IEEE ROBOTICS & AUTOMATION LETTERS
A PUBLICATION OF THE IEEE ROBOTICS & AUTOMATION SOCIETY

IEEE Xplore® Digital Library
http://ieeexplore.ieee.org/Xplore

Counting Apples and Oranges with Deep Learning: A Data Driven Approach

Steven W Chen[1], Shreyas S. Shivakumar[1], Sandeep Dcunha[2], Jnaneshwar Das[1], Edidiong Okon[1], Chao Qu[1], Camillo J. Taylor[1], and Vijay Kumar[1]

***Abstract*—This paper describes a fruit counting pipeline based on deep learning that accurately counts fruit in unstructured environments. Obtaining reliable fruit counts is challenging because of variations in appearance due to illumination changes and occlusions from foliage and neighboring fruits. We propose a novel approach that uses deep learning to map from input images to total fruit counts. The pipeline utilizes a custom crowd-sourcing platform to quickly label large data sets. A blob detector based on a fully convolutional network extracts candidate regions in the images. A counting algorithm based on a second convolutional network then estimates the number of fruit in each region. Finally, a linear regression model maps that fruit count estimate to a final fruit count. We analyze the performance of the pipeline on two distinct data sets of oranges in daylight, and green apples at night, utilizing human generated labels as ground truth. We also show that the pipeline has a short training time and performs well with a limited data set size. Our method generalizes across both data sets and is able to perform well even on highly occluded fruits that are challenging for human labelers to annotate.**

***Index Terms*—Agricultural Automation; Object detection, segmentation, categorization; Visual Learning**

I. Introduction

FRUIT counting is an important task for growers to estimate yield and manage orchards. An accurate automated fruit detection and counting algorithm gives agricultural enterprises the ability to optimize and streamline their harvest process. Through a better understanding of the variability of yield across their farmlands, growers can make more informed and cost-effective decisions for labor allotment, storage, packaging, and transportation. Smart sensor suites as well as autonomous robots such as unmanned aerial vehicles (UAVs) will benefit from data-driven fruit counting algorithms that enable growers to estimate yield at scale. For example, Figure 1 shows a UAV with onboard cameras that can be used for rapid estimation of yield by flying between rows of trees [1].

Estimation of fruit count from images is a challenging task for a number of reasons including appearance variability due to illumination, and occlusion due to surrounding foliage and fruits. Previous fruit counting algorithms relied on traditional computer vision methods involving hand-crafted features that exploited the shape, color, texture or spatial orientation of various fruit [2]. While these methods work well under specific conditions, they are usually fruit specific, require careful control of the environment, and cannot handle heavily occluded fruits.

An additional challenge to fruit counting that is not present in fruit detection is distinguishing multiple overlapping fruits. Most algorithms either ignore this problem, or use simple heuristics based on shape and size, which do not generalize to natural settings that feature heavy occlusion and high variability of depth in fruit location [3].

Fig. 1: UAVs can fly between rows of trees autonomously and produce fruit counts at scale, from onboard camera imagery.

Deep learning is a natural choice to deal with the unstructured environments that traditional computer vision methods have difficulty with [4]. This paper presents a novel pipeline that accurately estimates counts across different fruit types, illumination, and occlusion levels. The broad steps of the pipeline are:

- collect human-generated labels from a set of fruit images;
- train a blob detection fully convolutional network to perform image segmentation;
- train a count convolutional network to take the segmented image and output an intermediate estimate of the fruit count; and
- train a linear regression to map intermediate fruit count estimates to final counts using human-generated labels as ground truth.

Figure 2 displays the output of the blob detection network and Figure 3 displays the output of the count neural network.

Our goal is to provide a data-driven fruit counting methodology that an agricultural enterprise can easily adapt and apply in unstructured farm settings. As a result, the proposed method emphasizes labeling and training speed, generalizability, and accuracy.

The main contributions of this paper are:

- a labeling platform that easily scales to large amounts of data;

Manuscript received: September, 10, 2016; Revised December, 7, 2016; Accepted December, 26, 2016.

This paper was recommended for publication by Editor Wan Kyun Chung upon evaluation of the Associate Editor and Reviewers' comments. This work is supported by USDA grant 2015-67021-23857 under the National Robotics Initiative, and NSF grant CNS-1521617.

[1]These authors are with GRASP Lab, University of Pennsylvania, Philadelphia, PA 19104, USA, {chenste, sshreyas, djnan, quchao, cjtaylor, kumar}@seas.upenn.edu.

[2]This author is with University of Massachusetts Amherst, Amherst, MA 01003, USA, {sdcunha@umass.edu}.

Digital Object Identifier (DOI): see top of this page.

