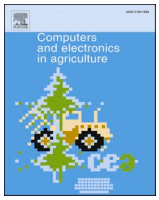
Computers and Electronics in Agriculture 194 (2022) 106734

Contents lists available at ScienceDirect

Computers and Electronics in Agriculture

journal homepage: www.elsevier.com/locate/compag



Weakly-supervised learning to automatically count cotton flowers from aerial imagery

Daniel Petti [a], Changying Li [b,*]

[a] Bio-Sensing and Instrumentation Laboratory, 702 Boyd Graduate Studies Building, University of Georgia, Athens 30602, Georgia, USA
[b] Phenomics and Plant Robotics Center, University of Georgia, USA

ARTICLE INFO

Keywords:
Multiple-instance learning
Active learning
Object counting
High-throughput phenotyping

ABSTRACT

Counting plant flowers is a common task with applications for estimating crop yields and selecting favorable genotypes. Typically, this requires a laborious manual process, rendering it impractical to obtain accurate flower counts throughout the growing season. The model proposed in this study uses weak supervision, based on Convolutional Neural Networks (CNNs), which automates such a counting task for cotton flowers using imagery collected from an unmanned aerial vehicle (UAV). Furthermore, the model is trained using Multiple Instance Learning (MIL) in order to reduce the required amount of annotated data. MIL is a binary classification task in which any image with at least one flower falls into the positive class, and all others are negative. In the process, a novel loss function was developed that is designed to improve the performance of image-processing models that use MIL. The model is trained on a large dataset of cotton plant imagery which was collected over several years and will be made publicly available. Additionally, an active-learning-based approach is employed in order to generate the annotations for the dataset while minimizing the required amount of human intervention. Despite having minimal supervision, the model still demonstrates good performance on the testing dataset. Multiple models were tested with different numbers of parameters and input sizes, achieving a minimum average absolute count error of 2.43. Overall, this study demonstrates that a weakly-supervised model is a promising method for solving the flower counting problem while minimizing the human labeling effort.

1. Introduction

Counting plant organs is a common problem within the broader field of high-throughput phenotyping (HTP) (Chawade et al., 2019). This technique is frequently used for generating yield estimates. Specifically, counts of flowers are useful both for yield estimation, and for assessing flowering patterns over the course of the growing season. This is critical to deriving flowering time, an important phenotypic trait reflecting the transition from vegetative growth to reproductive growth in plants (Oosterhuis, 1990). Traditional plant organ counting is done by human visual evaluation, which is time-consuming, tedious, and infeasible for large fields.

Plant organ counting using imaging methods has been explored within a wide range of prior works. The simplest approaches often rely entirely on classical computer vision techniques such as color thresholding, but can still achieve acceptable performance within limited domains (Guo et al., 2018; Thorp et al., 2016; Adamsen et al., 2000). More recently, Convolutional Neural Networks (CNNs) have been adopted for this task. Many of these approaches adapt existing object detection frameworks, such as Faster RCNN (Ren et al., 2015), to count plant organs. One successful CNN-based plant organ counting system is TasselNet (Lu et al., 2017), which is designed to count maize tassels in images taken from cameras mounted on poles. TasselNet is a counting-by-regression approach that takes inspiration from the redundant, patch-wise counting pioneered by Cohen et al. (2017). This allows for the averaging out of errors and even the production of primitive density maps, all with a relatively simplistic architecture that merely regresses a count value for each image patch. TasselNet is versatile, and it has been used to count other plant organs besides maize tassels. Xiong et al. (2019a) not only adapted TasselNet to count wheat spikes, but also improved the performance by enlarging the receptive field of the model in order to handle truncated instances better. Madec et al. (2019) also applied TasselNet to a large and diverse dataset of wheat spike images, but found that a simple counting-by-detection approach using Faster RCNN (Ren et al., 2015) worked better in certain cases. Rahnemoonfar and Sheppard (2017) adapted the counting-by-regression approach to

* Corresponding author at: Professor, College of Engineering; Director, Phenomics and Plant Robotics Center, University of Georgia.
E-mail addresses: djp44210@uga.edu (D. Petti), cyli@uga.edu (C. Li).

https://doi.org/10.1016/j.compag.2022.106734
Received 21 October 2021; Received in revised form 11 January 2022; Accepted 17 January 2022
Available online 15 February 2022

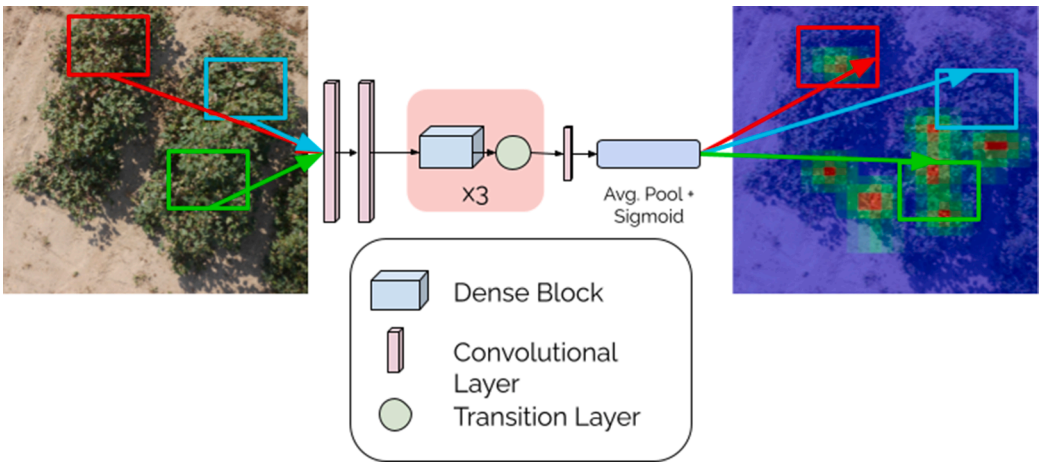


Fig. 1. An overview of the multi-instance learning approach for cotton bloom counting from aerial imagery. The same-colored bounding boxes represent the same patch from the input and output images. The model then computes the areas of the output density map corresponding to each patch.

fruit counting, proving that a CNN can accurately count tomatoes in the field even when trained on a purely synthetic dataset. An alternative approach to a similar problem is leveraged by Chen et al. (2017). In this case, the authors divided the problem into a semantic segmentation task for localizing overlapping fruit clusters, along with a separate counting task that regresses the number of fruits in each cluster.

Flower counting has been investigated by prior work using classical image processing methods—such as thresholding segmentation—for certain species with conspicuous blooms, leveraging their distinct colors in order to distinguish them from the surrounding canopy (Thorp et al., 2016; Adamsen et al., 2000). Such approaches, however, are limited in their accuracy and generalization ability. Jiang et al. (2020) proposed a counting-by-detection approach based on Faster RCNN to detect and count flowers in images taken from a ground vehicle. Compared to aerial counting, this approach suffers less from occlusion and low resolution, and thus can yield higher count accuracy at the expense of lower throughput. In contrast, the approach proposed by Xu et al. (2018) uses imagery collected from a drone and performs the counting using simple thresholding, with a CNN employed to reduce the false-positive rate. By applying the structure-from-motion 3D reconstruction technique, the researchers were able to recover an approximate 3D position for the detected flowers. One drawback of this approach is that it still relies on thresholding in order to propose candidate flowers, which has limited reliability in practice. Recently, Sun et al. (2021) attempted to couple a CNN trained to segment blossoms on fruit trees with an active contour refinement step to improve the quality of the segmentation results.

One major challenge for plant organ counting is that traditional supervised learning approaches tend to be limited by a relative paucity of annotated data. Weakly-supervised learning has been proposed as a general approach to overcome this issue, encompassing a broad category of techniques that require less supervision than standard fully-supervised learning. These approaches include ranking the density of image crops (Liu et al., 2018), and using unlabeled images to train an autoencoder (Sam et al., 2019). Some researchers employ a Generative Adversarial Network (GAN) as part of an auxiliary task to generate artificial training images or sharpen density maps (Olmschenk et al., 2019; Shen et al., 2018; Giuffrida et al., 2019). Alternatively, Akiva et al. (2020) use a novel combinatorial loss function with clever shape priors in order to segment cranberries with only point supervision, an approach that is similar conceptually to Laradji et al. (2018). Bellocchio et al. (2019) propose a weakly-supervised counting method to count several varieties of fruit. Their approach uses two separate CNN models, one of which performs binary classification of images depending on whether they contain fruit or not. Their final count is regressed by a secondary model that uses cross-scale consistency priors to provide a weak supervision signal, similar to the loss function proposed in this work.

