Velmurugan, M. Narra, D. M. Rudakiya, and D. Madamwar, "Sweet sorghum: A potential resource for bioenergy production," in *Refining Biomass Residues for Sustainable Energy and Bioproducts*. Academic Press, 2020, pp. 215–242, ISBN: 978-0-12-818996-2. DOI: 10.1016/B978-0-12-818996-2.00010-7.

[27] P. Börjesson and L. M. Tufvesson, "Agricultural crop-based biofuels resource efficiency and environmental performance including direct land use changes," *Journal of Cleaner Production*, vol. 19, no. 2, pp. 108–120, 2011, ISSN: 0959-6526. DOI: 10.1016/j.jclepro.2010.01.001.

[28] K. Smith, A. Mosier, P. Crutzen, and W. Winiwarter, "The role of n2o derived from crop-based biofuels, and from agriculture in general, in earth's climate," *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, vol. 367, pp. 1169–74, May 2012. DOI: 10.1098/rstb.2011.0313.

[29] I. Awudu and J. Zhang, “Uncertainties and sustainability concepts in biofuel supply chain management: A review,” *Renewable and Sustainable Energy Reviews*, vol. 16, no. 2, pp. 1359–1368, 2012, ISSN: 1364-0321. DOI: 10.1016/j.rser.2011.10.016.

[30] M. O’Hare, R. Plevin, J. Martin, A. Jones, A. Kendall, and E. Hopson, “Proper accounting for time increases crop-based biofuels greenhouse gas deficit versus petroleum,” *Environmental Research Letters*, vol. 4, p. 024 001, Apr. 2009. DOI: 10.1088/1748-9326/4/2/024001.

[31] N. Brosse, A. Dufour, X. Meng, Q. Sun, and A. Ragauskas, “Miscanthus: A fast-growing crop for biofuels and chemicals production,” *Biofuels Bioproducts and Biorefining*, vol. 6, pp. 580–598, Jan. 2012. DOI: 10.1002/bbb.1353.

[32] A. Turhollow, E. Webb, and M. Downing, “Review of sorghum production practices: Applications for bioenergy,” *Technical Report*, 2010, Oak Ridge National Laboratory, Oak Ridge, TN. [Online]. Available: https://info.ornl.gov/sites/publications/files/Pub22854.pdf.

[33] S. K. Panguluri and A. A. Kumar, “Phenotyping in sorghum,” *Phenotyping for Plant Breeding: Applications of Phenotyping Methods for Crop Improvement*, vol. 1, pp. 73–110, 2013. DOI: 10.1007/978-1-4614-8320-5.

[34] K. B. Abreha, M. Enyew, A. S. Carlsson, R. R. Vetukuri, T. Feyissa, T. Motlhaodi, D. Nguni, and M. Geleta, “Sorghum in dryland: Morphological, physiological, and molecular responses of sorghum under drought stress,” *Planta*, Jan. 2022. DOI: 10.1007/s00425-021-03799-7.

[35] Y. Assefa, S. Staggenborg, and P. V. V. Prasad, “Grain sorghum water requirement and responses to drought stress: A review,” *Crop Management*, vol. 9, Nov. 2010. DOI: 10.1094/CM-2010-1109-01-RV.

[36] A. Sanchez, P. Subudhi, D. Rosenow, and H. Nguyen, “Mapping qtls associated with drought resistance in sorghum (sorghum bicolor l. moench),” *Plant molecular biology*, vol. 48, pp. 713–26, Mar. 2002. DOI: 10.1023/A:1014894130270.

[37] D. Rosenow, J. Quisenberry, C. Wendt, and L. Clark, “Drought tolerant sorghum and cotton germplasm,” *Agricultural Water Management*, vol. 7, no. 1, pp. 207–222, 1983, Plant production and management under drought conditions, ISSN: 0378-3774. DOI: https://doi.org/10.1016/0378-3774(83)90084-7.

[38] A. Blum and A. Ebercon, “Genotypic responses in sorghum to drought stress. iii. free proline accumulation and drought resistance1,” *Crop Science - CROP SCI*, vol. 16, May 1976. DOI: 10.2135/cropsci1976.0011183X001600030030x.

[39] M. Ahmed, U. Qadeer, and M. Aslam, “Silicon application and drought tolerance mechanism of sorghum,” *African journal of agricultural research*, vol. 6, pp. 594–607, Feb. 2011. DOI: 10.5897/AJAR10.626.

[40] S. Mathur, A. V. Umakanth, V. A. Tonapi, R. Sharma, and M. K. Sharma, “Sweet sorghum as biofuel feedstock: Recent advances and available resources,” *Biotechnology for Biofuels*, vol. 10, no. 1, p. 146, 2017. DOI: 10.1186/s13068-017-0834-9.