Digital Object Identifier 10.1109/LRA.2017.2651944

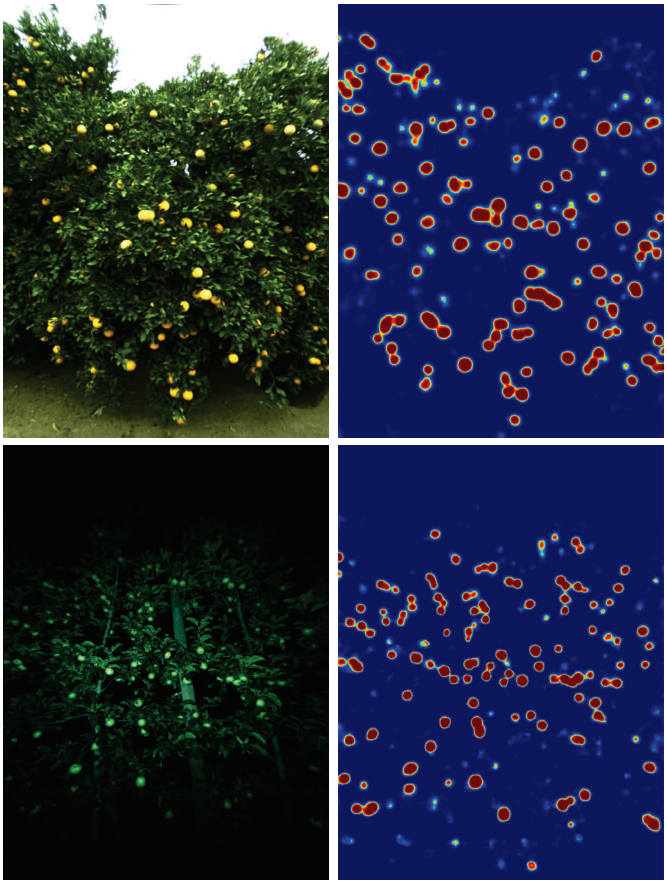

Fig. 2: ***Top left***: Example orange image; the orange data set features high levels of occlusion and variable depth in fruit locations. ***Top right***: Orange blob detection results ***Bottom left***: Example apple image; the apple data set features color similarity between fruit and foliage ***Bottom right***: Apple blob detection results

- a novel and efficient representation of human-generated labels utilizing scalable vector graphics (SVG) to simultaneously store both the location and the number of fruits;
- the application of a fully convolutional network to accurately detect blobs of fruits; and
- the application of a convolutional network to accurately count the number of fruit within each segmented blob.

The rest of the paper is organized as follows: 1) a summary of related work in fruit detection; 2) a formulation of the problem; 3) description of the proposed approach; 4) results and analysis; and 5) conclusion.

II. Related Work

Jimenez et al. provide a comprehensive survey of computer vision methods for fruit detection [2] from it's first mention in [5] and [6]. A compilation of more recent works describes the current state of the art in fruit detection [7]. Previous attempts at automating fruit counting relied on applying computer vision techniques, hand crafted for specific fruit by exploiting their unique attributes such as color [8][9][10][11][12][13][14][15][16], texture [8][13][14][16][17], shape [6][8][10][11][12][18] and spatial orientation [14] to separate fruit from background foliage. The drawbacks of relying on hand-engineered features are that they do not handle occlusion well and do not generalize to a variety of conditions and fruits. Our pipeline addresses these weaknesses by learning these features through deep learning techniques rather than hand-designing them.

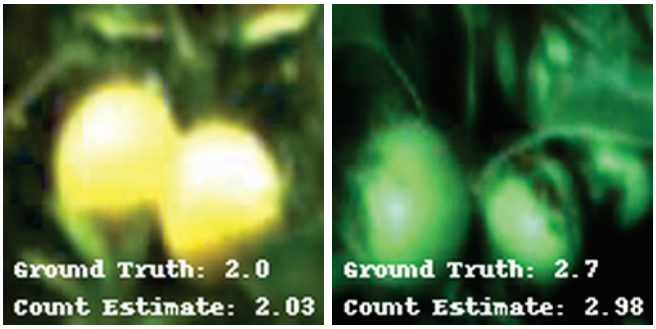

Fig. 3: ***Left***: Orange count estimate output by count neural network and regression. ***Right***: Apple count estimate output. The count neural network identifies the number of fruit in each blob to get accurate estimates of total fruit count.

Nuske et al. present a method of grape cluster counting using a radial symmetry transform to identify candidate berry locations followed by a k-Nearest Neighbor learning algorithm for final grape detection [19]. Wang et al. present a method of estimating apple yield using hue thresholding followed by the exploitation of the specular reflectance [3] characteristics of controlled artificial illumination to detect fruit [20]. These methods only work well on night time data sets since they control illumination to capture the specular features. Our pipeline generalizes to both night and day time data sets where the environment is not carefully controlled.

Recent work applying deep learning to agriculture involved the classification of crops from weeds and soil using a fully convolutional network (FCN) [4] using a large repository of synthetic data [21]. Mortensen et al. used the same modified VGG-16 deep neural network in the identification of crop such as barley and radish [22]. While our pipeline similarly uses the FCN architecture for our blob detector, it goes beyond these approaches by addressing the issue of overlapped fruit and treating fruit counting as a counting problem instead of a simple pixel-wise classification problem. We approach the counting problem using a second neural network and a linear regression to count the number of fruit within each blob detected by the FCN.

III. Problem Formulation

Consider a set of images x_i for $i = 1 \dots n$. Each image x_i has an unobservable state $z_i \in \mathbb{N}$ representing the actual number of fruit in the image x_i. Let $\tilde{z}_i \in \mathbb{R}_0^+$ be the human-generated ground truth count estimate of z_i.

Problem (Fruit Counting). Choose a function $f(x_i) \in \mathbb{R}_0^+$ representing the algorithm-generated count estimate of $\tilde{z}_i$ which minimizes the l^2 error:

$$\sqrt{\sum_{i=1}^{n}(f(x_i) - \tilde{z}_i)^2} \tag{1}$$

Notice that the l^2 error depends on the human-generated ground truth count, not the actual count.