Multiple instance learning (MIL) is one approach used in weakly-supervised learning, and can be thought of as a special case of a counting-by-classification problem in which there are only two classes: positive or negative (Foulds and Frank, 2010). As a framework for detection, MIL provides the distinct advantage of simplifying annotations: the annotator simply needs to select the positive or negative class for an image based on whether *any* object is visible. MIL has not been applied widely to counting problems, even less so for plant organ counting. Bollis et al. (2020), however, did employ MIL in order to detect citrus mites in magnified photographs, which is somewhat similar to plant organ counting. Compared to counting-by-regression, all counting-by-classification approaches have the drawback of introducing quantization errors into counts. In the case of MIL, this effect is especially severe, because any image with one or more examples will be lumped into the same positive class. Despite this, most prior work has found that the improved ability to handle unbalanced datasets that counting-by-classification affords generally outweighs any performance decrease from quantization error (Liu et al., 2019). Xiong et al. (2019b) also evaluated counting-by-classification in conjunction with a novel Spatial Divide-and-Conquer approach, producing state-of-the-art results.

Many existing plant organ counting approaches have significant disadvantages when naively applied to the problem of counting cotton blossoms. The patch-wise regression approach developed by Lu et al. (2017), for example, works best with plant organs that cluster *densely*, but cotton blossoms tend to be sparse. The object detection approach from Ghosal et al. (2019), in contrast, works well for these kinds of data, but requires detailed bounding-box annotations around each instance. Weekly-supervised approaches can help avoid the need for such cumbersome annotations, but come with their own trade-offs. The approach proposed by Bellocchio et al. (2019), for example, requires the training of two separate models, which increases the computational burden. Though the use of shape priors (Akiva et al., 2020) is convenient, it requires a target organ with a consistent and easily-described shape. Ubbens et al. (2020) proposed a completely unsupervised approach, but it is limited in its ability to detect plant organs in cases with heavy occlusion and low background contrast.

To address these issues, the weakly-supervised multi-instance learning (MIL) task (Foulds and Frank, 2010) is adapted for use in counting cotton blossoms in aerial images. This allows for the avoidance of expensive point annotations on much of the dataset, substituting simpler image-level annotations. Furthermore, unlike the two-model approach proposed by Bellocchio et al. (2019), the approach proposed

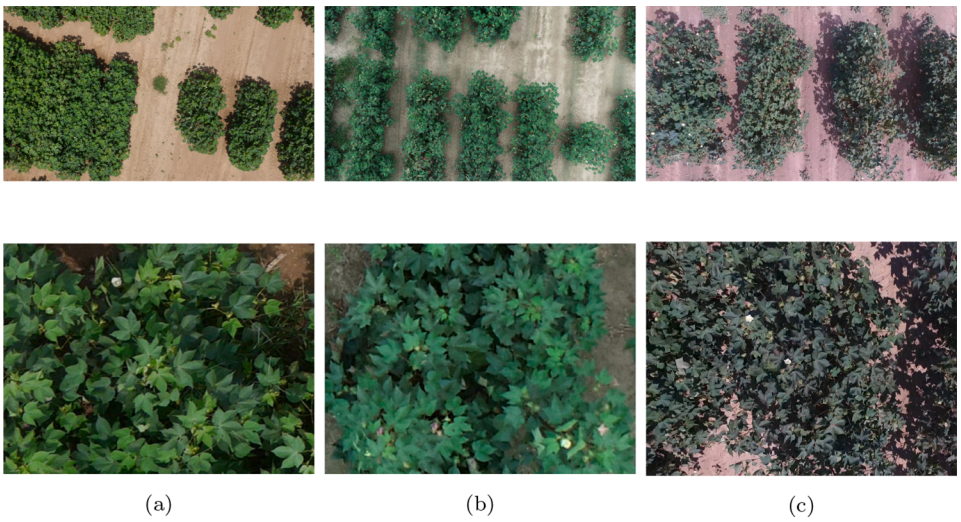


Fig. 2. Example images from the cotton blossom dataset, cropped and zoomed to show details. (a) shows an example image taken with a Panasonic Lumix G6 from 15 m. (b) shows an example image taken with a DJI Phantom 4 Pro from 10 m. (c) shows an example frame from a video taken by the same drone flying at 4 m. The top row shows the full images, and the bottom row shows zoomed regions.

here employs a single model with a fully-convolutional architecture which minimizes the computational burden. Inspired by Ghosal et al. (2019), a technique similar to active learning (Settles, 2009) is also adopted in order to accelerate the annotation of the data. A novel variant of the cross-scale consistency loss technique described by Shen et al. (2018) is employed as well in order to provide stronger supervision. The formulation of this loss is specifically adapted to the MIL task. Finally, this work makes use of a large dataset of annotated images—as collected by a drone from actual cotton fields—which will be made publicly available for the benefit of future researchers. The overall goal of this study was to investigate the feasibility of applying MIL to counting cotton blooms from aerial imagery. Our paper contains three main contributions:

1. We create a dataset of aerial cotton plant imagery using pseudo-active learning;
2. We develop and train a weakly-supervised learning framework that leverages the MIL problem formulation for counting cotton blooms;
3. We examine the key factors—such as model architecture, model depth, and patch size/stride—that impact the performance of MIL-based counting approaches.

2. Materials and methods

2.1. Overview of methodology

Broadly, the heart of the proposed method consists of a CNN that is trained on annotated aerial images of cotton plants to count visible blossoms (Fig. 1). It is common that attempts to count sparsely-distributed plant organs use a counting-by-detection approach (Ghosal et al., 2019; Jiang et al., 2020). This is a simple, intuitive paradigm that allows for the re-use of state-of-the-art object detection models. Counting-by-detection, however, is not very suitable for weakly-supervised learning. Most detectors are trained in a fully-supervised manner, using bounding-box annotations on the training images.

Instead, we use a MIL approach in which images are classified simply as containing at least one flower or not. A rough count can be computed with this model by dividing the input image into many small, overlapping patches, and applying the model to each patch individually. Assuming that the flowers are sparse enough and the patches are small enough that there are few cases where multiple flowers appear in one patch, this can be sufficient for accurate counting. Our counting model is based on DenseNet (Huang et al., 2017), which is a CNN that leverages dense connections in between its layers in order to encourage filter reuse. DenseNet therefore has high parameter efficiency compared to other approaches, making it useful for a counting problem with limited training data. Our DenseNet is configured as a simple binary classifier that takes image patches as input. Each component of the data processing pipeline is described in detail in the following sections.

2.2. Cotton blossom dataset

The dataset comprises images collected using various platforms flying at different altitudes. Data were collected at UGA's Iron Horse Farm research site (33°43'40.6"N, 83°18'15.2"W) during multiple sessions in 2016, 2018 and 2020, throughout the months from August to September. The fields used for data collection have a sandy soil type. The cotton was planted in rows with 6-foot spacing, and four plants per plot. A linear irrigation system was used to provide additional water when needed. The dataset includes data from a total of 6 of these sessions, two of which are used exclusively for testing (see below). Example images are shown in Fig. 2. See Appendix A for details about how the dataset was constructed.

The dataset is divided into two parts, A and B, each with different types of annotations. Part A is derived from 148 images with the point location of every single blossom annotated. Part B consists of 18,400 image patches, extracted from images not in part A. Most patches are 1/64 the size of an image in part A. However, since the original data contain some video, which is shot at a lower resolution and altitude, each patch extracted from these video frames is only 1/4 the size of the original frame. It was found empirically that this produces results that resemble the patches extracted from still images. Instead of annotations for every blossom, part B simply contains patch-wide annotations indicating whether or not at least one blossom is present in that patch. Though annotations for pollinated (pink) flowers are included in parts of the dataset, in practice, these are ignored during training and evaluation. That is, the model is trained using *only* un-pollinated (white) flowers as positive examples. This decision was made because knowing the number of pollinated flowers is less important to breeders. Additionally, a small external validation set of images—which we refer to as part C in Table 1—is used to explore the limits of the model's generalization ability. It contains 17 images collected and annotated in the same manner as part A, and is used later on to evaluate inference hyper-parameters (see Section 3.2).

Table 1
Comparison of the three parts of the cotton blossom dataset. Note that part A is split between training, testing, and validation (A1 and A2), and has point annotations. Part B is used entirely for training, and has binary annotations. Part C is used to evaluate hyperparameters.

