COUNTING PLANTS USING DEEP LEARNING

Javier Ribera[1], *Yuhao Chen*[1], *Christopher Boomsma*[2] *and Edward J. Delp*[1]

[1]Video and Image Processing Laboratory (VIPER), Purdue University, West Lafayette, Indiana, USA
[2] Department of Agronomy, Purdue University, West Lafayette, Indiana, USA

ABSTRACT

In this paper we address the task of counting crop plants in a field using Convolutional Neural Networks (CNN). The number of plants in an Unmanned Aerial Vehicle (UAV) image of the field is estimated using regression instead of classification. This avoids to need to know (or guess) the maximum expected number of plants. We also describe a method to extract images of sections or "plots" from an orthorectified image of the entire crop field. These images will be used for training and evaluation of the CNN. Our experiments show that we can obtain a Mean Absolute Percentage Error (MAPE) as low as 6.7% with the Inception-v3 CNN architecture.

***Index Terms*—** phenotyping, deep learning, neural network, counting, regression

1. INTRODUCTION

In important grain crops such as maize (*Zea mays* L.) and sorghum [*Sorghum bicolor* (L.) Moench], there is a close relationship between plant density (i.e, plants per unit area) and overall crop productivity (i.e, grain yield) [1, 2]. Genetic and management-based crop improvement efforts therefore often focus on identifying the optimum plant density at which to grow a crop variety for a given environment. This necessitates that plant density be determined in field research plots and large grower fields. This is typically accomplished using manual counting techniques and large work crews. Such techniques are laborious, error-prone, costly, and slow. Thus there is a clear need for technologies that enable the accurate and efficient counting of plants at multiple crop growth stages.

Image-based methods using Unmanned Aerial Vehicles (UAV) and field-deployable sensors are examples of technologies that, when integrated and properly employed, provide a viable solution for counting plants in challenging field research and production settings. Analysis techniques for image-based plant counting have not been widely reported.

In this paper we address the task of counting crop plants in a field using Convolutional Neural Networks (CNN). We also describe a method to extract images of sections or "plots" from an orthorectified image of the entire crop field. These images will be used for training and evaluation of the CNN. We make use of several CNN architectures with minimal modifications to estimate the number of plants. The plant count is regressed by the last layer of the network, consisting of a single neuron that indicates the number of plants in the image.

There exists a reasonable body of work in the literature related to counting plants and other objects. In [3], plants are counted by clustering LiDAR scans from a vehicle pushed along the field. The population of corn plants and their locations are estimated from the derived skeleton of the plants in [4]. In [5], tree crowns in a poplar plantation are counted by minimizing an energy function that models the trees as homogeneously spaced ellipsoids. In [6], objects are counted by estimating an object density function. The training procedure requires providing the object localization (i. e., pointing at the object).

Deep learning methods have been used to address a large number of computer vision tasks [7, 8, 9]. In deep learning, nonlinear features are automatically learned to represent the input data in order to build a classifier or other predictors [8, 10]. Recently, deep learning has been used in remote sensing tasks such as pixel-based classification, image restoration or target recognition [11]. There has been work reported using CNNs to count the number of objects in images. In [12], a neural network consisting of two convolutional layers followed by two linear layers is used to count objects, in particular counting digits using the MNIST dataset [13] and counting pedestrians using the UCSD dataset [14]. In [15], two techniques for counting objects in images are described. One is used for clustering a density function. The other technique uses Support Vector Machine (SVM) for the last pooling layer in the CNN and the first two fully-connected layers of the VGG19 CNN [16]. In [17] a variation of the ResNet architecture is proposed for counting and localizing cars from satellite images. In [18], a custom CNN is used to regress the number of people in dense crowds.

2. THE DATASET USED FOR OUR STUDY

We have constructed our own dataset for training and testing due to the lack of publicly available datasets for the task of plant counting. This dataset consists of 2,480 RGB images of regions of a sorghum field acquired by a UAV flying at an altitude of 40 meters. Figure 1 shows three examples of the RGB images in our dataset. All images are 546×146 pixels and contain between 0 and 22 plants.