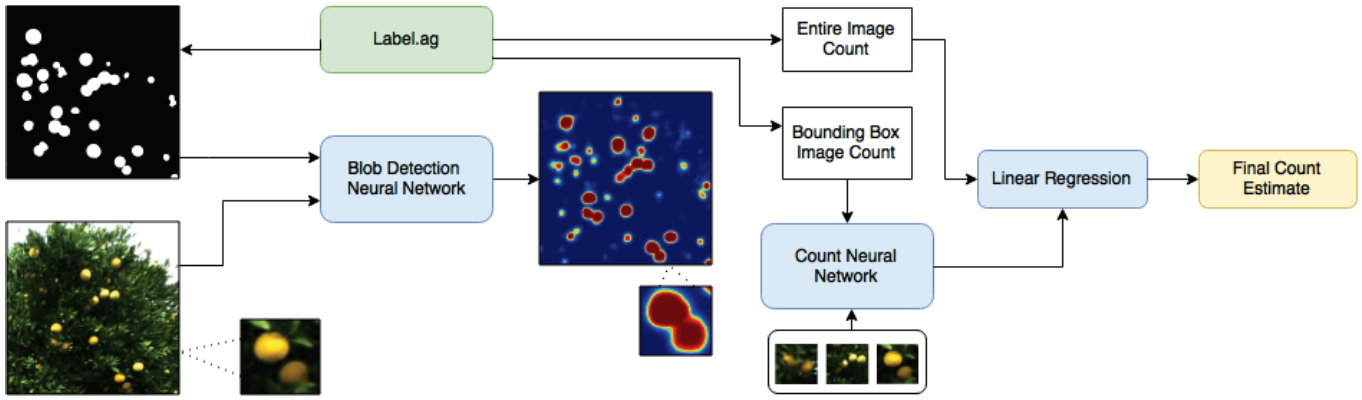

Fig. 4: The training pipeline starts with a given image. Label.ag then produces the corresponding ground truth *label map*, and these two inputs are used to train the **blob detection neural network**. This neural network outputs a segmented image, and the pipeline extracts the coordinates of the bounding boxes around each blob. These coordinates are then used to extract the corresponding window in the original image. Label.ag produces the corresponding ground truth *counts* for each bounding box, and these are used as inputs to train the **count neural network**. The count neural network then estimates and sums up the count for each blob in the segmented image to produce an intermediate count estimate. The intermediate count estimate is **regressed** on the entire image ground truth count provided by label.ag to produce the final count estimate.

IV. Proposed Approach

The proposed approach utilizes a pipeline of deep learning algorithms to detect and count fruit in unstructured environments (Figure 4). *Part 0* of the proposed approach uses an online crowd-sourcing labeling platform for rapid ground-truth label generation. *Part 1* performs blob detection using a fully convolutional network [4] to segment potential fruit clusters from the background. *Part 2* estimates the number of fruit in each blob using a convolutional network. *Part 3* runs the first and second part of the algorithm on each training image to get a count estimate. It then performs a linear regression of the count estimate on the ground truth count.

A. Part 0: Obtaining Ground Truth

We have developed a special purpose web-based labeling framework called label.ag to quickly collect and store ground truth labels from a team of human labelers. The system is designed to collect these labels in the vector based Scalable Vector Graphics (SVG) format. Instead of providing a pixel-based labeling of the image, the users generate SVG data by drawing circles around fruit in the image. The locations and radii of these circles are then stored for analysis and training.

Accuracy of labels: Presenting an entire 1280×960 image to a user containing over 100 fruit to label is a daunting task. To reduce the cognitive load on the users, each orange image is subdivided into 16 320×240 windows (apple images are subdivided into 16 480×300 windows) and these tiles are presented instead. These tiles are automatically padded to provide sufficient context for fruit at the edges. Each window is presented to 3 different users to increase the diversity of the labels. We had 22 users label approximately 5000 image tiles over the span of two weeks. Our final data set is 71 1280×960 orange images and 21 1920×1200 apple images.

Efficiency of labels: Traditionally, image labels are stored as a matrix where each pixel entry is assigned a class label. This format ignores information about total count and the distinction between the fruits. While this approach is sufficient for image segmentation, it is a problem for counting which depends on the information thrown away. The proposed approach thus increases the *efficiency* of the labels by storing them in a SVG format that preserves location, size, and number of label circles, allowing us to distinguish each individual marking. These SVG labels can then be converted into the desired training images at each part of the proposed pipeline. The blob detection component uses the data to train pixel-level classifiers while the counting component uses the number of user markings to discriminate overlapping fruit.

Preparing labels for blob detector: Converting SVG labels to create a label matrix is relatively straightforward. Each pixel is treated independently and averaged across all window tiles and labelers to determine how often that pixel is labeled as fruit. Pixels that have been labeled as fruit more than 50% of the time are classified as fruit.

Preparing labels for counting The count neural network needs to give bounding box coordinates and receive a ground truth fruit count in that bounding box. Obtaining these ground truth counts is not straightforward. First, the labels are averaged over *fruit*, and unlike a pixel, a fruit has no clear correspondence between labelers since the specific location of the SVG label will vary. Second, since we presented smaller overlapped tiles of the full-sized images in order to improve label accuracy, we have to establish a correspondence between tiles.

The proposed approach uses Alg. 1 to construct ground truth counts. The intuition of the algorithm is as follows: Consider a bounding box around a blob returned by our blob detector. The true unobservable count z_i is that there is 1 fruit in the blob, and we need to get the human generated ground truth $\tilde{z}_i$ from our SVG string. This full-sized image has been labeled by 3 different labelers, thus out of the total 48 presented window tiles, every center inside the bounding box has appeared in 3 tiles. To complicate things, since the fruit is difficult to see, only 2 out of 3 labelers labeled it. The algorithm proceeds as follows. It finds 2 centers in the SVG string that lie inside the bounding box. For each center, it finds that that center has

appeared in 3 windows. As a result it adds up $\frac{1}{3} + \frac{1}{3}$ to get a total ground truth estimate of $\frac{2}{3}$ fruit inside the bounding box. This algorithm works for the various potential configurations between multiple labelers, window tiles and fruit number.