Part	# of Examples	% Negative	Image Size (px)
A1 (training)	108 images	78%	4608 × 3456
B (training)	19,180 patches	68%	576 × 432
A2 (testing)	21 images	91%	4608 × 3456
A2 (validation)	19 images	91%	4608 × 3456
C (validation)	17 images	80%	4608 × 3456

The dataset as a whole is relatively unbalanced, with roughly 78% of the patches in the part A training set containing no flowers. Part B is a little more balanced, with roughly 68% of images containing no flowers. Consequently, when training the models, deliberate steps are taken to alleviate the imbalance. For the initial investigations, part A of the data is divided into training and test/validation sets, which are denoted as "A1" and "A2", respectively (Table 1). Since the data was collected in multiple sessions, A2 is derived from a single session that is split into testing and validation sets, while the rest of the data from part A is used for training (A1). Since there are no sessions split between A1 and A2, the testing and validation sets should represent a "worst case" scenario in terms of performance. Most results are reported on the validation set, which is made up of 19 raw images. Part B is for training exclusively, since it contains no point-wise annotations, and therefore provides no ground-truth count estimates that can be used for evaluation.

2.3. Pseudo-active learning

Part B of the cotton blossom dataset is generated using an approach similar to active learning, but more simplistic. Inspired by Ghosal et al. (2019), a model is initially trained using only patches from part A1. 640 unannotated patches from part B are then selected, and the trained model is used to generate approximate annotations. In contrast to "standard" active learning approaches, random selection is employed in order to select new images to annotate, as opposed to any sort of heuristic. These annotations are then verified and corrected by a human using the open-source CVAT application. Finally, the corrected data are used to re-train the model. The process is then repeated, doubling the number of images that are annotated each cycle. In this way, all the images in part B are annotated with minimal human intervention.

2.4. Counting model

The proposed model adheres to the MIL task. MIL is a binary classification task with negative and positive classes. An example image falls into the positive class if it contains at least one blossom, and the negative class otherwise (Foulds and Frank, 2010).

Specifically, the model uses a fully-convolutional architecture based on DenseNet-121 (Huang et al., 2017). The model is trained using inputs of size 576 × 432. These inputs consist of patches that are extracted from the raw input images, as described in Section 2.2. Some slight modifications are made to the DenseNet architecture, including replacing the initial 7x7 convolution with 2 consecutive 3x3 convolutions (Fig. 3). This reduces the parameter count and experimental evidence suggests that it improves performance on the small dataset used in this study. Huang et al. (2017) define the concept of a "growth rate", which measures how many additional filters are added at each layer. Though the original DenseNet work used a growth rate of 32, a growth rate of only 4 was determined empirically to work well in this application. The detailed architecture is shown in Table 2. Additionally, several model variations are tested which differ only in depth. Table 2 shows the deepest model, with 58 layers in the dense blocks, distributed in a pattern of 6, 12, 24, and 16 per block. Versions with 29 layers (3, 6, 12 and 8 per block), and 15 layers (2, 3, 6, and 4 per block) are also explored.

Periodic batch normalization (Ioffe and Szegedy, 2015) layers are used in order to improve convergence speed, as specified in Huang et al. (2017). There are a total of four max pooling layers which down-sample the input by a factor of 16. The resulting features then undergo global average pooling and sigmoid activation in order to generate logits for each class.

2.4.1. Training

The model is trained using both parts A1 and B of the cotton blossom dataset. For part A1, random patches are extracted from the input images that are the same size as the images in part B. A positive or negative label is then generated for the extracted patch based on whether or not point annotations are present within it. For part B, no such manipulation is necessary. However, the positive class is oversampled when constructing minibatches in order to help alleviate the class imbalance.

In all the experiments, the model is trained for a total of 35 epochs, employing momentum stochastic gradient descent (SGD) with warm restarts (Loshchilov and Hutter, 2016). The initial learning rate is 0.01,

Table 2
Detailed 58-layer model architecture, based on DenseNet-121 (Huang et al., 2017), with $k = 4$.

Layer	Output Size	Details
Convolution	432 × 576 × 48	3 × 3 conv
Convolution	432 × 576 × 48	3 × 3 conv
Pooling	216 × 288 × 48	2 × 2 max pool
Dense Block (1)	216 × 288 × 4	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	216 × 288 × 2	1 × 1 conv
	108 × 144 × 2	2 × 2 max pool
Dense Block (2)	108 × 144 × 4	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	108 × 144 × 2	1 × 1 conv
	54 × 72 × 2	2 × 2 max pool
Dense Block (3)	54 × 72 × 4	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$
Transition Layer (3)	54 × 72 × 2	1 × 1 conv
	27 × 36 × 2	2 × 2 max pool
Dense Block (4)	27 × 36 × 4	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$
Classification	27 × 36 × 1	1 × 1 conv
	1 × 1 × 1	27 × 36 average pool + sigmoid

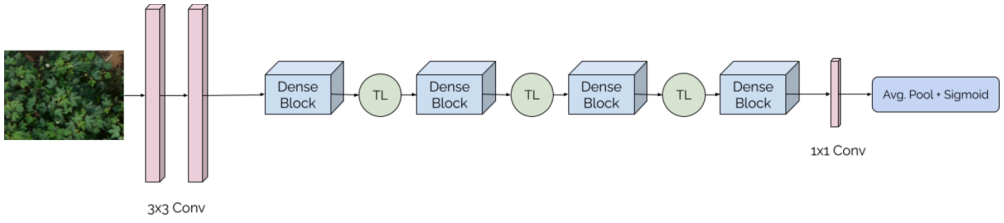


Fig. 3. The model contains 4 dense blocks, with transition layers (TL) between them, as described by Huang et al. (2017).

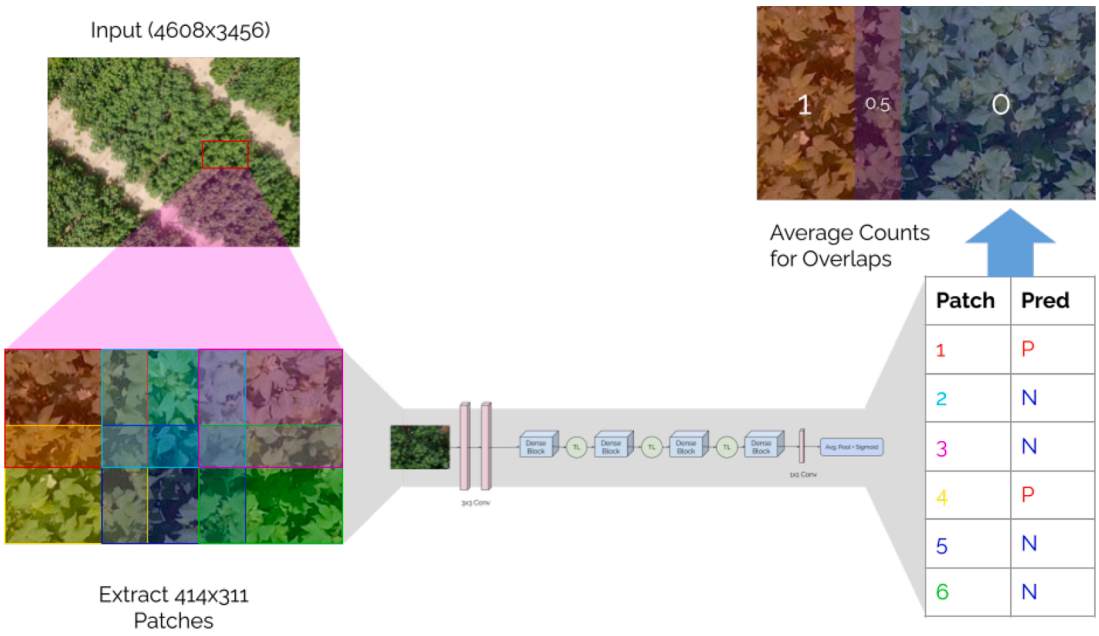


Fig. 4. Inference is conducted by extracting overlapping patches (bottom left) from the entire input image (top left). (The extraction process is only shown on a subset of the image here.) These are then fed through the model (center), producing patch-wise predictions (center right). Uniform density is assumed for each patch, and density values for the overlapping pixels in each patch are averaged to produce a pseudo-density map (bottom right). Best viewed in color.

which is decreased to 1e-7 over the course of each restart period. Training is initialized with a period of 5 epochs between restarts, which is doubled after every restart. This process takes approximately 5 h on an Nvidia V100 GPU with a batch size of 16.

2.4.2. *Inference*

Because MIL introduces severe quantization error, the best counting accuracy can be achieved by dividing an input image up into small patches and predicting each individually. Inspired by Cohen et al. (2017), patches are chosen such that they overlap slightly in order to average out errors and eliminate issues with detecting blossoms that appear right on patch boundaries.

During inference, all such patches are extracted from an input. If the model classifies a particular patch as the positive class, each pixel in the patch is assigned a density of 1.0 divided by the number of pixels in the patch, such that the summation of all pixel values in the patch is one. The density value for each pixel is then divided by the number of patches that overlap at that pixel to eliminate the redundant counts. This process is illustrated in Fig. 4. Even though the model does not directly produce a density map as its output—as is the case with counting-by-density-regression approaches—this patch-wise inference process enables the production of a pseudo-density map for an entire input image. As with a normal density map, the full count can be estimated by summing the density values for every pixel.