These images are cropped from orthoimages constructed of a sorghum field acquired using a UAV at various dates in the growing season [19]. In order to generate the orthoimages, the UAV cameras are calibrated to determine their internal parameters. The location and orientation of each UAV image and the coordinate of sparse points are estimated by triangulation. The method used to generate the orthoimages is described in [19]. Figure 2 shows an orthoim-

The information, data, or work presented herein was funded in part by the Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy, under Award Number DE-AR0000593. The views and opinions of the authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof. Address all correspondence to Edward J. Delp, ace@ecn.purdue.edu

We thank Professor Ayman Habib and the Digital Photogrammetry Research Group (DPRG) from the School of Civil Engineering at Purdue University for generating the orthophotos used in this paper.

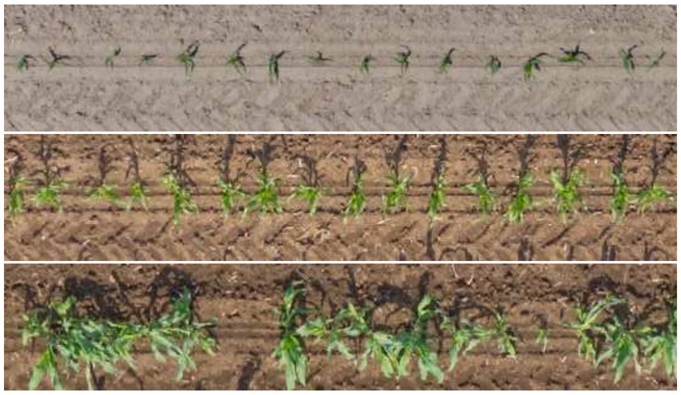

Fig. 1: Three images from our dataset, each with one sub-row of plants. Top: An image from June 13, 2016 with 16 plants. Center: The same region of the field on June 21, 2016. Bottom: An image from June 21, 2016 whose label (plant count) is uncertain.

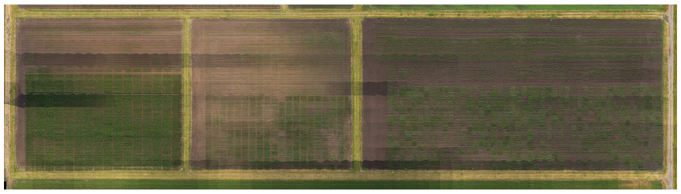

Fig. 2: Orthoimage of a sorghum field acquired on June 21, 2016. Of the three panels, only the central panel is used for analysis.

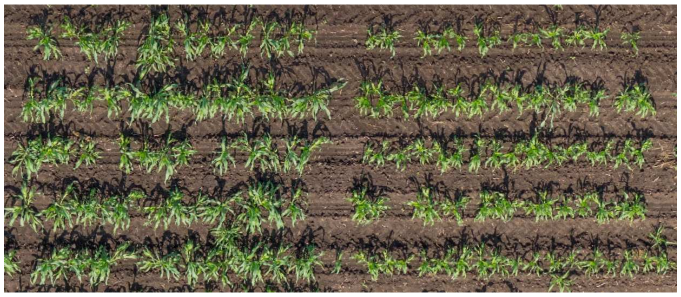

Fig. 3: Section of the orthophoto acquired on June 21, 2016, with two ranges and five rows, forming ten sub-rows.

age constructed from UAV images acquired on June 21, 2016. Each of the three main divisions of the field is known as a panel. Only the top half of the central panel is used for analysis because it is the only area that can be groundtruthed. In other areas of the field the plants are too clustered to be manually counted. Within a panel, divisions along the direction of the alignment of the plants are known as ranges, and divisions in the perpendicular direction are known as rows. A panel with M rows and N ranges has MN sub-rows, which is an aligned ensemble of a few plants such as the images in Figure 1. Figure 3 shows a section of the orthoimage with two ranges and five rows, forming ten sub-rows.

The images in our dataset are constructed from the orthophoto as follows. First, the Region Of Interest (ROI) of the orthophoto is cropped using coordinates provided by the user. For our dataset, the ROI corresponds to the central panel of Figure 2. The coordinates do not need to be very precise, but the ROI must completely include the panel and a few pixels of surrounding soil. Second, we generate a mask

$$I(x,y) = \begin{cases} 1 & \text{if pixel } (x,y) \text{ is plant material} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

using the segmentation technique we reported in [19]. This method detects which pixels correspond to plant material by selecting a region of the HSV color space as the "plants." It should be noted that the segmentation method we use for constructing the dataset is not critical and any other segmentation technique could be employed. We then define the energy function