[41] C. V. Ratnavathi, S. K. Chakravarthy, V. V. Komala, U. D. Chavan, and J. V. Patil, “Sweet sorghum as feedstock for biofuel production: A review,” *Sugar Tech*, vol. 13, pp. 399–407, 4 Dec. 2011. DOI: 10.1007/s12355-011-0112-2.

[42] P. S. Rao, R. S. Prakasham, P. P. Rao, S. Chopra, and S. Jose, “Sorghum as a sustainable feedstock for biofuels,” *Biomass and Biofuels*, pp. 27–47, Jan. 2015. DOI: 10.1201/b18398.

[43] O. S. Stamenkovi, K. Siliveru, V. B. Veljkovi, I. B. Bankovi-Ili, M. B. Tasi, I. A. Ciampitti, I. G. alovi, P. M. Mitrovi, V. . Sikora, and P. V. Prasad, “Production of biofuels from sorghum,” *Renewable and Sustainable Energy Reviews*, vol. 124, p. 109 769, 2020, ISSN: 1364-0321. DOI: 10.1016/j.rser.2020.109769.

[44] A. Umakanth, A. A. Kumar, W. Vermerris, and V. Tonapi, “Chapter 16 - sweet sorghum for biofuel industry,” in *Breeding Sorghum for Diverse End Uses*, ser. Woodhead Publishing Series in Food Science, Technology and Nutrition, Woodhead Publishing, 2019, pp. 255–270, ISBN: 978-0-08-101879-8. DOI: 10.1016/B978-0-08-101879-8.00016-4.

[45] N. Baral, J. Dahlberg, D. Putnam, J. Mortimer, and C. Scown, “Supply cost and life-cycle greenhouse gas footprint of dry and ensiled biomass sorghum for biofuel production,” *ACS Sustainable Chemistry & Engineering*, Oct. 2020. DOI: 10.1021/acssuschemeng.0c03784.

[46] M. Ewing and S. Msangi, “Biofuels production in developing countries: Assessing tradeoffs in welfare and food security,” *Environmental Science & Policy*, vol. 12, no. 4, pp. 520–528, 2009, Special Issue: Food Security and Environmental Change, ISSN: 1462-9011. DOI: 10.1016/j.envsci.2008.10.002.

[47] J. J. Cheng and G. R. Timilsina, “Status and barriers of advanced biofuel technologies: A review,” *Renewable Energy*, vol. 36, no. 12, pp. 3541–3549, 2011, ISSN: 0960-1481. DOI: 10.1016/j.renene.2011.04.031.

[48] W. Rooney, J. Blumenthal, B. Bean, and J. Mullet, “Designing sorghum as a dedicated bioenergy feedstock,” *Biofuels, Bioproducts and Biorefining*, vol. 1, pp. 147–157, Oct. 2007. DOI: 10.1002/bbb.15.

[49] R. Dar, E. Dar, A. Kaur, and U. Phutela, “Sweet sorghum-a promising alternative feedstock for biofuel production,” *Renewable and Sustainable Energy Reviews*, vol. 82, Nov. 2017. DOI: 10.1016/j.rser.2017.10.066.

[50] T. Gerik, B. Bean, and R. Vanderlip, “Sorghum growth and development,” *Technical Report*, 2003, Texas A&M University, College Station, TX. [Online]. Available: http://glasscock.agrilife.org/files/2015/05/Sorghum-Growth-and-Development.pdf.

[51] R. L. Vanderlip, *How a Sorghum Plant Develops*. Manhattan, KS: Cooperative Extension Service, Kansas State University, 1972. [Online]. Available: https://bookstore.ksre.ksu.edu/pubs/s3.pdf.

[52] J. D. Plessis, *Sorghum production*. Department of Agriculture of Republic of South Africa, 2003. [Online]. Available: http://www.nda.agric.za/docs/Infopaks/FieldCrops_Sorghum.pdf.

[53] X. Wang, C. Hunt, A. Cruickshank, E. Mace, G. Hammer, and D. Jordan, “The impacts of flowering time and tillering on grain yield of sorghum hybrids across diverse environments,” *Agronomy*, vol. 10, no. 1, 2020, ISSN: 2073-4395. DOI: 10.3390/agronomy10010135.

[54] E. Mace, C. Hunt, and D. Jordan, “Supermodels: Sorghum and maize provide mutual insight into the genetics of flowering time,” *Theoretical and applied genetics*, vol. 126, Mar. 2013. DOI: 10.1007/s00122-013-2059-z.