Though applying the model individually to every overlapping patch can be slow, the fully-convolutional nature of the model can be leveraged in order to optimize this process. Specifically, inference is performed on the *entire input image*, and the resulting activation map is extracted right before performing global average pooling. The patch extraction and global average pooling for each patch are then performed directly on the activation map. This dramatically speeds up the inference process without changing the result, although note that, in practice, the memory usage will increase also due to inference being performed on a substantially larger input.

2.5. *Cross-scale consistency loss*

Since the model works on image patches, it is always possible to divide each patch up into smaller sub-patches and apply the model to each sub-patch individually. Indeed, since the model is fully convolutional, this can essentially be done for free: applying the model separately to each sub-patch is equivalent to applying the model once to the entire patch and then extracting equivalent sub-patches from the final feature map (Fig. 5). This section explains how to leverage this property

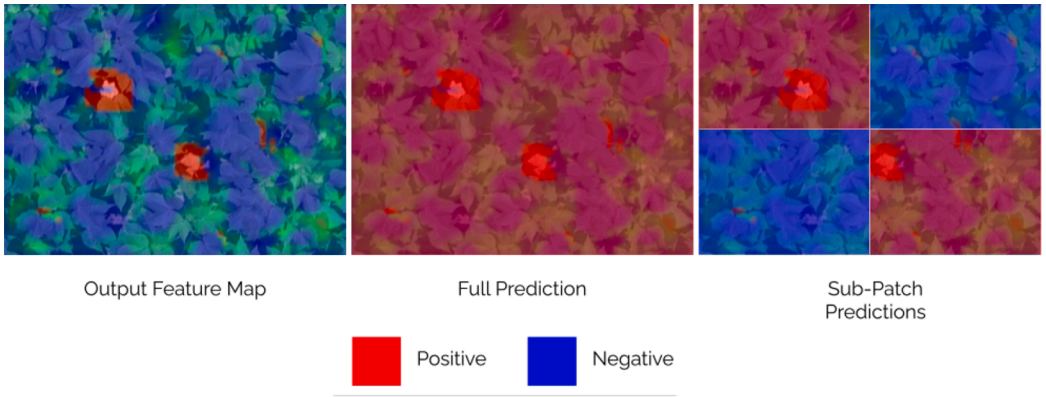


Fig. 5. When calculating the cross-scale consistency loss, the model is initially run on an input image patch (576×432) in order to obtain the output feature map (left). The loss works on the principle that the binary class predictions for subsets of this feature map (right) are not necessarily the same as the binary class prediction for the entire feature map (center). In the illustrated case, although the entire patch is predicted as positive, 2 sub-patches are actually predicted as negative. Best viewed in color.

to provide additional supervision when training the model.

One possible way to accomplish this is through a variant of the "cross-scale consistency loss" proposed by Shen et al. (2018), which is tweaked slightly in order to adapt it to the MIL task. Specifically, the cross-scale consistency loss takes the following as input: the output feature map for the complete image patch (which is the output of the final 1×1 convolution shown in Fig. 3), the output feature maps for a corresponding series of evenly-spaced crops of this input patch ("sub-patches"), and the corresponding ground-truth class. The sub-patches can be any size, with or without overlap, but in these experiments, we divide the input evenly into quadrants. For speed, we acquire the sub-patch activation maps by simply extracting sub-patches from the output feature map for the full patch, as opposed to applying the model multiple times.

The loss is then calculated as

$$L_{csc} = \begin{cases} (1-\hat{y}) \sum_{i=1}^{P} \phi(y, \hat{y}_i^{patch}) & \text{if } \hat{y} \leqslant \tau \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where y denotes the ground-truth class, $\hat{y}$ denotes the output probability for the full input patch, $\hat{y}_i^{patch}$ denotes the output probability for sub-patch i, P is the number of sub-patches, and ϕ denotes the binary cross-entropy function. In (1), we make the assumption that the model outputs the probability of an example belonging to the *positive* class, e.g. a probability of zero indicates the negative class, and a probability of one indicates the positive class.

τ is a threshold that exists to prevent the loss from penalizing the model in cases where the full input patch is predicted as the positive class. For these experiments, we set it to 0.5. Intuitively, when the full input is predicted as negative, any sub-patches that are predicted as positive increase the loss, encouraging the model not to do this. The complete loss for the model is constructed as a weighted sum between standard binary cross-entropy loss and the cross-scale consistency loss

$$L = L_{bce} + \lambda L_{csc} \quad (2)$$

where λ is a weighting factor for the loss, which we determine experimentally. The standard binary cross-entropy loss is defined as follows with y and $\hat{y}$ being the ground truth and model prediction, respectively:

$$L_{bce}(y, \hat{y}) = -(y \log(\hat{y}) + (1-y)\log(1-\hat{y})) \quad (3)$$

2.6. Implementation details

All of the code for training and data preparation is written in Python. We implement the models with TensorFlow 2.4 (Abadi et al., 2015), specifically using the Keras (Chollet, 2020) frontend. We train on a Nvidia V100 GPU with 16 GB of VRAM. We also use the Kedro (Balan et al., 2020) library for managing the training and data engineering pipeline. We plan to release the code publicly as soon as any relevant materials are published. Finally, we use the open-source Computer Vision Annotation Tool (CVAT) (Sekachev et al., 3 2019.) for performing manual labeling.

2.7. Evaluation metrics

Initially, we wish to evaluate the binary classification accuracy of the models. We use the validation dataset for comparison between models. This dataset is massively unbalanced, with around 91% of the extracted patches falling into the negative class, i.e. not containing any flowers. Consequently, raw binary accuracy is not a useful metric, so we instead report the Area Under the (Receiver Operating) Curve (AUC). When calculating this specific metric, we extract a grid of non-overlapping patches, each $\frac{1}{64}$th the size of the image, and apply the model to each of them. This closely approximates the data that was used to train the model.

Table 3
Comparison of models of different depths. All models use a growth rate of 4, and differ only in the number of layers in each dense block.

# of Layers	Count MAE	AUC	Precision	Recall
15	3.34	0.84	0.93	0.96
29	**2.43**	**0.87**	**0.94**	**0.99**
58	2.97	0.82	0.93	0.97

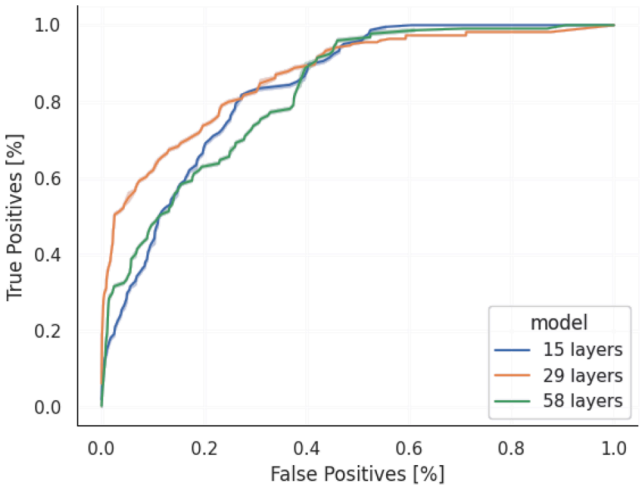


Fig. 6. ROC curves for the models shown in Table 3.

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y - \hat{y}| \quad (4)$$

In addition to the MIL task AUC, we also evaluate and report the mean absolute count error computed on the validation set images, according to Eq. 4. In contrast to the AUC calculation, inference is done as described in Section 2.4.2, with overlapping patches, similar to Cohen et al. (2017). Nominally, we use a patch size that is around 0.09 times the size of the input image, with a 66% overlap between patches in both the vertical and horizontal directions (see below for exact numbers).

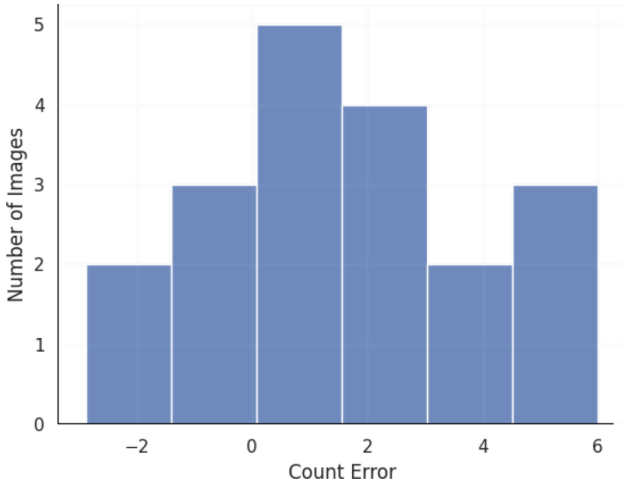


Fig. 7. Histogram of the counting error on the validation set. It can be seen that the model has a tendency towards undercounting.

