Phenotyping problems of parts-per-object count

Anonymous CVPPP submission

Paper ID 12

Abstract. The need to count the number of parts per object arises in many yield estimation problems, like counting the number of bananas in a bunch, or the number of spikelets in a wheat spike. We propose a two-stage detection and counting approach for such tasks, operating in field conditions with multiple objects per image. The approach is implemented as a single network, tested on the two mentioned problems. Experiments were conducted to find the optimal counting architecture and the most suitable training configuration. In both problems, the approach showed promising results, achieving a mean relative error in range of $11\%-12\%$ of the total visible count. For wheat, the method was tested in estimating the average count in an image, and was shown to be preferable to a simpler alternative. For bananas, estimation of the actual physical bunch count was tested, yielding mean relative error of 12.4%.

Keywords: deep neural networks, object detection, part counting, yield estimation

1 Introduction

This work handled a specific kind of phenotyping problems: visual objects' part counting, done by first detecting the objects in the image, and then counting their constituting parts. Such problems repeatedly arise in field phenotyping contexts. Examples include counting the number of bananas in banana bunches [48], the number of spikelets in wheat spikes [37, 3], the number of flowers on apple trees [10, 13], or the number of leaves in potted plants [11, 21]. While such problems involve (object) detection and (part) counting, they should not be confused with single-stage counting or detection problems, and require a non-trivial combination of them.

Object detection is a basic step in several types of agricultural applications. One important kind is robotic manipulation, like harvesting [43, 4], plants' diseases and pests recognition [15] or autonomous selective spraying [8]. In another type of applications, object detection is the first stage for measuring object properties like height, width and length [5, 49], cluster size estimation [24], or object classification [18, 54, 25]. In a third type of applications object detection is used as part of an objects (not parts) counting task [35]. In recent years the detection task is performed mainly with deep Convolutional Neural Networks (CNNs) [41, 28, 45], that enabled dramatically increased accuracy.

Fig. 1: Example images with ground truth annotations. **Left:** Bananas in banana bunches. **Right:** Wheat spikelets in spikes. Blue bounding boxes are placed around measurable objects. Black dots are placed on countable parts. The target count number (number of visible parts) is stated above each bounding box.

Object counting is also a common phenotyping need for tasks such as yield estimation [29, 52, 35], blossom level estimation for fruit thinning decisions [10, 51, 13], or plants' leaf counting for growth stage and health estimation [34, 7, 11, 21]. Different CNN architectures are currently the state of the art for counting as well [36, 9] and for other phenotyping tasks [22].

Specifically, we address two part-counting problems in this work: banana count in a bunch, and spikelet count in a wheat spike. Image examples with annotations for each task are shown in Fig. 1. In both cases, the part count is an important yield indicator, and measuring it automatically is important for yield prediction and breed selection [17]. Spikelets counting can assist in crop management as it can be seen as a form of yield quantification for wheat crops [3] and the number of bananas in a bunch is one of the properties that is related to bunch weight and thus productivity [50]. These two very different problems are handled here with the same algorithm, so we believe the proposed method is fairly general and can handle other part-counting problems with minor adjustments.

One may wonder why the 'parts-counting' task (which is essentially a 'detect-then-count' procedure) cannot be reduced into plain counting, by simply taking pictures containing a single object in each image. Indeed, in principal this is possible. In some of the problems mentioned above the two-stage part-object nature was often bypassed by using images of a single centralized object [13, 34, 7]. However, solving a part counting problem by taking images of single objects has significant disadvantages. The counting procedure needs to be automated, rather than being done by tedious manual approach. If a human has to detect the objects and take single-object images, the process is partially manual with human intervention. It is hence slower, costs more, and less scalable than a fully automated system. Another option is to have a robotic system first detecting the objects, than moving to a close position to each object for taking a second image, isolating it from a wider scene. While this does not involve human intervention,

such a system is much more complex to develop, and it also is likely to be slower than a system based on high resolution images containing multiple objects. Hence for automatic and flexible field phenotyping, the solution of keeping one object per image is significantly less scalable.

While parts counting is essentially a two-stage 'detect then count' task, it nevertheless raises some unique difficulties. First, parts are much smaller than their containing objects. Therefore high initial resolution and a re-scaling mechanism are required. Second, the system has to filter which detected objects are passed to the counting stage, and this involves a complex array of considerations. On the one hand not all objects are 'countable': some objects are too far, out of focus, occluded, or not well-detected. On the other hand one can usually afford measuring only a subset of the objects, and sometimes it is better to have good measurements for few than inaccurate measurement for many. While the exact policy may be task-dependent, the filtering creates dependency between the detector and the data distribution seen by the counter. As will be demonstrated in Section 4.2, this has implications w.r.t the preferred training procedure.

A third issue arises since counting itself is not a solved problem: there are several competing approaches, and the better choice seems to depend on data distribution and quantity, and on availability of annotation. Some counting networks work by explicit detection of the objects to count [6, 17, 33], some use density estimation [44, 26], and others directly regress the count locally or globally, without intermediate stages [32, 52].