$$p_h(x) = \sum_y I(x,y), \tag{2}$$

where the sum is performed vertically for each column x of the ROI. We denote p_h as the "horizontal profile." Figure 4 shows an example of a horizontal profile. The valleys in $p_h(\cdot)$ correspond to vertical lines between ranges of the field. Ideally, these lines should not intersect with any plant, and only traverse along soil not covered by plants. In practice, p_h at the valleys is higher than zero because the rows of the field are not completely straight or there may be unexpected weeds. The rise near $x = 12,500$ can be explained by grass and weeds from outside the field growing into the field. We estimate the location of the range-separating lines as

$$\hat{X}_n = \hat{X}_0 + n\hat{\Delta}X \tag{3}$$

$$\hat{X}_0, \hat{\Delta}X = \underset{x_0,\Delta x}{\arg\min} \sum_{n=0}^{N-1} p_h(x_0 + n\Delta x), \tag{4}$$

where $N-1$ is the number of ranges in the field. The optimization in Equation (4) is done by brute force due to the low dimensionality of the problem. Equation (3) assumes that the distance between ranges of the field is constant (ΔX). This assumption is reasonable for orthophotos that are sufficiently aligned. Once the field is divided into ranges, the same as above is done to each range separately. The goal of this is to obtain lines that divide each range into rows. We proceed by selecting only the region of the mask that corresponds to each range r as

$$I_r(x,y) = \begin{cases} 1 & \text{if } I(x,y) = 1 \text{ and } X_r \le x < X_{r+1} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

for all ranges $r = 0, \ldots, N-2$. For each range r, a vertical profile is obtained as

$$p_v^r(y) = \sum_x I_r(x,y), \tag{6}$$

where the sum is horizontal for each image row y of the ROI. In each range r, we estimate the lines that separate field rows as

$$\hat{Y}_m^r = \hat{Y}_0^r + m\hat{\Delta}Y^r \tag{7}$$

$$\hat{Y}_0^r, \hat{\Delta}Y^r = \underset{y_0^r,\Delta y^r}{\arg\min} \sum_{m=0}^{M-1} p_v^r(y_0^r + m\Delta y^r), \tag{8}$$

where $M-1$ is the number of field rows. The intersections between the vertical and horizontal lines result in a grid of NM coordinates. These coordinates construct bounding boxes around each individual sub-row of plants. These bounding boxes may not have all the same size. In order to obtain a collection of images of the same size, all bounding boxes are reshaped to the median bounding box size, preserving the same centroid for each bounding box. The pixels inside these bounding boxes form the images of our dataset.

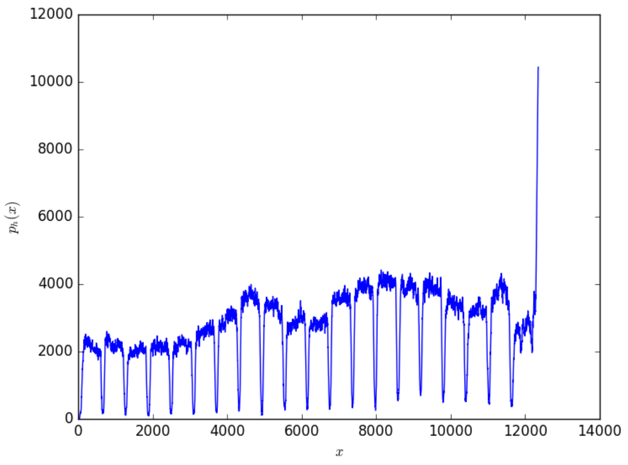

Fig. 4: Horizontal profile of the central panel of the field in Figure 2.

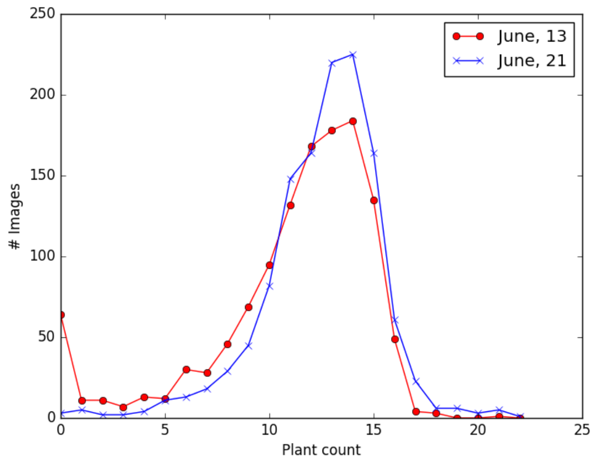

Fig. 5: Distribution of the labels of our dataset.