Table 4
Comparison of the best-performing model trained with different values for λ, the cross-scale consistency loss weight.

λ	Count MAE	AUC
0.0	3.35	0.85
0.25	2.68	0.78
0.5	**2.43**	**0.87**
0.75	3.10	0.82
1.0	3.69	0.79

3. Results

3.1. Quantitative results

Several different DenseNet models with varying depths (Table 3) were tested. The best AUC achieved by any model is 0.87 over the validation dataset. ROC curves are shown in Fig. 6. Overall, performance is generally found to be better with increased model depth. However, at 58 layers, the model seems to start overfitting, resulting in the 29-layer model providing the best performance overall. All models deviated somewhat from what was proposed by Huang et al. (2017), including in the use of smaller convolutional layers at the bottom and in the selection of a much smaller growth rate. These changes all serve to reduce the parameter count of the models, and were determined empirically to perform well.

Across all experiments, the best mean absolute count error achieved by the model is 2.43. Though a variety of different hyperparameter configurations were tested, none produced better results. Fig. 7 shows the distribution of count errors over the validation dataset (part A2 in Table 1).

As shown in Table 4, higher and lower values of the weighting hyperparameter λ (Eqn. 2) both lead to a degradation in performance, with all else being equal. Setting too small a value for λ lowers the contribution to the overall loss of the cross-scale consistency term to the point that it becomes negligible. Setting too large a value will cause learning to focus on optimizing the consistency task to the detriment of the counting task. The value of λ is chosen empirically. Through experimentation, it was found that a value of 0.5 for λ works best.

3.2. Effects of inference hyperparameters

The patch-wise inference process used in this study has several hyperparameters that can be adjusted, including the size of the overlapping patches and the amount of overlap. In order to evaluate the influence of these settings on performance, 17 additional images with point annotations (part C in Table 1) were used to evaluate the model. Though they were collected using the same equipment and setup as the images in part A of the dataset, they were taken on August 19, 2016. Furthermore, and crucially, the density of flowers within these images is significantly higher, with a total of 353 annotated flowers over 17 frames.

The trained model was evaluated on these data with the same hyperparameters as we used before, and a counting MAE of 10.21 (Fig. 8 (b)) was achieved. In this case, the model appears to be grossly undercounting, often by as much as 50%. However, the reported AUC on these data is 0.95, which suggests that the poor counting performance is an artifact of the inference hyperparameters. Indeed, when a patch scale of 0.0625 and stride of 0.03125 are selected, the same configuration as is depicted in the penultimate row of Table 5, the overall counting MAE decreases to a much more respectable 4.12 (Fig. 8(c)).

3.3. Qualitative results

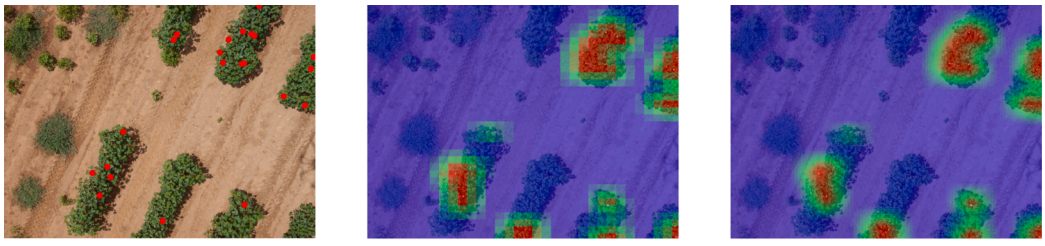
Qualitative results include some example density maps produced from four validation set images (Fig. 9). The left two columns show the two images with the lowest MAE, and the right two show the images with the highest MAE. Additionally, because the model is fully-convolutional, it is possible to visualize the activation maps directly (Fig. 10). Activation maps are extracted from the final layer of the model, just before global average pooling and sigmoid activation are applied. Sigmoid is then applied manually in order to normalize the values between 0 and 1, before overlaying the result on the input as a heatmap. The results shown in Fig. 10 indicate that the model indeed learns to recognize flowers, despite the weak supervision (e.g. image-level binary classification).

3.4. Model architecture comparison

Multiple experiments were performed with alternative model architectures, in an effort to determine the effect of the chosen architecture on model performance. The procedures for these experiments are exactly as described above, but with different model architectures. Crucially, no changes are made to any hyperparameters. Specifically, fully-convolutional extensions of the classic LeNet, AlexNet, and VGG-16 architectures are tested. These architectures were selected because they have been proven to work on a similar plant-organ-counting problem by Lu et al. (2017). Furthermore, because the data have a much larger patch size than that used by Lu et al. (2017), the fully convolutional nature of these models is advantageous, because it allows them to be applied to the cotton image dataset without modification.

Table 5
Comparison of different inference hyperparameters on the counting performance of a 29-layer model. Patch scale and patch stride are both provided in fractions of the input frame size.

Patch Scale	Patch Stride	Count MAE
0.09	0.01	**2.43**
0.09	0.03	**2.43**
0.125	0.03125	4.53
0.125	0.0625	4.49
0.0625	0.03125	4.26
0.0625	0.0625	4.21



(a) (b) (c)

Fig. 8. Example density maps generated from the 2016 data. The version with tuned hyperparameters achieves much higher counting accuracy. (a) shows the ground truth annotations, (b) shows the density map made with default hyperparameters, and (c) shows the density map made with tuned hyperparameters.

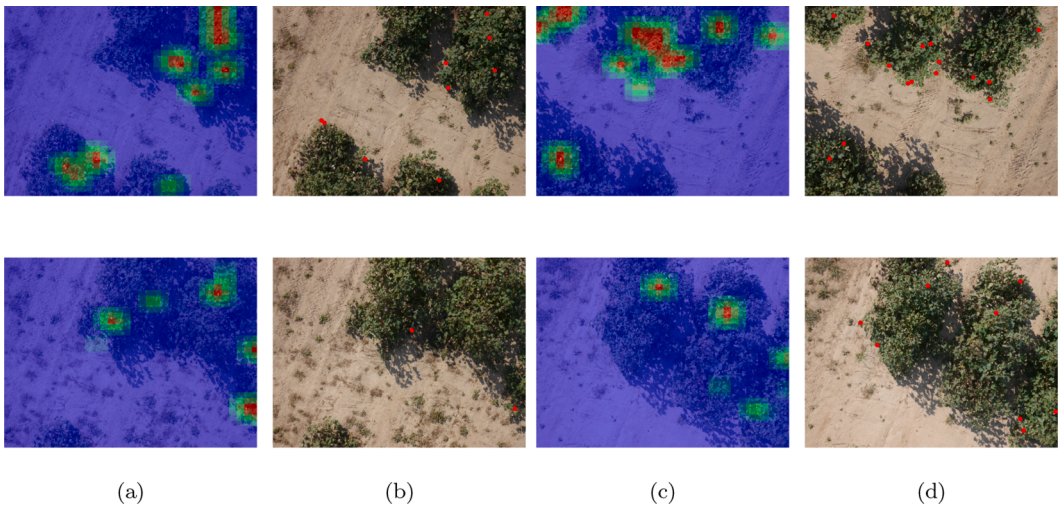


Fig. 9. Four examples of qualitative results of one of the counting models on validation data from the cotton blossom dataset. These are generated with a patch size of 0.09 times the width and height of the input, with an overlap of 0.03 times the width and height of the input. Two images in (a) show the two best-predicted examples while two images in (c) show the two worst-predicted examples. Four images in (b) and (d) are the corresponding ground truth for images in (a) and (c), respectively, with flowers marked as red dots.

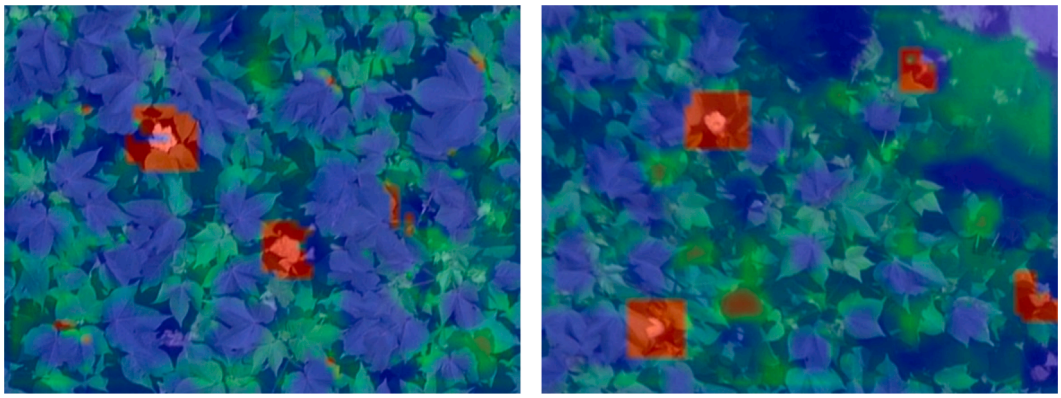


Fig. 10. Two examples of visualizations of the final activation layer of the model.