Algorithm 1 Ground Truth Count

```
procedure GETCOUNT(Box, Centers, Windows)
    count ← 0
    c ← center in Centers that lie in Box
    for center in c do
        w ← window in Windows that contain center
        n ← 0
        for window in w do
            n ← n + 1
        count ← count + 1/n
    return count
```

B. Part 1: Blob Detection Neural Network

The blob detector is a fully convolutional network that takes in an image of size $h \times w \times 3$, and outputs a score tensor of size $h \times w \times n$ where n is the number of object classes. Each element x_{ijk} represents the score of pixel at spatial location (i, j) for class k. The probability of a pixel being in class k is obtained by feeding the scores of that pixel for each class through a softmax function.

The blob detection network has the same architecture as the original fully convolutional network in [4], except the output is only 2 classes instead of the original 21 classes. Utilizing the same architecture allows the proposed approach to perform *net surgery* or *finetuning* [23], a technique which initializes weights from the original FCN as opposed to random initialization, and greatly speeds up the training process. The original fully convolutional network was in turn initialized from the weights of the VGG network [24]. The fully convolutional network does not have any fully connected layers. The benefit of having all fully convolutional layers is that the neural network can take in images of any size. This flexibility means that the network can be trained on images of one size, and applied on images of another size. Figure 5 displays a segmentation output of the blob detection network on the orange data set.

C. Part 2: Count Neural Network

The presence of overlapped fruit in a cluttered environment means that each detected blob can contain multiple fruit. This overlap may not be a problem in minimizing the pixel-wise error, but it is a problem in obtaining accurate fruit counts. The proposed approach solves this problem by using a convolutional neural network that takes in a bounding box around each blob and outputs the estimate of the count of fruit inside the box.

The count network has the same architecture as the blob detection network, except that it does not include any of the deconvolutional layers and the output is a single number as opposed to a vector. The loss function is an l^2 loss. This similar architecture allows the count network to be initialized from the features of the blob detection network using the same technique of *finetuning* and further speeding up the training process.

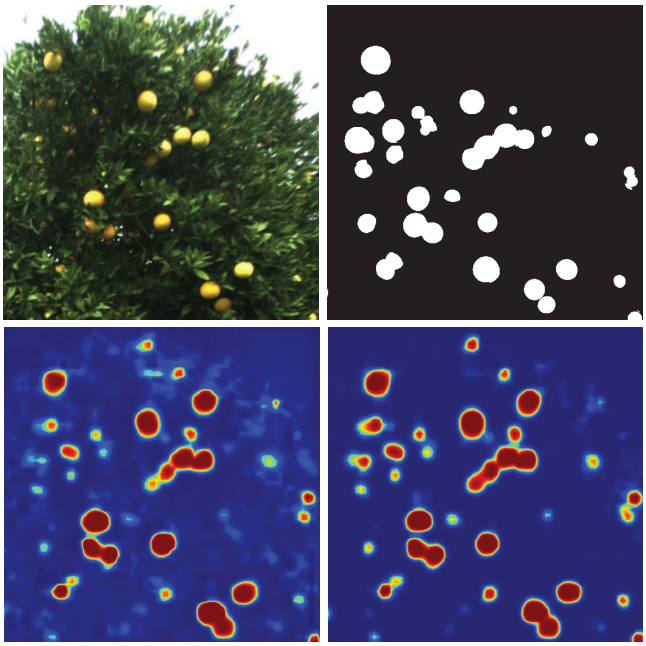

Fig. 5: ***Top left***: Original image; ***Top right***: Ground truth label; Notice the frequent occurrence of overlapped fruit. ***Bottom left***: Blob detector after $2,500$ iterations; ***Bottom right***: Blob detector after $50,000$ iterations. The segmentation is cleaner and more accurate after $50,000$ iterations, but the segmentation at $2,500$ is still comparable.

The first step is to create training data for the count network. The proposed approach runs the blob detection network on all the training images to produce segmented image outputs. It then takes these image outputs and obtains bounding boxes around each blob in each image. Next it takes these bounding boxes and extracts the corresponding window from the original image. The extracted windows are then re-sized to a common 128×128 spatial dimension. All of these extracted windows are grouped together to constitute the training set for the count network.

The second step is to create the associated training ground truth counts for the count network. Using the bounding boxes obtained from the previous step, the proposed approach calls Alg. 1 to obtain the ground truth count for each bounding box. This step has thus associated each window in the count training set with the ground truth estimate of the actual count. The automatically generated data set is then used to train the count network.

Even though the count network is initialized from the blob network, it is able to improve over the blob network because it is optimizing a completely different loss function. The task of image classification, which our fruit counting task is similar to, is much simpler than the task of image segmentation since the network is not required to output locations. Approaching the count problem of large fruit images using only one network is extremely complex and difficult due to the large variations in appearance and count. The application of two neural networks is thus a way to simplify the problem into two modular sub-problems.

D. Part 3: Final Count Linear Regression

The function between the the human-generated ground truth count and the intermediate output obtained by applying the blob network and count network is:

$$\tilde{z}_i = g(\hat{z}_i) = g\left(\sum_{j=1}^{b_i} c_j\right) \tag{2}$$

where $\tilde{z}_i$ is the human-generated ground truth count, $\hat{z}_i$ is the intermediate count estimate output, b_i is the number of blobs in image i output by the blob network, and c_j is the estimated count of blob j output by the count network.

Using $\hat{z}_i$ as an estimate of $\tilde{z}_i$ therefore assumes that the function g is the identity function. This assumption is not optimal because both the blob network and the count network never minimized a loss function that directly depended on the human-generated ground truth estimate $\tilde{z}_i$.

The final step is thus a linear regression minimizing the following loss function:

$$\alpha, \beta = \arg\min_{\alpha,\beta} \sum_{i=1}^{n} \left[\frac{\tilde{z}_i}{b_i} - \left(\alpha \frac{\hat{z}}{b_i} + \beta\right)\right]^2 \tag{3}$$

Note that applying the linear regression on the counts adjusted by b_i adds an additional non-linearity since the function mapping the intermediate fruit count estimates to the number of blobs is non-linear.