[55] R. Higgins, C. Thurber, I. Assaranurak, and P. Brown, “Multiparental mapping of plant height and flowering time qtl in partially isogenic sorghum families,” *G3 (Bethesda, Md.)*, vol. 4, pp. 1593–602, Sep. 2014. DOI: 10.1534/g3.114.013318.

[56] R. Murphy, D. Morishige, J. Brady, W. Rooney, S. Yang, P. Klein, and J. Mullet, “Ghd7 (ma(6)) represses sorghum flowering in long days: Ghd7 alleles enhance biomass accumulation and grain production,” *The Plant Genome*, vol. 7, no. 2, Jul. 2014. DOI: 10.3835/plantgenome2013.11.0040.

[57] S. U. Bhosale, B. Stich, H. F. W. Rattunde, E. Weltzien, B. I. G. Haussmann, C. T. Hash, P. Ramu, H. E. Cuevas, A. H. Paterson, A. E. Melchinger, and H. K. Parzies, “Association analysis of photoperiodic flowering time genes in west and central african sorghum [sorghum bicolor (l.) moench],” *BMC Plant Biology*, vol. 12, pp. 32–32, 2012. DOI: 10.1186/1471-2229-12-32.

[58] S. Sukumaran, X. Li, X. Li, C. Zhu, R. Perumal, M. Tuinstra, P. V. V. Prasad, S. Mitchell, T. Tesso, and J. Yu, “Qtl mapping for grain yield, flowering time, and stay-green traits in sorghum with genotyping-by-sequencing markers,” *Crop Science*, vol. 56, Jul. 2016. DOI: 10.2135/cropsci2015.02.0097.

[59] Y. El Mannai, T. Shehzad, and K. Okuno, “Variation in flowering time in sorghum core collection and mapping of qtls controlling flowering time by association analysis,” *Genetic Resources and Crop Evolution*, vol. 58, pp. 983–989, Oct. 2011. DOI: 10.1007/s10722-011-9737-y.

[60] C. Allen, P. Prior, and A. Hayward, *Bacterial wilt disease and the Ralstonia solanacearum species complex*. American Phytopathological Society, 2005. [Online]. Available: https://agritrop.cirad.fr/524964/.

[61] S. C. Vanitha, S. R. Niranjana, C. N. Mortensen, and S. Umesha, “Bacterial wilt of tomato in karnataka and its management by pseudomonas fluorescens,” *BioControl*, vol. 54, pp. 685–695, 5 Oct. 2009. DOI: 10.1007/s10526-009-9217-x.

[62] A. C. Hayward, “Biology and epidemiology of bacterial wilt caused by pseudomonas solanacearum,” *Annual Review of Phytopathology*, vol. 29, no. 1, pp. 65–87, 1991. DOI: 10.1146/annurev.py.29.090191.000433.

[63] J. Scott, J.-F. Wang, and P. Hanson, “Breeding tomatoes for resistance to bacterial wilt, a global view,” *Acta Horticulturae*, vol. 695, pp. 161–172, Nov. 2005. DOI: 10.17660/ActaHortic.2005.695.18.

[64] D. Danesh, S. R. Aarons, G. E. McGill, and N. D. Young, “Genetic dissection of oligogenic resistance to bacterial wilt in tomato,” *Molecular plant-microbe interactions : MPMI*, vol. 7 4, pp. 464–71, 1994. DOI: 10.1094/mpmi-7-0464.

[65] Y. Zhang, A. Hu, J. Zhou, W. Zhang, and P. Li, “Comparison of bacterial communities in soil samples with and without tomato bacterial wilt caused by ralstonia solanacearum species complex,” *BMC Microbiology*, vol. 20, Apr. 2020. DOI: 10.1186/s12866-020-01774-y.

[66] Y. Chen, J. Ribera, C. Boomsma, and E. J. Delp, “Locating crop plant centers from UAV-based RGB imagery,” *Proceedings of the IEEE International Conference on Computer Vision, Workshop on Computer Vision Problems in Plant Phenotyping*, Oct. 2017, Venice, Italy. DOI: 10.1109/ICCVW.2017.238.

[67] A. Zeggada, F. Melgani, and Y. Bazi, “A deep learning approach to UAV image multilabeling,” *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 5, pp. 694–698, May 2017. DOI: 10.1109/LGRS.2017.2671922.

[68] Y. Ampatzidis and V. Partel, “UAV-Based high throughput phenotyping in citrus utilizing multispectral imaging and artificial intelligence,” *Remote Sensing*, vol. 11, p. 410, Feb. 2019. DOI: 10.3390/rs11040410.

[69] Y. Chen, J. Ribera, and E. J. Delp, “Estimating plant centers using a deep binary classifier,” *Proceedings of IEEE Southwest Symposium on Image Analysis and Interpretation*, pp. 105–108, Apr. 2018, Las Vegas, NV. DOI: 10.1109/SSIAI.2018.8470367.