Table 6
Comparison of models with different architectures.

Architecture	Count MAE	AUC	# of Parameters
LeNet	4.60	0.76	141,153
AlexNet	3.84	0.79	247,265
VGG-16	2.90	0.82	2,357,969
DenseNet (Ours)	**2.43**	**0.87**	**54,423**

As indicated in Table 6, none of the models out-performs the best DenseNet-based model, even though the larger models do steadily better. It is interesting that the VGG-16 model is still outperformed by the DenseNet model, despite having more than 40x the number of parameters. Overall, these results are not unexpected, given the results reported by Huang et al. (2017). As explained in that paper, DenseNet derives its efficiency from re-using existing features from shallower layers in deeper layers as opposed to learning new ones from scratch. However, the proposed models are significantly smaller than even the smallest model proposed by Huang et al. (2017) (800,000 parameters), indicating that the DenseNet architecture remains powerful even when scaled down to extreme levels.

Note that these comparisons may not be entirely fair, because no optimization of hyperparameters is performed for individual architectures. Instead, sane hyperparameters are derived empirically for DenseNet, and subsequently fixed for all further experiments. Potentially, other architectures could yield improved results with better optimization.

4. Discussion

The model contains only $\sim 54,000$ parameters, and yet there remains a notable discrepancy in the MIL task AUC between the training and validation sets, which suggests some mild overfitting. Though the model is relatively small, this suggests that the approach could benefit from a larger dataset. It could be that this phenomenon is partially an effect of pulling the testing and validation data from an entirely separate session that isn't represented in the training data, as explained above. Since it is collected from various different cameras from plants in various growth stages, with a variety of settings and altitudes, the dataset exhibits a lot of inter-session variation, which makes it paramount to include data from as many sessions as possible. Dropout and regularization were also explored, but were found to not improve the performance of this model. Future experiments could include different model architectures and additional data. Additionally, the model seems to be much more prone to undercounting than overcounting, which could also be caused by the dataset, but is more likely to be the result of the quantization error caused by the MIL formulation. Any counting-by-classification approach has this problem to some degree (Liu et al.,

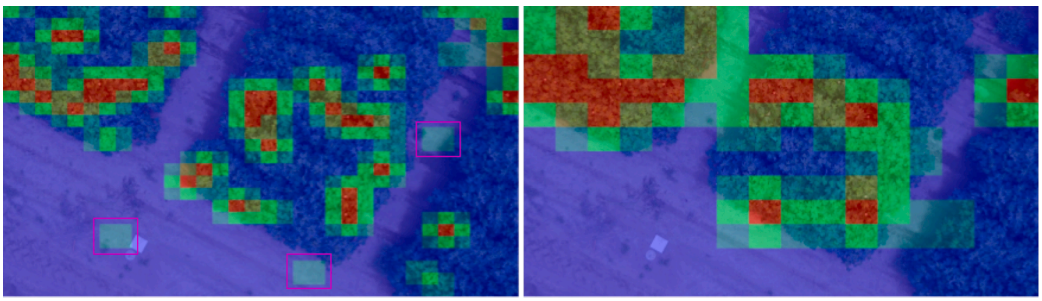


Fig. 11. A number of false positives (outlined in pink rectangles) can be seen when predicting with a smaller patch size (left) which disappear when predicting with a larger patch size (right).

2018).

It is also worth noting that our method has two important limitations. The most glaring issue is that our dataset lacks any sort of ground-truth flower count. All our metrics are derived instead by comparing the predicted count with the *annotated* count. The single aerial vantage point of our data, however, means that some flowers will inevitably remain occluded beneath the canopy, implying that the annotated count will not necessarily match the true count. Further, our data are divided into individual images, each of which shows multiple (partial) plots. This makes it fairly impractical for us to provide a per-plot counting accuracy metric, which is a particularly useful number for plant breeders. Addressing both of these limitations is a goal for future work.

For these reasons, it is hard to definitively assess whether our method performs "well enough". Xu et al. (2018) do not directly report counting errors for their method. However, their reported per-plot error range is similar to our per-*image* error range, suggesting that our approach matches or possibly even exceeds their performance. This is a good sanity-check seeing as their approach shares some training data with this one. Similarly, Jiang et al. (2020) report their best counting RMSE from their fully-supervised multi-camera ground-based method as 0.88, which is substantially better than the method proposed here, given that the latter has counting RMSE of 3.03. Since their approach is based on proximal ground imagery, however, this mainly validates the assumed trade-off between accuracy and throughput from ground and aerial systems.

4.1. Analysis of cross-scale consistency loss

With regards to the MIL task, there are two conditions that should be met for every prediction:

1. If the entire image is predicted as the positive class, there should exist at least one sub-patch that is also predicted as positive.
2. If the entire image is predicted as the negative class, then no sub-patches should be predicted as positive.

Since the model is fully convolutional, the output class is determined by taking the global average of the final feature maps and applying the sigmoid activation to the results. When considering global average pooling, it is easy to see that condition (1) is mathematically guaranteed (see appendix B for proof). However, condition (2) is not, and it is easy to find representative cases where it does not hold (Fig. 11). This phenomenon provides the opportunity to add some additional supervision without changing the annotations.

To understand intuitively why this discrepancy arises, consider that the model outputs pixel-wise predictions, which are then averaged across the entire input. If, for a given input, positive pixels are concentrated in a relatively small area, then it is possible that they will be "drowned out" by the vast preponderance of negative pixels. This could cause the entire input to be predicted as negative, even when it contains a flower. Conversely, this can also result in false-positive predictions when the input size is decreased (Fig. 11).

The cross-scale consistency loss used during training functions by penalizing violations of condition (2) above. If the entire patch is predicted as negative, the proposed loss will penalize positive predictions for any sub-patches. In this manner, it was found that cross-scale consistency loss is able to improve the performance of the model without the need for more precisely-annotated data simply by leveraging spatial intuition. This conclusion aligns well with the results of similar approaches (Bellocchio et al., 2019; Shen et al., 2018).

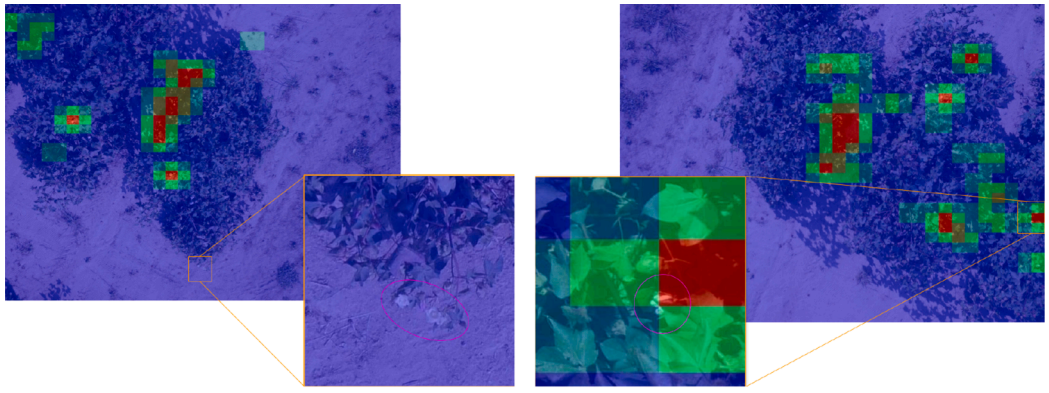


Fig. 12. Common failure cases for the model. (a) Example of the model failing to detect flowers that are growing at the edges of the plot, outside the canopy. (b) Example of the model falsely counting a partially-opened boll as a flower.

4.2. Analysis of inference hyperparameters

One disadvantage of the MIL-based approach to counting is that it introduces additional hyperparameters in the form of the patch size and amount of overlap to use during inference. As shown in Table 5, the selection of these hyperparameters does have an effect on the counting performance. There is also an effect on the inference speed, as increasing the amount of overlap significantly increases the computational cost.