V. Results and Analysis

This section presents the results of the proposed approach on the orange and apple data sets. We chose these data sets in order to demonstrate the proposed approach's ability to generalize since the two data sets vary greatly in terms of lighting, occlusion, and overall environment. We use Caffe [25] for our deep learning models and train on an NVIDIA Titan X GPU.

We present the results of the proposed algorithm in both traditional pixel-wise accuracy measures and the more challenging count-based measures. We then present an empirical analysis of the effect of training time and training data set size to gauge how quickly and easily a fruit grower can apply the algorithm in the real world. Finally we present a few instances when the algorithm has outperformed the human labelers.

Description of Dataset: We purposefully analyze the performance of our pipeline on two datasets that differ in lighting condition, occlusion levels, resolution and camera type in order to demonstrate that our method generalizes well across different conditions. We view this generalizability as one of the main strengths of this approach. We collected the orange fruit data set during the day with no artificial lighting at Booth Ranch LLC in California. The orange trees were in a non-trellis arrangement. We acquire images of size 1280×960 using a Bluefox USB 2 camera at 10Hz. We collected the apple fruit data set at night using an external flash setup at Washington State. The apple trees were in a trellis arrangement. We acquired images of size 1920×1200 using a PointGrey USB 3 camera at 6Hz. The orange images were collected with our sensor package mounted on a steady cam and carried by human operator at walking speed. The apple images were collected using a utility vehicle driving down the row at around $1\ m/s$.

The orange data set is a challenging data set due to its non-trellis arrangement and daytime image. As a result there are higher levels of occlusion, more clusters of fruit, more variation in depth, and uncontrolled illumination – all characteristics of fruit in nature. The apple data set is challenging due to color similarity between the apples and the foliage, however it features less occlusion, depth variation, and has controlled illumination.

Description of Metrics: We evaluate pixel-wise accuracy using metrics standard to common semantic segmentation tasks such as mean Intersection over Union (IU) as well as true positive and false positive rates to generate Receiver Operating Characteristic (ROC) curves. Figure 6 depicts ROC and mean IU curves across varying probability threshold values for the apple data set. The Intersection over Union metric, also known as the Jaccard index, is a measure of similarity between two segmented areas – in our case ground truth labels and model predictions. It is the ratio of the number of pixels present in both segmentations (intersection) to the total number of pixels in the segmentations (union). It is preferable to accuracy measures because it is a ratio and controls for the total number of pixels in each class. Let n_{ij} be the pixels in class i predicted to belong to class j, where there are n_{cl} classes. In our example we have 2 classes of fruit and non-fruit. Let t_i be the total number of pixels of class i. Mean IU is defined:

$$\left(\frac{1}{n_{cl}}\right) \sum_i \frac{t_i n_{ii}}{t_i + \sum_j n_{ji} - n_{ii}} \tag{4}$$

We evaluate count accuracy using an l^2 error, mean error in the form of ratio of total fruit counted, and standard deviation of the errors. Recall from the problem formulation that the main metric is the l^2 norm. While a simple accuracy, or mean error, is appealing because it is easy to interpret, it is misleading because a model can have high accuracy over the entire data set, but low accuracy per image since the errors will wash each other out. In addition, accuracy will get better as the data set size increases, which is not a desirable property. Standard deviation will capture this variation in error, but it is also lacking since it does not capture accuracy. Using the l^2 norm takes into account both the mean and standard deviation of error, so it is the metric of choice.

A. Results

The batch size for the blob detection neural network is 1 image because each pixel is treated as a separate training example. The batch size for the count neural network is 24 128×128 images. The results of a texture-based fruit detection algorithm [3] are also presented as comparison to our method in the case of nighttime apples. Since the texture-based algorithm relies on controlled illumination, we do not compare it to our daytime orange results.

Oranges: There are a total of $7,200$ oranges over 71 images giving on average 102 oranges per image. We sequentially partition the data set into a training set of 36 full-sized images and a testing set of 35 full-sized images in order to prevent the

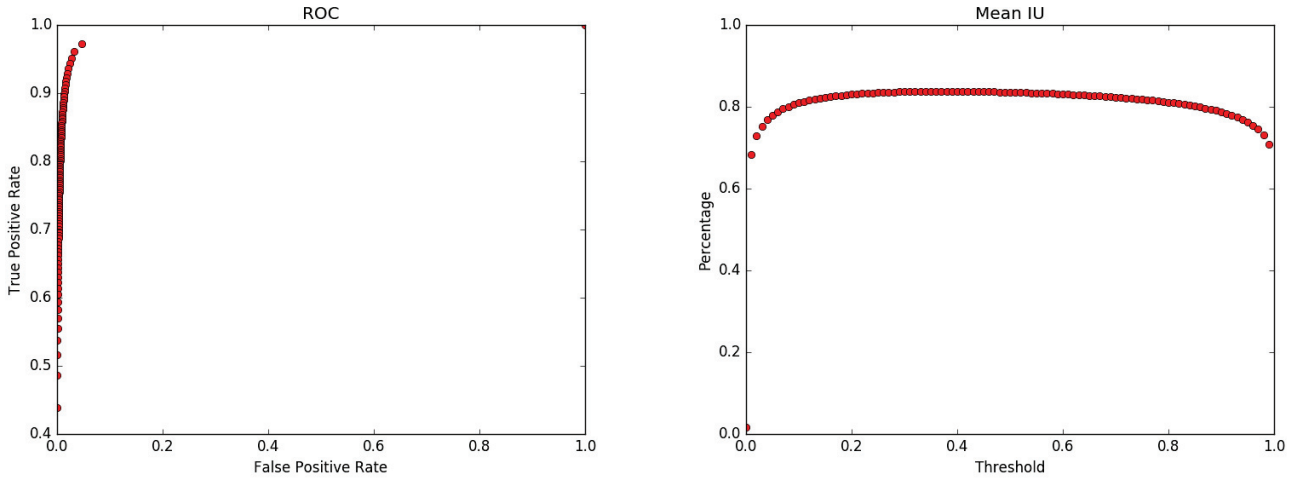

Fig. 6: ***Left***: Apple ROC; ***Right***: Apple Mean Intersection over Union; The best ROC value occurs at threshold 0.02 with a true positive rate of 0.961 and a false positive rate of 0.033. The best mean IU is 0.838 at the 0.37 threshold. A perfect ROC point is at the top-left corner (0% false positive rate, 100% true positive rate). A perfect mean IU is at 1.0.