We collected images on two different dates, June 13, 2016 and June 21, 2016. The coordinates of the bounding boxes obtained for June 21, 2016 were used to obtain the images from June 13, 2016. We groundtruthed the dataset by manually counting the number of plants in each image. Figure 5 shows the distribution of the ground truth for each date. 1,240 images correspond to June 13, 2016 and the other 1,240 images correspond to June 21, 2016. One can note a slight increase in the number of plants between these two dates.

3. PLANT COUNTING

Our goal is to count the number of plants in an image. Our approach differs from [12, 15, 17] in that we do not consider the plant count as the class number. Commonly, the final layer of the CNN consists of as many neurons as the maximum number of plants in the picture. Such approach corresponds to a classification problem. In a classification task, a common loss function is the cross entropy:

$$H(q,p) = -\sum_x p(x) \log q(x) \tag{9}$$

where $q(x)$ are the activations at each of the neurons x of the last layer, and $p(x)$ is the label. Setting $p(x)$ with one-hot encoding, i.e,

$$p(x) = \begin{cases} 1 & x = C_i \\ 0 & x \neq C_i, \end{cases} \tag{10}$$

where C_i is the label (plant count) of the i-th image, then the loss function only takes into account a single neuron (the C_i-th neuron):

$$H(q,p) = -\log q(C_i) \tag{11}$$

This implies that during training, we would ignore all the remaining activations. One can interpret $p(x)$ as the certainty that an image belongs to the class x. Equations (10) and (11) are then appropriate when classes are independent. However, the classes in our dataset are not completely independent. An example is the case when an image contains 2 plants. If the network's estimate is 3, the penalty for the network should be much lower than if the estimate was 5. This is aggravated in our dataset because there is important label noise. When plants are clustered, the label may be more than one count incorrect from the real value (see Figure 1). The classification approach also presents the problem of having to set the maximum number of classes a priori.

To address these problems, we employ the $\mathcal{L}_p$ norm as the loss function to be minimized:

$$\mathcal{L}_p(\hat{x}, C_i) = |\hat{x} - C_i|^p, \tag{12}$$

where $\hat{x}$ is our plant count estimate. The $\mathcal{L}_p$ loss becomes the Mean Squared Error (MSE), a common loss function for regression, when $p = 2$. When $p = 1$, this error can be interpreted as the average error in the number of plants that we make when estimating the number of plants in an image.

We tested various CNN architectures (AlexNet [9], Inception-v2 [20], Inception-v3 [21], and Inception-v4 [22]), where the final layer has been replaced with a single neuron without non-linearity. The activation of this neuron corresponds to the pre-softmax activation of the output layer. The value of the activation may be non-integer and occasionally negative. Because of this, we take the absolute value and round it to the closest integer. The final value is our plant count estimate denoted as $\hat{x}$. In this way, the task of plant counting is posed as a regression problem instead of as a classification problem. Unlike with the cross entropy, when using the $\mathcal{L}_p$ norm, the further an estimate is from the label, the more it is penalized.

Many neural network architectures require rectangular images for training, e.g, in AlexNet and Inception-v2, the input images are 224×224 and 299×299, respectively. However, the images of our dataset are not rectangular, but 546×103. To overcome this limitation, we slightly modify the last layers of the networks. Our modifications consist only on removing one pooling layer. In AlexNet [9], our input image size would make the size of the feature maps after the last max pooling to be 15×2. As a 2 pixels-wide feature map may be too narrow to capture horizontal features, we remove the last max pooling layer. Thus, the feature map before the first fully-connected layer becomes of size 32×5. In Inception-v2, Inception-v3, and Inception-v4, after the last convolutional layer we obtain a feature map of size 18×4, 15×1, and 15×1, respectively. These feature maps cannot be used with the 8×8 average pooling layer. We remove this last average pooling layer and we use the feature map directly as input to the fully-connected layer. Figure 6 shows the last part of Inception-v3, with our modifications.