[70] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, “A survey on deep transfer learning,” *Proceedings of the International Conference on Artificial Neural Networks*, pp. 270–279, Oct. 2018, Rhodes, Greece. DOI: 10.1007/978-3-030-01424-7_27.

[71] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *Proceedings of the International Conference on Learning Representations*, May 2015, San Diego, CA. DOI: 10.48550/arXiv.1409.1556.

[72] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, Jun. 2016, Las Vegas, NV, ISSN: 1063-6919. DOI: 10.1109/CVPR.2016.90.

[73] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, Jul. 2017, pp. 2261–2269. DOI: 10.1109/CVPR.2017.243.

[74] A. Asgarian, P. Sobhani, J. C. Zhang, M. Mihailescu, A. Sibilia, A. B. Ashraf, and B. Taati, “A hybrid instance-based transfer learning method,” *ArXiv*, vol. abs/1812.01063, 2018. DOI: 10.48550/arXiv.1812.01063.

[75] D. Lin, X. An, and J. Zhang, “Double-bootstrapping source data selection for instance-based transfer learning,” *Pattern Recognition Letters*, vol. 34, no. 11, pp. 1279–1285, 2013, ISSN: 0167-8655. DOI: 10.1016/j.patrec.2013.04.012.

[76] T. Wang, J. Huan, and M. Zhu, “Instance-based deep transfer learning,” in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, Waikoloa Village, HI, 2019, pp. 367–375. DOI: 10.1109/WACV.2019.00045.

[77] B. Wang, M. Qiu, X. Wang, Y. Li, Y. Gong, X. Zeng, J. Huang, B. Zheng, D. Cai, and J. Zhou, “A minimax game for instance based selective transfer learning,” *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 34–43, 2019, Anchorage, AK. DOI: 10.1145/3292500.3330841.

[78] L. Cai, J. Gu, J. Ma, and Z. Jin, “Probabilistic wind power forecasting approach via instance-based transfer learning embedded gradient boosting decision trees,” *Energies*, vol. 12, no. 1, 2019, ISSN: 1996-1073. DOI: 10.3390/en12010159.

[79] L. Liu, Y. Dong, W. Huang, X. Du, J. Luo, Y. Shi, and H. Ma, “Enhanced regional monitoring of wheat powdery mildew based on an instance-based transfer learning method,” *Remote Sensing*, vol. 11, no. 3, 2019, ISSN: 2072-4292. DOI: 10.3390/rs11030298.

[80] G. Kuhlmann and P. Stone, “Graph-based domain mapping for transfer learning in general games,” in *Proceedings of the European Conference on Machine Learning*, Sep. 2007, pp. 188–200. DOI: 10.1007/978-3-540-74958-5_20.

[81] M. Long, Y. Cao, J. Wang, and M. I. Jordan, “Learning transferable features with deep adaptation networks,” *Proceedings of the International Conference on International Conference on Machine Learning*, vol. 37, pp. 97–105, 2015, Lille, France. DOI: 10.48550/arXiv.1502.02791.

[82] Y. Wu, X. Qin, Y. Pan, and C. Yuan, “Convolution neural network based transfer learning for classification of flowers,” in *Proceedings of the IEEE 3rd International Conference on Signal and Image Processing*, 2018, pp. 562–566. DOI: 10.1109/SIPROCESS.2018.8600536.

[83] Q. Yang, W. Shi, J. Chen, and W. Lin, “Deep convolution neural network-based transfer learning method for civil infrastructure crack detection,” *Automation in Construction*, vol. 116, p. 103 199, 2020, ISSN: 0926-5805. DOI: 10.1016/j.autcon.2020.103199.

[84] Y. Lockner and C. Hopmann, “Induced network-based transfer learning in injection molding for process modelling and optimization with artificial neural networks,” *The International Journal of Advanced Manufacturing Technology*, vol. 112, Feb. 2021. DOI: 10.1007/s00170-020-06511-3.

[85] X. Li, T. Pang, B. Xiong, W. Liu, P. Liang, and T. Wang, “Convolutional neural networks based transfer learning for diabetic retinopathy fundus image classification,” *Proceedings of the International Congress on Image and Signal Processing, BioMedical Engineering and Informatics*, pp. 1–11, 2017. DOI: 10.1109/CISP-BMEI.2017.8301998.

[86] C. Yu, J. Wang, Y. Chen, and M. Huang, “Transfer learning with dynamic adversarial adaptation network,” *Proceedings of the IEEE International Conference on Data Mining*, pp. 778–786, 2019. DOI: 10.1109/ICDM.2019.00088.