same tree being present in both the train and test set. Each full sized 1280×960 image in the training set is partitioned into 100 randomly cropped and flipped 320×240 sub-images as a form of data augmentation. Testing is done on the original full-sized image.

We evaluate pixel-wise metrics of the blob detection neural network trained for $50,000$ iterations. The blob detection network outputs a probability map, and we vary the cutoff threshold to create a binary map. The threshold value for the best ROC is 0.03, yielding a true positive rate of 0.957 and false positive rate of 0.051. The best mean IU is 0.813 at the 0.38 threshold value.

We next evaluate count-based metrics of the entire pipeline using a blob detection network trained for $50,000$ iterations and a count network trained for $25,000$ iterations. The threshold value for the blob detection network was set at 0.03. This model achieves an l^2 error of 13.8 corresponding to a ratio of 0.968 total oranges counted with a standard deviation of 13.5 oranges across the entire test set.

In order to quantify the effect of each component of the pipeline, we also ran the algorithm using just the blob detection output where each blob represents 1 fruit, a blob detection with only a count network, and a blob detection with only linear regression. Figure 7 displays the counting metrics for the oranges.

Orange Model	l^2 error	Ratio Counted	Std Dev
blob	16.9	0.935	15.6
blob+regression	15.9	0.999	15.9
blob+count	19.2	0.851	12.7
blob+count+regression	**13.8**	**0.968**	**13.5**

Fig. 7: *Performance of count on the orange data set*: The best performing algorithm is *blob+count+regression*. The count neural network brings the standard deviation down, and the linear regression reduces the bias.

Notice that although the blob detection with linear regression has an almost perfect ratio counted (accuracy), it has a higher l^2 error due to the higher variation in accuracy per image. These results demonstrate that all three parts of the algorithm improve various parts of the pipeline error: specifically the count network reduces the standard deviation of the errors and the linear regression corrects for the bias in counting.

Apples: There are a total of $1,749$ apples over 21 images giving on average 83 apples per image. We sequentially partition the data set into a training set of 11 full sized images and a testing set of 10 full sized images. The same data augmentation strategy as in the orange case is applied here.

The pixel-wise metrics for the blob detection network is evaluated at $50,000$ iterations, yielding a best threshold value of 0.02. The true positive rate with this threshold is 0.961 and the false positive rate is 0.033. The mean IU is 0.838 at the 0.37 threshold.

Apple Model	l^2 error	Ratio Counted	Std Dev
blob	46.5	1.475	24.9
blob+regression	20.4	1.025	20.3
blob+count	20.9	0.767	8.4
blob+count+regression	**10.5**	**0.913**	**7.7**
texture-based	33.8	0.617	12.8
texture-based+regression	28.8	0.682	12.8

Fig. 8: *Performance of count on the apple data set*: The best performing algorithm is *blob+count+regression*. The blob network is generous in assigning pixels as apples. The count network and linear regression corrects for these biases and reduces the standard deviation. The texture-based methods [3] have higher l^2 errors than our methods.

We ran the count-based metrics for the apple data using a blob detection neural network trained for 50,000 iterations and a count neural network trained for 25,000 iterations. The threshold value for the blob detection network was set at 0.02. Figure 8 displays the counting metrics for apples.

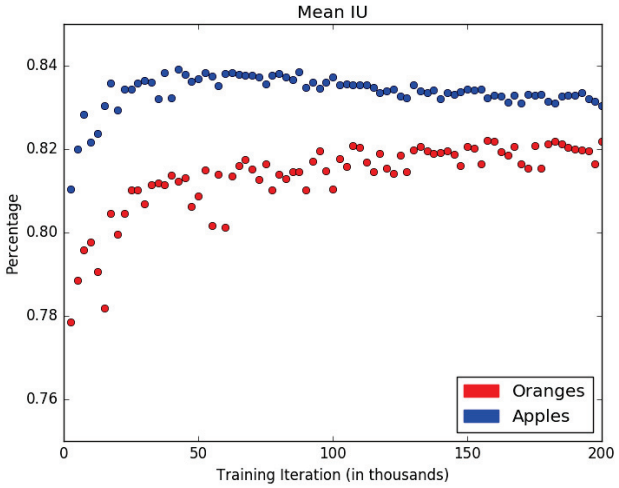

Fig. 9: *Orange and Apple Mean Intersection over Union (IU) by training iteration*: Improvement slows dramatically around $30,000$ iterations. The apple mean IU begins to deteriorate with more training, which may be due overfitting to the smaller apple data set (21 apple images compared to 71 orange images). Also notice that percentage axis is truncated to begin at 0.75.