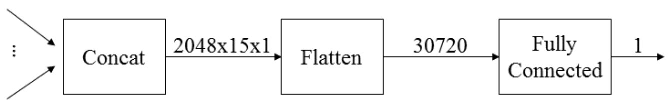

Fig. 6: Modified part of Inception-v3. Note that we have removed the max pooling. The last neuron regresses the number of plants.

Architecture	Testing MAPE (w/o data augmentation)	Testing MAPE (w/ data augmentation)
AlexNet	8.3 %	7.9 %
Inception-v2	8.2 %	6.7 %
Inception-v3	**7.1** %	**6.7** %
Inception-v4	12.4 %	11.4 %

Table 1: Performance of various neural network architectures using our dataset ($p = 1$).

4. EXPERIMENTAL RESULTS

Each channel of our dataset is normalized using the average and standard deviation of the training set. A limitation of our dataset is the small size of the training set. One way to address this problem is to make use of data augmentation techniques. Linear and nonlinear transforms are used to create "new" training images without changing the label (plant count). We use four types of data augmentation: vertical flip, horizontal flip, brightness change, and contrast change [23]. The data augmentation techniques are used in random order at training.

The dataset is randomly split such that 80% of the samples are used for training, 10% for validation, and 10% for testing. With a total of 2,480 images, this partition results into 1,983 images for training, 250 for validation, and 249 for testing. For evaluation purposes, the error metric we used is the Mean Absolute Percentage Error (MAPE):

$$\text{MAPE} = 100 \overline{\frac{|\hat{x} - C|}{C}} \tag{13}$$

where $\overline{(\cdot)}$ indicates the average along all the images in the testing set.

First, we evaluated the effect of p (in Equation 12) on the resulting MAPE, using the AlexNet architecture. We employ AlexNet for the evaluation of p because it corresponds to the shortest training time. Table 2 shows the results for different values of p. We obtain the lowest MAPE with $p = 1$. As the MAPE metric and the $\mathcal{L}_1$ norm only differ in a constant $\frac{100}{C}$, it is reasonable that training with $p = 1$ yields the lowest error. Then, setting $p = 1$, we evaluated the performance of various neural network architectures, including Alexnet, Inception-v2, Inception-v3, and Inception-v4. Table 1 shows the error on the testing set. We obtain the lowest MAPE of 6.7% with Inception-v3. A MAPE of 6.7%, if the true label is $C = 14$, means that our estimate is in average less than one plant off the real value. Also, note from Table 1 that the error is consistently reduced when using data augmentation. Data augmentation accounts for a reduction of the error between 0.4% and 1%. Inception-v4, the CNN with the most number of parameters, produces a higher error than any other architecture. This may indicate overfitting and the need to increase the dataset size, to use more data augmentation techniques, or to investigate more aggressive regularization techniques.

We use TensorFlow [24] for the implementation of the networks. We did not use pre-trained model parameters for the CNN for the results shown in Tables 1 and 2. The minimization technique used

p	Testing MAPE (w/ data augmentation)
1	**7.9** %
1.5	8.5 %
1.8	8.4 %
2	8.2 %

Table 2: Impact of p on the MAPE when using AlexNet and data augmentation.

was Adam [25] with exponential decay rates of $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and $\epsilon = 1e^{-8}$. When using AlexNet and Inception-v2, the learning rate was set to $\alpha = 1e^{-4}$, and when using Inception-v3 and Inception-v4, the learning rate was set to $\alpha = 1e^{-5}$. We used a batch size of 96. The stopping criteria was the MAPE on the validation set. The training was run for 50,000 iterations when training AlexNet and Inception-v2, and for 500,000 iterations when training Inception-v3 and Inception-v4. The iteration at which the model returned the lowest validation error was selected. The reported testing errors correspond to the testing error of the model at this iteration. This occurred after 12 hours of training for the Inception-v3 case. The machine used for computation was an Intel i7-5930K and one GeForce GTX Titan X.

5. CONCLUSIONS AND FUTURE WORK

We described a method utilizing CNNs to estimate the number of crop plants in a field using our own dataset of RGB UAV images of sorghum plants. We evaluated the performance of various neural networks for the task of plant counting with minimal modifications of their architecture. We observed the smallest Mean Absolute Percentage Error (MAPE) of 6.7 % on the testing set with the Inception-v3 architecture [21]. The value for p in the $\mathcal{L}_p$ norm was $p = 1$ in that it provided the lowest error in our experiments. Future work will include incorporating more data, adopting more data augmentation techniques, and estimating other properties of plants such as the leaf count.