[87] C. Cheng, B. Zhou, G. Ma, D. Wu, and Y. Yuan, “Wasserstein distance based deep adversarial transfer learning for intelligent fault diagnosis with unlabeled or insufficient labeled data,” *Neurocomputing*, vol. 409, pp. 35–45, 2020, ISSN: 0925-2312. DOI: 10.1016/j.neucom.2020.05.040.

[88] K. Bu, Y. He, X. Jing, and J. Han, “Adversarial transfer learning for deep learning based automatic modulation classification,” *IEEE Signal Processing Letters*, vol. 27, pp. 880–884, 2020. DOI: 10.1109/LSP.2020.2991875.

[89] Y. He, H. Tang, Y. Ren, and A. Kumar, “A deep multi-signal fusion adversarial model based transfer learning and residual network for axial piston pump fault diagnosis,” *Measurement*, vol. 192, p. 110 889, 2022, ISSN: 0263-2241. DOI: 10.1016/j.measurement.2022.110889.

[90] J. Li, R. Huang, G. He, S. Wang, G. Li, and W. Li, “A deep adversarial transfer learning network for machinery emerging fault detection,” *IEEE Sensors Journal*, vol. 20, no. 15, pp. 8413–8422, 2020. DOI: 10.1109/JSEN.2020.2975286.

[91] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 234–241, Oct. 2015, Munich, Germany. DOI: 10.1007/978-3-319-24574-4_28.

[92] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “Imagenet large scale visual recognition challenge,” *International Journal of Computer Vision*, vol. 11, no. 3, pp. 211–252, Dec. 2015. DOI: 10.1007/s11263-015-0816-y.

[93] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" *Proceedings of the International Conference on Neural Information Processing Systems*, vol. 2, pp. 3320–3328, Dec. 2014, Montreal, Canada. DOI: 10.48550/arXiv.1411.1792.

[94] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1717–1724, Jun. 2014, Columbus, OH. DOI: 10.1109/CVPR.2014.222.

[95] H. Ng, V. D. Nguyen, V. Vonikakis, and S. Winkler, "Deep learning for emotion recognition on small datasets using transfer learning," *Proceedings of the ACM International Conference on Multimodal Interaction*, pp. 443–449, 2015, Seattle, WA. DOI: 10.1145/2818346.2830593.

[96] A. Tapas, "Transfer learning for image classification and plant phenotyping," *International Journal of Advanced Research in Computer Engineering and Technology*, vol. 5, no. 11, pp. 2664–2669, 2016. [Online]. Available: http://ijarcet.org/wp-content/uploads/IJARCET-VOL-5-ISSUE-11-2664-2669.pdf.

[97] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, Jun. 2015, Boston, MA. DOI: 10.1109/CVPR.2015.7298594.

[98] M. M. Ghazi, B. Yanikoglu, and E. Aptoula, "Plant identification using deep neural networks via optimization of transfer learning parameters," *Neurocomputing*, vol. 235, no. C, pp. 228–235, Apr. 2017, ISSN: 0925-2312. DOI: 10.1016/j.neucom.2017.01.018.

[99] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 6, pp. 1137–1149, Jun. 2016, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2016.2577031.

[100] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2980–2988, Oct. 2017, Venice, Italy. DOI: 10.1109/iccv.2017.322.

[101] J. Ribera, D. Guera, Y. Chen, and E. J. Delp, "Locating objects without bounding boxes," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6472–6482, Jun. 2019, Long Beach, CA. DOI: 10.1109/CVPR.2019.00664.

[102] S. Aich, I. Ahmed, I. Obsyannikov, I. Stavness, A. Josuttes, K. Strueby, H. Duddu, C. Pozniak, and S. Shirtliffe, "Deepwheat: Estimating phenotypic traits from crop images with deep learning," *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, Mar. 2018, Stateline, NV. DOI: 10.1109/WACV.2018.00042.

[103] J. Wu, G. Yang, X. Yang, B. Xu, L. Han, and Y. Zhu, "Automatic counting of in situ rice seedlings from UAV images based on a deep fully convolutional neural network," *Remote Sensing*, vol. 11, p. 691, Mar. 2019. DOI: 10.3390/rs11060691.

[104] A. Karami, M. Crawford, and E. Delp, "Automatic plant counting and location based on a few-shot learning technique," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 5872–5886, Jan. 2020. DOI: 10.1109/JSTARS.2020.3025790.

[105] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, pp. 62–66, Jan. 1979. DOI: 10.1109/TSMC.1979.4310076.

[106] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 47–60, Nov. 1996. DOI: 10.1109/79.543975.

[107] D. M. W. Powers, "Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011. DOI: 10.9735/2229-3981.