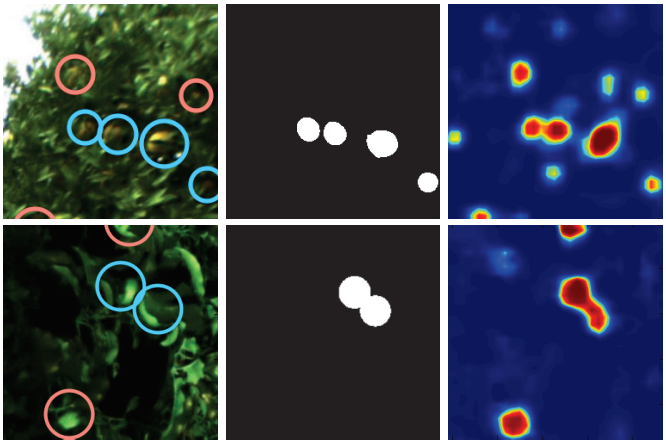

Fig. 10: Beyond ground truth labels - oranges (top) and apples (bottom). ***Left***: Original image. Fruits highlighted in red indicate those missed by labelers and blue indicates those present in ground truth labels. ***Center***: Ground truth label; the human labelers missed a few fruit. ***Right***: The blob detector picked up the missed fruit. The orange data set is challenging due to high levels of occlusion and sunlight reflecting off leaves. The apple data set is challenging due to similarity in color between apples and foliage.

Our methods outperform the texture-based methods with our best *blob+count+regression* model achieving an l^2 error of 10.5 compared to the best *texture-based+regression* model l^2 error of 28.8. The F1 score of the texture-based method is 0.76. Since our pipeline detects blob regions and estimates counts rather than identifying the specific fruit centers, we do not calculate an F1 score for our method.

The blob detector is too generous in assigning pixels as fruit, and this over-counting can be mitigated using a higher threshold value for the blob detector at the expense of potentially missing possible fruit regions. A better method, however, is to use the count neural network to accurately count the fruit in each candidate region. Like the orange count network, the apple count network reduces the standard deviation, but has a negative bias. The linear regression corrects for this bias and reduces the overall l^2 error.

Our deep learning pipeline is similar to the texture-based method in that both approaches first identity candidate regions or keypoints, and then apply a learning algorithm to count or classify. The texture-based analogue to the blob detector network is the Angular Invariant Maximal (AIM) detector which exploits specular reflection to detect keypoints. The analogue to the count neural network is the random forest classifier using Radial Histogram of Oriented Gradients (RadHOG) and Radial Pairwise Intensity Comparisons (RadPIC) as features. While this random forest classifier only outputs a binary classification, our count neural network instead outputs a numerical count estimate. By exploiting the flexibility of deep learning, our pipeline not only achieves higher accuracy, lower standard deviation, and lower overall l^2 error in our nighttime apple data set, but it is also more generalizable to other data sets such as our orange data set because it does not rely on controlled illumination.

Data Set Size: We analyze the effect of data set size on the blob network performance by training with various training set sizes of $1, 5, 10, 20$, and 35 full-sized orange images over $15,000$ iterations and testing on 10 common testing images. We did not find any noticeable improvement with a larger training set size and all, even the data set with only 1 training image, had comparable pixel-wise metrics to the previously presented orange metric with 35 training images. The ability to train a reasonable performing fruit detector with such a small training set suggests that the pre-trained network has provided the blob detector with features that truly generalize well to the fruit.

Training Time: We then analyze the performance of the blob detector as training time increases. Figure 9 demonstrates that performance plateaus around $30,000$ iterations, which corresponds to approximately 1.5 hours on a NVIDIA Titan X GPU.

We finally analyze the performance of the count network as training time increases. Performance plateaus around $20,000$ iterations, which corresponds to approximately 3 hours on a NVIDIA Titan X GPU. This analysis shows that through the use of freely available pre-trained networks, these algorithms can be trained quickly and easily.

Beyond Ground Truth Labels: We observed interesting cases during our training process where the blob detector network outperforms the human generated ground truth labels. Figure 10 illustrates situations where our human labelers miss certain fruit (highlighted in red) resulting in "inaccurate" ground truth labels. However, our blob detection algorithm detects these fruits correctly. These situations highlight the fact that our ground truth count estimates $\tilde{z}_i$ are not necessarily the same as the actual fruit count z_i. The interesting question is how these inaccurate labels affects our analysis. Our performance analysis compares predicted fruit count against the ground truth labels, so situations such as these would indicate poorer performance, when in reality our predictions are more "accurate" than the ground truth itself. We did not quantify the

extent to which this "accuracy beyond ground truth" affected our performance.

VI. Conclusion

We have presented a novel data-driven end-to-end fruit counting pipeline based on deep learning that generalizes across various unstructured environments. In order to demonstrate this generalization, we chose data sets that are challenging in unique ways: the orange data set features high level of occlusions, depth variation, and uncontrolled illumination, and the apple data set features high color similarity between fruit and foliage.

We first introduced label.ag, a crowd sourced label collection platform, along with our novel usage of Support Vector Graphics (SVG) for the purpose of storing and propagating label information. We then presented the blob detector neural network, which has high pixel-wise accuracy, achieving a mean IU of 0.813 on the oranges and 0.838 on the apples. We next presented the count neural network and a linear regression model, and demonstrated the ability of the pipeline to accurately count the number of apples in the images. We achieved a best l^2 error of 13.8 on the oranges, and 10.5 on the apples. We quantitatively and qualitatively compared our deep learning pipeline against a state-of-the-art texture-based method on our apple data set and demonstrated higher accuracy, lower standard deviation, and lower l^2 error. Additionally, our method suggests accurate counting from a limited labeled data set with a short training time. A current limitation in our approach is that we are susceptible to errors in the human-generated labels. While potentially unavoidable, we can minimize these errors by calibrating human-generated labels with ground-truth per-tree fruit counts obtained after harvest. Due to the generalizability of our pipeline, we envision our methodology being applied beyond precision agriculture to applications such as plant phenotyping (identifying as well as counting plants), phytopathology (identifying and monitoring visual disease symptoms), and even microbiology cell counting.