Overall, these hyperparameters need to be set empirically based on data representative of the specific problem domain. Theoretically, the main limitation on patch size is that the patches should be large enough to encompass the largest flowers in the dataset. At the other end of the patch-size scale, quantization error can be mitigated through the use of larger overlaps (Cohen et al., 2017). For the experiments in this paper, a patch scale of 0.09 and stride of 0.03 times the image size were chosen, as this afforded a good balance between accuracy and performance.

In summary, it seems that to get ideal results, the inference hyperparameters must be tuned based upon the expected density of flowers in the analyzed data. Given that the flower density can vary over the course of the growing season, or even over different parts of a single field, this could cause difficulties deploying the proposed method. Needless to say, this is sub-optimal, and an indication that the overall method would likely benefit from finding a more reliable way to conduct inference.

4.3. Analysis of failure cases

Overall, the model has more of a predilection towards undercounting than overcounting. As can be seen in Fig. 12(a), it is common for the model to under-count flowers that grow outside the canopy. One theory is that this is due to the relative lack of examples of this phenomenon that exist in the training data. This problem could likely be resolved by augmenting the training set with new data that have this property. Conversely, Fig. 12(b) shows a case where the model has significantly over-counted the number of flowers. Some of these false positives appear to be triggered by partially or fully-opened bolls. In low-resolution images, bolls and flowers can sometimes be difficult to distinguish from one another, even for a human, so this behavior is not extremely surprising. Nonetheless, a significant portion of the cotton image dataset was collected earlier in the growing season and thus does not include bolls at all. It is likely that the rate of these misclassifications can be reduced by including more data from later sessions, which do contain bolls. Both of these failure cases can also be seen quite clearly in Fig. 9.

4.4. Comparison to similar work

As stated above, the proposed approach employs the redundant counting technique first proposed by Cohen et al. (2017). TasselNet (Lu et al., 2017) was possibly the first work to employ this technique in the domain of plant-organ counting, specifically applying it to maize tassels. TasselNet, however, worked by directly regressing count values, in contrast to the proposed approach, which employs the MIL task in what could be considered an extreme form of counting-by-classification. Even so, TasselNet, and the later TasselNetv2 (Xiong et al., 2019a), provide good evidence for the efficacy of the redundant counting method when applied to plant organs.

Image-based counting approaches that focus specifically on flowers have been around for decades (Adamsen et al., 2000), but even some recent approaches (Thorp et al., 2016) have stuck to classical image processing methods as opposed to applying more modern Deep Learning techniques. Very recently, Xu et al. (2018) were able to demonstrate a CNN-based system for counting cotton blooms, but the proposed model uses a relatively simplistic architecture, and relies heavily on a thresholding-based pre-segmentation step that is not always reliable. Jiang et al. (2020) also demonstrate that accurate cotton flower counts can be obtained through CNN-based detection from ground imagery, but this method has a lower throughput compared to aerial imaging and the ground vehicle may cause damage to plants. Additionally, weakly-supervised counting has been applied successfully by Bellocchio et al. (2019) within the domain of fruit counting. Their approach has many of the advantages of ours, namely reduced need for training data with complex annotations. However, the authors do not attempt to apply their approach to aerial imagery, they do not explore active learning as a means to further reduce the data annotation burden, and their approach relies on multiple models that cannot be trained end-to-end. This, along with the proposed approach, provides evidence that weakly-supervised counting is a viable strategy for a diverse set of HTP problems.

4.5. Further work

There are a variety of changes that could be made to the proposed model and training procedure which might yield better results. The accuracy and robustness of the model could doubtlessly be improved by the addition of more annotated data. Since the annotation process is relatively quick, this should be straightforward to do as more raw data are collected. This is especially true given that, although the cotton blossom dataset is large and diverse, there are many possible appearance variations that are not covered in the existing data. Breeders grow a variety of cotton cultivars, some of which have different leaf shapes and colors, as well as flowers that range in hue from white to yellow. Fields can contain a significant amount of weeds, which could confuse the detector. Also, the experiments have indicated that model performance is highly dependent on the exact architecture. Consequently, Neural Architecture Search (Elsken et al., 2019) could conceivably be expored as a method for improving the model architecture further.

In addition, there is the possibility of integrating the proposed model into a full, drone-based system for cotton flower counting throughout a growing season that encompasses data acquisition and post-processing. Xu et al. (2018) proposed the core components of such a system, and indeed, the model proposed in the current work could be used as a drop-in replacement for their original counting approach. The model can be leveraged to derive flowering progression over time and compare it with the counting results from a ground imaging system. Another caveat of this study is that most of the results are reported on a per-image or per-frame level instead of a per-plot level, as this work is mainly focused on investigating the underlying method. Nevertheless, future work should likely verify the method's ability to produce counts on a plot-by-plot level, as any practical counting system would require this feature.

5. Conclusions

This paper introduces a weakly-supervised-learning approach for counting cotton blossoms from aerial imagery, which requires simpler, less expensive annotations compared to previous approaches. Specifically, this method shows that it is possible to count sparsely-distributed objects by adapting a simple presence/absence classifier, a method which only requires weak supervision. It is also shown that this method is particularly amenable to active learning, which further facilitates the annotation process. This approach has the potential to form the basis of an efficient, practical system for counting cotton flowers in field conditions. Such a system could require nothing more than an inexpensive drone and a laptop on which to analyze data. The underlying method is also general enough to be applied to count other plant organs, particularly on other species with a similar canopy structure to cotton. The method developed in this study could help solve the flower counting problem while minimizing the data acquisition and human labeling efforts, facilitating genetic studies and contributing to crop improvement.

CRediT authorship contribution statement

Daniel Petti: Methodology, Software, Investigation, Data curation, Writing – original draft. **Changying Li:** Conceptualization, Writing – review & editing, Supervision, Project administration, Funding

Table 7
Parameters for the imagery sources used in this dataset. Note that the fifth column is the number of raw *images*, not the number of extracted patches. Ground Sample Distance (GSD) is calculated with an online tool (https://www.propelleraero.com/gsd-calculator/) using known camera parameters.

Camera	Img. Size (px)	Alt. (mAGL)	GSD (cm/px)	No. of Images	Year
Panasonic Lumix G6	4608 × 3456	32	0.23	328	2016 & 2018
DJI Phantom 4 Pro (Image)	5472 × 3648	10	0.27	70	2020
DJI Phantom 4 Pro (Video)	2704 × 1520	4	0.24	795	2018

acquisition.

Appendix A. Additional information about the dataset

The cotton image dataset was collected using a Panasonic Lumix G6 mirrorless camera mounted on a DJI Matrice 600 flying at 15 m, as well as a DJI Phantom 4 Pro, with its own built-in camera, flying at two different altitudes (see Table 7). Images were collected once every second, except for several sessions that were collected using video mode on the Phantom 4 at 30 frames per second (fps). Both drones were programmed with an automated flight path that allows for coverage of the entire field while simultaneously allowing for a small amount of overlap between adjacent images.

Appendix B. Proof of condition (1) for cross-scale consistency loss

Theorem 1. *Let S be a set where $|S| = n$. Let $P_{1\ldots d}$ define d arbitrary disjoint subsets of S, where $|P_i| = r_i$ and $P_1 \cup P_2 \cup \ldots \cup P_d = S$. Let τ be an arbitrary threshold.*

$$\text{If } \frac{\sum_{i=1}^{n} S_i}{n} > \tau, \text{ then } \exists\ i \text{ such that } \frac{\sum_{j=1}^{r_i} P_{ij}}{r_i} > \tau$$

Proof. Assume that the theorem does not hold, i.e. $\frac{\sum_{i=1}^{n} S_i}{n} > \tau$ and $\frac{\sum_{j=1}^{r_i} P_{ij}}{r_i} \leqslant \tau \forall i$. Then it follows that

$$\sum_{j=1}^{r_i} P_{ij} \leqslant r_i \tau \forall i$$

$$\sum_{i=1}^{d} \sum_{j=1}^{r_i} P_{ij} \leqslant \sum_{i=1}^{d} r_i \tau$$

$$\sum_{i=1}^{d} \sum_{j=1}^{r_i} P_{ij} \leqslant \tau \sum_{i=1}^{d} r_i$$

$$\sum_{i=1}^{d} \sum_{j=1}^{r_i} P_{ij} \leqslant n\tau$$

$$\sum_{i=1}^{n} S_i \leqslant n\tau$$

$$\frac{\sum_{i=1}^{n} S_i}{n} \leqslant \tau$$

which is a contradiction. □

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper

Acknowledgment

This work was supported by the National Science Foundation Growing Convergence Research (Award No. 1934481) and Georgia Cotton Commission. The authors gratefully thank Rui Xu for his assistance in field data collection.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. URL: https://www.tensorflow.org/.