[108] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Proceedings of the International Conference for Learning Representations*, vol. abs/1412.6980, Apr. 2015, San Diego, CA. DOI: 10.48550/arXiv.1412.6980.

[109] Z. Lin and W. Guo, "Sorghum panicle detection and counting using unmanned aerial system images and deep learning," *Frontiers in Plant Science*, vol. 11, p. 1346, 2020. DOI: 10.3389/fpls.2020.534853.

[110] S. Ghosal, B. Zheng, S. Chapman, A. Potgieter, D. Jordan, X. Wang, A. K. Singh, A. Singh, M. Hirafuji, S. Ninomiya, B. Ganapathysubramanian, S. Sarkar, and W. Guo, "A weakly supervised deep learning framework for sorghum head detection and counting," *Plant Phenomics*, vol. 2019, 2019. DOI: 10.34133/2019/1525874.

[111] A. L. Chandra, S. V. Desai, V. N. Balasubramanian, S. Ninomiya, and W. Guo, "Active learning with point supervision for cost-effective panicle detection in cereal crops," *Plant Methods*, vol. 16, Mar. 2020. DOI: 10.1186/s13007-020-00575-8.

[112] V. Giuffrida, P. Doerner, and S. Tsaftaris, “Pheno-deep counter: A unified and versatile deep learning architecture for leaf counting,” *The Plant Journal*, vol. 96, Aug. 2018. DOI: 10.1111/tpj.14064.

[113] S. Kolhar and J. Jagtap, “Plant trait estimation and classification studies in plant phenotyping using machine vision a review,” *Information Processing in Agriculture*, vol. 10, no. 1, pp. 114–135, 2023, ISSN: 2214-3173. DOI: 10.1016/j.inpa.2021.02.006.

[114] J. Xu, J. Yao, H. Zhai, Q. Li, Q. Xu, Y. Xiang, Y. Liu, T. Liu, H. Ma, Y. Mao, F. Wu, Q. Wang, X. Feng, J. Mu, and Y. Lu, “Trichomeyolo: A neural network for automatic maize trichome counting,” *Plant Phenomics*, vol. 5, p. 0024, 2023. DOI: 10.34133/plantphenomics.0024.

[115] A. Dobrescu, V. Giuffrida, and S. Tsaftaris, “Doing more with less: A multitask deep learning approach in plant phenotyping,” *Frontiers in Plant Science*, vol. 11, p. 141, Feb. 2020. DOI: 10.3389/fpls.2020.00141.

[116] A. Dobrescu, M. V. Giuffrida, and S. A. Tsaftaris, “Understanding deep neural networks for regression in leaf counting,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 2600–2608, 2019, Long Beach, CA. DOI: 10.1109/CVPRW.2019.00316.

[117] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2999–3007, Oct. 2017, Venice, Italy. DOI: 10.1109/ICCV.2017.324.

[118] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” *Lecture Notes in Computer Science*, pp. 740–755, 2014, ISSN: 1611-3349. DOI: 10.1007/978-3-319-10602-1_48.

[119] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 936–944, Jul. 2017, Honolulu,HI. DOI: 10.1109/CVPR.2017.106.

[120] G. Jocher *et al.*, *ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation*, version v7.0, Nov. 2022. DOI: 10.5281/zenodo.7347926.

[121] C. Wang, H. M. Liao, Y. Wu, P. Chen, J. Hsieh, and I. Yeh, "Cspnet: A new backbone that can enhance learning capability of cnn," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1571–1580, Jun. 2020. DOI: 10.1109/CVPRW50498.2020.00203.

[122] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8759–8768, Jun. 2018, Salt Lake City, UT. DOI: 10.1109/CVPR.2018.00913.

[123] R. Xu, H. Lin, K. Lu, L. Cao, and Y. Liu, "A forest fire detection system based on ensemble learning," *Forests*, vol. 12, p. 217, Feb. 2021. DOI: 10.3390/f12020217.

[124] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Proceedings of the International Conference on Neural Information Processing Systems*, vol. 25, 2012. DOI: 10.1145/3065386.

[125] T. M. K. C. Shorten, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, Jun. 2019. DOI: 10.1186/s40537-019-0197-0.

[126] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 2672–2680, Dec. 2014, Montreal, Canada. DOI: 10.48550/arXiv.1406.2661.

[127] L. Bi and G. Hu, "Improving image-based plant disease classification with generative adversarial network under limited training set," *Frontiers in Plant Science*, vol. 11, 2020, ISSN: 1664-462X. DOI: 10.3389/fpls.2020.583438.

[128] S. L. Madsen, M. Dyrmann, R. N. Jørgensen, and H. Karstoft, "Generating artificial images of plant seedlings using generative adversarial networks," *Biosystems Engineering*, vol. 187, pp. 147–159, 2019, ISSN: 1537-5110. DOI: 10.1016/j.biosystemseng.2019.09.005.