6. REFERENCES

[1] Y. O. M'Khaitir and R. L. Vanderlip, "Grain sorghum and pearl millet response to date and rate of planting," *Agronomy Journal*, vol. 84, no. 4, pp. 579–582, July 1992.

[2] I.S. Tokatlidis and S. D. Koutroubas, "A review of maize hybrids' dependence on high plant populations and its implications for crop yield stability," *Field Crops Research*, vol. 88, no. 2, pp. 103–114, August 2004.

[3] Y. Shi, N. Wang, R. K. Taylor, W. R. Raun, and J. A. Hardin, "Automatic corn plant location and spacing measurement using laser line-scan technique," *Precision Agriculture*, vol. 14, no. 5, pp. 478–494, 2013.

[4] C. Wang, X. Guo, and C. Zhao, "Detection of corn plant population and row spacing using computer vision," *Proceedings of the International Conference on Digital Manufacturing and Automation*, pp. 405–408, August 2011, Zhangjiajie, China.

[5] G. Perrin, X. Descombes, and J. Zerubia, "A marked point process model for tree crown extraction in plantations," *Proceedings of the IEEE International Conference on Image Processing*, vol. 1, pp. I–661–4, September 2005, Genoa, Italy.

[6] V. Lempitsky and A. Zisserman, “Learning to count objects in images,” *Proceedings of the Advances in Neural Information Processing Systems*, pp. 1324–1332, December 2010, Vancouver, Canada.

[7] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, “Deep learning,” *Nature*, vol. 521, pp. 436–444, May 2015.

[8] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, August 2013.

[9] Alex Krizhevsky, “One weird trick for parallelizing convolutional neural networks,” *arXiv preprint arXiv:1404.5997*, April 2014.

[10] C.-C. Jay Kuo, “Understanding convolutional neural networks with a mathematical model,” *Visual Communication and Image Representation*, vol. 41, pp. 406–413, November 2016.

[11] L. Zhang, L. Zhang, and B. Du, “Deep learning for remote sensing data: A technical tutorial on the state of the art,” *IEEE Geoscience and Remote Sensing Magazine*, vol. 4, no. 2, pp. 22–40, June 2016.

[12] S. Seguí, O. Pujol, and J. Vitria, “Learning to count with deep object features,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 90–96, June 2015, Boston, MA.

[13] Y. LeCun, C. Cortes, and C. JC. Burges, “The MNIST database of handwritten digits,” 1998.

[14] A. B. Chan and N. Vasconcelos, “Counting people with low-level features and bayesian regression,” *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 2160–2177, April 2012.

[15] J. Shao, D. Wang, X. Xue, and Z. Zhang, “Learning to point and count,” *arXiv preprint arXiv:1512.02326*, December 2015.

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

[17] T. N. Mundhenk, G. Konjevod, W. A. Sakla, and K. Boakye, “A large contextual dataset for classification, detection and counting of cars with deep learning,” *Proceedings of the European Conference on Computer Vision*, pp. 785–800, October 2016, Amsterdam, The Netherlands.

[18] C. Wang, H. Zhang, L. Yang, S. Liu, and X. Cao, “Deep people counting in extremely dense crowds,” *Proceedings of the ACM International Conference on Multimedia*, pp. 1299–1302, October 2015, Brisbane, Australia.

[19] J. Ribera, F. He, Y. Chen, A. F. Habib, and E. J. Delp, “Estimating phenotypic traits from UAV based RGB imagery,” *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Workshop on Data Science for Food, Energy, and Water*, August 2016, San Francisco, CA (to appear).

[20] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, March 2015.

[21] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, June 2016, Las Vegas, NV.

[22] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” *arXiv preprint arXiv:1602.07261*, August 2016.

[23] Artem Rozantsev, Vincent Lepetit, and Pascal Fua, “On rendering synthetic images for training an object detector,” *Computer Vision and Image Understanding*, vol. 137, pp. 24–37, August 2015.

[24] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” *arXiv preprint arXiv:1603.04467*, March 2016.

[25] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *Proceedings of the International Conference for Learning Representations*, vol. abs/1412.6980, April 2015, San Diego, CA.