VII. Acknowledgements

We gratefully acknowledge members of GRASP Lab at University of Pennsylvania, Yuhpyng Chen, Tien-Chay Chen, Delaney Kaufman, Daniel Orol, DaVonne Henry, Stephanie Kemna, and others who helped in the labeling effort. We acknowledge the help of Anurag Makineni and Andrew Block in the data collection effort. We dedicate this work to the memory of Stephen Kyle Wilshusen.

References

[1] J. Das, G. Cross, C. Qu, A. Makineni, P. Tokekar, Y. Mulgaonkar, and V. Kumar, "Devices, systems, and methods for automated monitoring enabling precision agriculture," in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, Aug 2015, pp. 462–469.

[2] A. Jimenez, R. Ceres, and J. Pons, "A survey of computer vision methods for locating fruit on trees," *Transactions of the ASAE*, vol. 43, no. 6, pp. 1911–1920, 2000.

[3] Z. S. Pothen and S. Nuske, "Texture-based fruit detection via images using the smooth patterns on the fruit," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2016, pp. 5171–5176.

[4] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, Jun 2015, pp. 3431–3440.

[5] C. Schertz and G. Brown, "Basic considerations in mechanizing citrus harvest," *Transactions of the ASAE*, vol. 11, no. 3, pp. 343–0346, 1968.

[6] E. Parrish and A. Goksel, "Pictorial pattern recognition applied to fruit harvesting," *Transactions of the ASAE*, vol. 20, no. 5, pp. 822–0827, 1977.

[7] A. Syal, D. Garg, and S. Sharma, "A survey of computer vision methods for counting fruits and yield prediction," *International Journal of Computer Science Engineering*, vol. 2, no. 6, pp. 346–350, 2013.

[8] J. Zhao, J. Tow, and J. Katupitiya, "On-tree fruit recognition using texture properties and color data," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Aug 2005, pp. 263–268.

[9] S. Hayashi, T. Ota, K. Kubota, K. Ganno, and N. Kondo, "Robotic harvesting technology for fruit vegetables in protected horticultural production," *Information and Technology for Sustainable Fruit and Vegetable Production, Frutic*, vol. 5, pp. 227–236, Sep 2005.

[10] M. Hannan, T. Burks, and D. M. Bulanon, "A machine vision algorithm combining adaptive segmentation and shape analysis for orange fruit detection," *Agricultural Engineering International: CIGR Journal*, vol. 11, 2009.

[11] W. C. Seng and S. H. Mirisaee, "A new method for fruits recognition system," in *2009 International Conference on Electrical Engineering and Informatics*, vol. 1. IEEE, Aug 2009, pp. 130–134.

[12] M. B. Lak, S. Minaei, J. Amiriparian, and B. Beheshti, "Apple fruits recognition under natural luminance using machine vision," *Advance Journal of Food Science and Technology*, vol. 2, no. 6, pp. 325–327, 2010.

[13] S. Arivazhagan, R. N. Shebiah, S. S. Nidhyanandhan, and L. Ganesan, "Fruit recognition using color and texture features," *Journal of Emerging Trends in Computing and Information Sciences*, vol. 1, no. 2, pp. 90–94, 2010.

[14] H. N. Patel, R. Jain, and M. V. Joshi, "Fruit detection using improved multiple features based algorithm," *International Journal of Computer Applications*, vol. 13, no. 2, pp. 1–5, 2011.

[15] U.-O. Dorj, K.-k. Lee, and M. Lee, "A computer vision algorithm for tangerineyield estimation," *International Journal of Bio-Science and Bio-Technology*, vol. 5, no. 5, pp. 101–110, 2013.

[16] W. Guo, A. B. Potgieter, D. Jordan, R. Armstrong, K. Lawn, W. Kakerua, T. Duan, B. Zheng, H. Iwata, S. Chapman, and S. Ninomiya, "Automatic detecting and counting of sorghum heads in breeding field using rgb imagery from uav," in *International Conference on Agricultural Engineering 2016*, Jun 2016.

[17] W. Guo, U. K. Rage, and S. Ninomiya, "Illumination invariant segmentation of vegetation for time series wheat images based on decision tree model," *Computers and Electronics in Agriculture*, vol. 96, pp. 58–66, 2013.

[18] H. Patel, R. Jain, and M. Joshi, "Automatic segmentation and yield measurement of fruit using shape analysis," *International Journal of Computer Applications*, vol. 45, no. 7, pp. 19–24, 2012.

[19] S. Nuske, S. Achar, T. Bates, S. Narasimhan, and S. Singh, "Yield estimation in vineyards by visual grape detection," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Sep 2011, pp. 2352–2358.

[20] Q. Wang, S. Nuske, M. Bergerman, and S. Singh, *Automated Crop Yield Estimation for Apple Orchards*. Heidelberg: Springer International Publishing, 2013, pp. 745–758.

[21] M. Dyrmann, A. K. Mortensen, H. S. Midtiby, and R. N. Jørgensen, "Pixel-wise classification of weeds and crops in images by using a fully convolutional neural network," in *International Conference on Agricultural Engineering 2016*, Jun 2016.

[22] A. K. Mortensen, M. Dyrmann, H. Karstoft, R. N. Jørgensen, and R. Gislum, "Semantic segmentation of mixed crops using deep convolutional neural network," in *CIGR 2016, World Congress*, Jun 2016.

[23] G. Mesnil, Y. Dauphin, X. Glorot, S. Rifai, Y. Bengio, I. J. Goodfellow, E. Lavoie, X. Muller, G. Desjardins, D. Warde-Farley *et al.*, "Unsupervised and transfer learning challenge: a deep learning approach." vol. 27, pp. 97–110, 2012.

[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[25] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.