[129] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length and helmholtz free energy," *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 3–10, 1993, Denver, Colorado. [Online]. Available: https://papers.nips.cc/paper_files/paper/1993/file/9e3cfc48eccf81a0d57663e129aef3cb-Paper.pdf.

[130] M. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)a review of applications in the atmospheric sciences," *Atmospheric Environment*, vol. 32, no. 14, pp. 2627–2636, 1998, ISSN: 1352-2310. DOI: 10.1016/S1352-2310(97)00447-0.

[131] A. Jabbar, X. Li, and B. Omar, "A survey on generative adversarial networks: Variants, applications, and training," *ACM Computing Surveys.*, vol. 54, no. 8, Oct. 2021, ISSN: 0360-0300. DOI: 10.1145/3463475.

[132] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," *Proceedings of the International Conference on Machine Learning*, pp. 214–223, Aug. 2017, Sydney, Australia. DOI: 10.48550/arXiv.1701.07875.

[133] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, pp. 99–121, 2000. DOI: 10.1023/A:1026543900054.

[134] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *Proceedings of the International Conference on Learning Representations*, Nov. 2016, San Juan, Puerto Rico. DOI: 10.48550/arXiv.1511.06434.

[135] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014. DOI: 10.48550/arXiv.1411.1784.

[136] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," *Proceedings of The International Conference on Machine Learning*, vol. 48, pp. 1060–1069, Jun. 2016. DOI: 10.48550/arXiv.1605.05396.

[137] M. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 700–708, 2017, Long Beach, CA. DOI: 10.48550/arXiv.1703.00848.

[138] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4396–4405, 2019, Long Beach, CA. DOI: 10.1109/CVPR.2019.00453.

[139] P. Isola, J. Zhu, T. Zhou, and A. Efros, "Image-to-image translation with conditional adversarial networks," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5967–5976, Jul. 2017, Honolulu, HI. DOI: 10.1109/CVPR.2017.632.

[140] J. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2242–2251, 2017, Venice, Italy. DOI: 10.1109/ICCV.2017.244.

[141] S. Milz, T. Rüdiger, and S. Süss, “Aerial ganeration: Towards realistic data augmentation using conditional gans,” *Proceedings of the European Conference on Computer Vision*, pp. 59–72, Jan. 2019. DOI: 10.1007/978-3-030-11012-3_5.

[142] V. Sandfort, K. Yan, P. Pickhardt, and R. Summers, “Data augmentation using generative adversarial networks (cyclegan) to improve generalizability in ct segmentation tasks,” *Scientific Reports*, vol. 9, p. 16 884, Nov. 2019. DOI: 10.1038/s41598-019-52737-x.

[143] T. Wang, M. Liu, J. Zhu, A. Tao, J. Kautz, and B. Catanzaro, “High-resolution image synthesis and semantic manipulation with conditional GANs,” *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8798–8807, Jun. 2018, Los Alamitos, CA. DOI: 10.1109/CVPR.2018.00917.

[144] T. Park, M. Liu, T. Wang, and J. Zhu, “Semantic image synthesis with spatially-adaptive normalization,” *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2332–2341, Jun. 2019, Long Beach, CA. DOI: 10.1109/CVPR.2019.00244.

[145] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, “Scene parsing through ade20k dataset,” *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5122–5130, 2017, Honolulu, HA. DOI: 10.1109/CVPR.2017.544.

[146] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for semantic urban scene understanding,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3213–3223, 2016, Las Vegas, NV. DOI: 10.1109/CVPR.2016.350.

[147] Y. Grandvalet and Y. Bengio, “Semi-supervised learning by entropy minimization,” *Proceedings of the International Conference on Neural Information Processing Systems*, vol. 17, L. Saul, Y. Weiss, and L. Bottou, Eds., 2004. [Online]. Available: https://proceedings.neurips.cc/paper/2004/file/96f2b50b5d3613adf9c27049b2a888c7-Paper.pdf.

[148] D. Berthelot, N. Carlini, I. Goodfellow, A. Oliver, N. Papernot, and C. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," *Proceedings of the International Conference on Neural Information Processing Systems*, 2019. DOI: 10.48550/arXiv.1905.02249.

[149] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *Proceedings of the International Conference on Learning Representations*, 2017. DOI: 10.48550/arXiv.1609.02907.

[150] Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, and Q. V. Le, "Unsupervised data augmentation for consistency training," *Proceedings of the International Conference on Neural Information Processing Systems*, 2020. DOI: 10.48550/arXiv.1904.12848.

[151] D. Lee, "Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks," *International Conference on Machine Learning Workshop: Challenges in Representation Learning*, Jul. 2013.