Adamsen, F., Coffelt, T., Nelson, J.M., Barnes, E.M., Rice, R.C., 2000. Method for using images from a color digital camera to estimate flower number. Crop Sci. 40 (3), 704–709. URL https://acsess.onlinelibrary.wiley.com/doi/abs/10.2135/cropsci2000.403704x.

Akiva, P., Dana, K., Oudemans, P., Mars, M., 2020. Finding berries: Segmentation and counting of cranberries using point supervision and shape priors. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 219–228.

Bellocchio, E., Ciarfuglia, T.A., Costante, G., Valigi, P., 2019. Weakly supervised fruit counting for yield estimation using spatial consistency. IEEE Robot. Automat. Lett. 4 (3), 2348–2355.

Balan, L., (Kiyo), K.K., Deriabin, D., Hoang, L., Ivaniuk, A., Dada, Y., Datta, D., Patel, Z., Wrigley, G., Danov, I., Stichbury, J., Khan, N., Tsaousis, N., Theisen, M., Walker, W., Nguyen, T., Westenra, R., Carvalho, L., Trevisani, M.D., Bertoli, S., Mawjee, S., sasaki takeru, Nijholt, B., Vukolov, D., Fischer, K., Vijaykumar, Minami, Y., bru5, dr3s, Dec. 2020. quantumblacklabs/kedro: 0.17.0. URL: https://doi.org/10.5281/zenodo.4336685.

Bollis, E., Pedrini, H., Avila, S., 6 2020. Weakly supervised learning guided by activation mapping applied to a novel citrus pest benchmark. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 310–319.

Chawade, A., van Ham, J., Blomquist, H., Bagge, O., Alexandersson, E., Ortiz, R., 2019. High-throughput field-phenotyping tools for plant breeding and precision agriculture. Agronomy 9 (5), 258.

Chen, S.W., Shivakumar, S.S., Dcunha, S., Das, J., Okon, E., Qu, C., Taylor, C.J., Kumar, V., 2017. Counting apples and oranges with deep learning: A data-driven approach. IEEE Robot. Automat. Lett. 2 (2), 781–788.

Chollet, F., 2020. Keras: Simple, powerful, flexible. URL: https://keras.io/.

Cohen, J., Boucher, G., Glastonbury, C.A., Lo, H.Z., Bengio, Y., 10 2017. Count-ception: Counting by fully convolutional redundant counting. In: 2017 IEEE International Conference on Computer Vision Workshop (ICCVW). IEEE Computer Society, Los Alamitos, CA, USA, pp. 18–26. URL: https://doi.ieeecomputersociety.org/10.1109/ICCVW.2017.9.

Elsken, T., Metzen, J.H., Hutter, F., 2019. Neural architecture search: A survey. J. Machine Learn. Res. 20 (1), 1997–2017.

Foulds, J., Frank, E., 2010. A review of multi-instance learning assumptions. Knowledge Eng. Rev. 25.

Ghosal, S., Zheng, B., Chapman, S.C., Potgieter, A.B., Jordan, D.R., Wang, X., Singh, A.K., Singh, A., Hirafuji, M., Ninomiya, S., Ganapathysubramanian, B., Sarkar, S., Guo, W., 2019. A weakly supervised deep learning framework for sorghum head detection and counting. Plant Phenomics 2019, 1525874. URL https://doi.org/10.34133/2019/1525874.

Giuffrida, M.V., Dobrescu, A., Doerner, P., Tsaftaris, S.A., Leaf counting without annotations using adversarial unsupervised domain adaptation. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 2590–2599.

Guo, W., Zheng, B., Potgieter, A., Diot, J., Watanabe, K., Noshita, K., Jordan, D., Wang, X., Watson, J., Ninomiya, S., Chapman, S., 2018. Aerial imagery analysis – quantifying appearance and number of sorghum heads for applications in breeding and agronomy. Front. Plant Sci. 9, 1544.

Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q., 7 2017. Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Bach, F., Blei, D. (Eds.), Proceedings of the 32nd International Conference on Machine Learning. Vol. 37 of Proceedings of Machine Learning Research. PMLR, Lille, France, pp. 448–456. URL: http://proceedings.mlr.press/v37/ioffe15.html.

Jiang, Y., Li, C., Xu, R., Sun, S., Robertson, J.S., Paterson, A.H., 2020. Deepflower: a deep learning-based approach to characterize flowering patterns of cotton plants in the field. Plant Methods 16, 156. URL: https://doi.org/10.1186/s13007-020-00698-y.

Laradji, I.H., Rostamzadeh, N., Pinheiro, P.O., Vazquez, D., Schmidt, M., 9 2018. Where are the blobs: Counting by localization with point supervision. In: The European Conference on Computer Vision (ECCV).

Liu, L., Lu, H., Xiong, H., Xian, K., Cao, Z., Shen, C., 2019. Counting objects by blockwise classification. IEEE Trans. Circuits Syst. Video Technol. 1–1.

Liu, X., van de Weijer, J., Bagdanov, A.D., 6 2018. Leveraging unlabeled data for crowd counting by learning to rank. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Loshchilov, I., Hutter, F., 2016. Sgdr: Stochastic gradient descent with warm restarts, arXiv:1608.03983. URL: https://ui.adsabs.harvard.edu/abs/2016arXiv160803983L.

Lu, H., Cao, Z., Xiao, Y., Zhuang, B., Shen, C., 2017. Tasselnet: counting maize tassels in the wild via local counts regression network. Plant Methods 13.

Madec, S., Jin, X., Lu, H., de Solan, B., Liu, S., Duyme, F., Heritier, E., Frederic, B., 2019. Ear density estimation from high resolution rgb imagery using deep learning technique. Agric. For. Meteorol. 264, 225–234.

Olmschenk, G., Chen, J., Tang, H., Zhu, Z., 2019. Dense crowd counting convolutional neural networks with minimal data using semi-supervised dual-goal generative adversarial networks. Journal Name: IEEE Conference on Computer Vision and Pattern Recognition: Learning with Imperfect Data Workshop, Medium: X; Size: 21–28.

Oosterhuis, D.M., 1990. Growth and development of a cotton plant. Nitrogen nutrition of cotton: Practical issues, 1–24.

Rahnemoonfar, M., Sheppard, C., 2017. Deep count: Fruit counting based on deep simulated learning. Sensors 17 (4). URL https://www.mdpi.com/1424-8220/17/4/905.

Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In: Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., Garnett, R. (Eds.), Advances in Neural Information Processing Systems. Vol. 28. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

Sam, D., Sajjan, N., Maurya, H., Babu, R., 2019. Almost unsupervised learning for dense crowd counting. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 8868–8875.

Sekachev, B., Zhavoronkov, A., Manovich, N., 3 2019. Computer vision annotation tool: A universal approach to data annotation. Tech. rep. URL: https://software.intel.com/content/www/us/en/develop/articles/computer-vision-annotation-tool-a-universal-approach-to-data-annotation.html.

Settles, B., 2009. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

Shen, Z., Xu, Y., Ni, B., Wang, M., Hu, J., Yang, X., 2018. Crowd counting via adversarial cross-scale consistency pursuit. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5245–5254.

Sun, K., Wang, X., Liu, S., Liu, C., 2021. Apple, peach, and pear flower detection using semantic segmentation network and shape constraint level set. Comput. Electron. Agric. 185, 106150. URL https://www.sciencedirect.com/science/article/pii/S016816992100168X.

Thorp, K., Wang, G., Badaruddin, M., Bronson, K., 2016. Lesquerella seed yield estimation using color image segmentation to track flowering dynamics in response to variable water and nitrogen management. Ind. Crops Prod. 86, 186–195. URL http://www.sciencedirect.com/science/article/pii/S0926669016301819.

Ubbens, J.R., Ayalew, T.W., Shirtliffe, S., Josuttes, A., Pozniak, C., Stavness, I., 2020. Autocount: Unsupervised segmentation and counting of organs in field images. In: Bartoli, A., Fusiello, A. (Eds.), Computer Vision – ECCV 2020 Workshops. Springer International Publishing, Cham, pp. 391–399.

Xiong, H., Cao, Z.-G., Lu, H., Madec, S., Liu, L., Shen, C., 2019a. Tasselnetv2: in-field counting of wheat spikes with context-augmented local regression networks. Plant Methods 15.

Xiong, H., Lu, H., Liu, C., Liu, L., Cao, Z., Shen, C., 10 2019b. From open set to closed set: Counting objects by spatial divide-and-conquer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).

Xu, R., Li, C., Paterson, A.H., Jiang, Y., Sun, S., Robertson, J.S., 2018. Aerial images and convolutional neural network for cotton bloom detection. Front. Plant Sci. 8, 2235. URL https://www.frontiersin.org/article/10.3389/fpls.2017.02235.