[152] K. Sohn, D. Berthelot, C. Li, Z. Zhang, N. Carlini, E. D. Cubuk, A. Kurakin, H. Zhang, and C. Raffel, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," *Proceedings of the International Conference on Neural Information Processing Systems*, 2020. DOI: 10.48550/arXiv.2001.07685.

[153] K. Sohn, Z. Zhang, C. Li, H. Zhang, C. Lee, and T. Pfister, "A simple semi-supervised learning framework for object detection," *arXiv:2005.04757*, 2020. DOI: 10.48550/arXiv.2005.04757.

[154] Y. Liu, C. Ma, Z. He, C. Kuo, K. Chen, P. Zhang, B. Wu, Z. Kira, and P. Vajda, "Unbiased teacher for semi-supervised object detection," in *Proceedings of the International Conference on Learning Representations*, 2021. DOI: 10.48550/arXiv.2102.09480.

[155] M. Xu, Z. Zhang, H. Hu, J. Wang, L. Wang, F. Wei, X. Bai, and Z. Liu, "End-to-end semi-supervised object detection with soft teacher," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3040–3049, 2021. DOI: 10.1109/ICCV48922.2021.00305.

[156] B. Xu, M. Chen, W. Guan, and L. Hu, "Efficient teacher: Semi-supervised object detection for yolov5," *arXiv preprint arXiv:2302.07577*, 2023. DOI: 10.48550/arXiv.2302.07577.

[157] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,” *arXiv preprint arXiv:2207.02696*, 2022. DOI: 10.48550/arXiv.2207.02696.

[158] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, “Deep unsupervised learning using nonequilibrium thermodynamics,” in *Proceedings of the International Conference on Machine Learning*, Lille, France, vol. 37, PMLR, Jul. 2015, pp. 2256–2265. DOI: 10.48550/arXiv.1503.03585.

[159] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” *Proceedings of the International Conference on Neural Information Processing Systems*, vol. 33, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., pp. 6840–6851, 2020. DOI: 10.48550/arXiv.2006.11239.

[160] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, . Kaiser, and I. Polosukhin, “Attention is all you need,” *Proceedings of the International Conference on Neural Information Processing Systems*, vol. 30, 2017. DOI: 10.48550/arXiv.1706.03762.

[161] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, “High-resolution image synthesis with latent diffusion models,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10 674–10 685, Jun. 2022, New Orleans, LA. DOI: 10.1109/CVPR52688.2022.01042.

[162] M. Agarwal, A. Singh, S. Arjaria, A. Sinha, and S. Gupta, “Toled: Tomato leaf disease detection using convolution neural network,” *Procedia Computer Science*, vol. 167, pp. 293–301, 2020, ISSN: 1877-0509. DOI: 10.1016/j.procs.2020.03.225.

[163] S. Zhao, Y. Peng, J. Liu, and S. Wu, “Tomato leaf disease diagnosis based on improved convolution neural network by attention module,” *Agriculture*, vol. 11, no. 7, p. 651, 2021, ISSN: 2077-0472. DOI: 10.3390/agriculture11070651.

[164] A. Hidayatuloh, M. Nursalman, and E. Nugraha, “Identification of tomato plant diseases by leaf image using squeezenet model,” *Proceedings of International Conference on Information Technology Systems and Innovation*, pp. 199–204, Oct. 2018. DOI: 10.1109/ICITSI.2018.8696087.

[165] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size,” *ArXiv*, vol. abs/1602.07360, 2016. DOI: 10.48550/arXiv.1602.07360.

[166] C. G. Harris and M. J. Stephens, “A combined corner and edge detector,” *Proceedings of Alvey Vision Conference*, vol. 15, no. 50, pp. 147–151, Aug. 1988. [Online]. Available: http://www.bmva.org/bmvc/1988/avc-88-023.pdf.

[167] “Multimedia systems and equipment - colour measurement and management - part 2-1: Colour management - default rgb colour space - srgb, iec 61966-2-1:1999,” *International Electrotechnical Commission (IEC)*, 1999, Geneva, Switzerland. [Online]. Available: https://webstore.iec.ch/publication/6169.

VITA

Enyu Cai was born in Hunan, China. In 2018, he received his Bachelor of Science degree in Electrical Engineering from Purdue University, West Lafayette, Indiana, USA. He started his Ph.D. program at Purdue University and joined the Video and Image Processing (VIPER) laboratory in 2018. While in the graduate program at Purdue, he worked on projects sponsored by the Advanced Research Projects Agency-Energy (ARPA-E). His current research interests include machine learning, deep learning, image processing, and computer vision. He is a student member of the IEEE and the IEEE Signal Processing Society.