This work explores the design space of possible solutions for parts-counting, and finds a useful and relatively simple solution. We proposed a single network combining stages for the object detector, cropping and scaling of the objects, and then a counting stage with two optional implementations. One uses a counting module which is a direct regressor, hence part position annotation is not necessary for training. The other does require the objects' parts annotations, but in addition to counting, it also provides parts localization. Following experiments, we recommend choosing between the two possible solutions based on computational efficiency and the need for part localization considerations, since both solution provide good and comparable results. The training procedure of the entire network is also discussed, showing that it can be done jointly or separately, as long as proper noise is introduced during learning.

We benchmark the proposed solutions on the two problems using several metrics. The most intuitive count measure is the (average) relative deviation, stating the average deviation of the count from its true value in percentage. Statistically, an important measure is the fraction of explained variance in the data. In relative deviation terms, the networks achieve 10.1% and 10.8% for banana and spikelet count respectively. Statistically, we are able to explain 0.58 and 0.75 of the count variance for the two problems respectively.

While the obtained accuracy is encouraging, it is obtained for estimation of the visible number of parts. For bananas, which are round and with significant volume, the visible number and the actual number of bananas in a bunch may be far from each other, as approximately half the bananas cannot be seen. We

have collected a separate dataset of bunches with actual counts made manually, and were able to establish a strong linear connection between visible and actual banana count. Hence fairly accurate actual counts can also be obtained.

Our contribution is hence two-folded: developing a successful general algorithmic framework for the parts-per-object counting problem, and showing its merit on two important phenotyping cases. The rest of the paper is organized as follows: we review related work in Section 2, describe the algorithm in Section 3, present results in Section 4, and discuss further work in Section 5.

2 Related work

Since our network is composed of two sub-networks for detection and counting, we discuss how these tasks are approached in general and in the agricultural domain, than focus on the works closest to our specific tasks.

Visual object detection had significantly advanced in recent years by incorporating deep neural networks extending CNN architectures. This enables tackling more intricate tasks as we do here. One line of influential networks includes two-staged networks like Faster R-CNN [41] or Mask R-CNN [19]. These networks include a stage of Region Proposal Network (RPN) which finds initial object candidates, and a second sub-network which accepts the candidates and handles final classification and bounding-box fine-tuning. Object Candidates are passed between the two stages by sampling the feature maps produced by the RPN at Regions of Interest (RoIs). Technically, this is done with a non-differentiable layer termed RoI Pooling in Faster-R-CNN [16, 41], and RoI Align in Mask-R-CNN [19], which samples feature map regions into small tensors with a fixed spatial extent. In our work we use an RoI Align layer to pass detected objects from the detector into the counting sub-network.

Another line of work includes fully differentiable one-stage architectures. Early one-stage networks as YOLO [40] and SSD [31] presented faster detectors, but with accuracy lower by $10-40\%$ relative to two-stage methods [28]. Later one-stage networks like RetinaNet [28] and more recently EfficientDet [47] were able to remove the accuracy gap. Specifically RetinaNet introduced the usage of a Feature Pyramid Network (FPN) [27] for handling larger scale variability of objects, and a 'focal loss' variant which well balances between positive examples (which are few in a detection task) and negative examples.

In the agricultural context, object detection has numerous usages as it is a basic building block in automated harvesting [43, 4], yield estimation [52, 13], and phenotyping [5, 49]. Several detection benchmarks [42, 54, 6] have shown that current detectors obtain high F_1 and AP scores in agricultural settings, hence encouraging two-staged tasks as pursued here. However, in [42, 54] the datasets included mostly images taken very near to the crop of interest, hence including only few (typically less than 5) large objects. These images are significantly less challenging than images taken in our wheat spikelet counting task. In another work, images were taken in field conditions from a significant distance, with the target of fruit yield estimation [6]. To overcome the problem of object occlusions,

it was assumed that on average there is a constant ratio of visible to occluded fruits, and detection outputs were calibrated with ground-truth counts. We use a similar assumption in this work.

Counting tasks in agriculture contexts were approached with traditional computer vision techniques [30, 1, 2], but has advanced in recent years by the incorporation of CNNs. Basically, there are two main approaches for building counting networks. The first is based on some form of explicit object localization: objects are first detected, then counted. The detection task can be defined as detection of the object's centers, and so the output is a 'density estimation' heat map, a two dimensional map showing where 'objecthood probability' is high [44, 21]. Alternatively localization can be based on bounding box detection [17, 33, 35] or segmentation [38, 14, 55]. This methods require object location annotations like center point annotations (for density estimation), bounding boxes (for detection) or surrounding polygons (for segmentation). The second approach is via a global [39, 11, 21] or local direct regression model [32, 52]. In global regression, the model implements a function mapping the entire image (or region) to a single number, which is the predicted count. In local regression the image (or region) is divided into local regions and each of them is mapped to a local predicted count. The local counts are then summed to get the global count.

The literature contains contrasting evidence regarding the relative accuracy of localization-based versus direct regression methods. In [33] a detection based method was found superior to direct regression and it is claimed to be more robust to train-test domain changes. Contrary, at [32] direct regression provides higher accuracy than density estimation. At [21] the two approaches show similar performance. An advantage of detection based methods is in being more "debugable" and "explainable", as it allows to visually inspect where in the image the network finds objects. It is also able to provide the objects location information when it is needed for further processing. However, more computation effort is required at test time, and an additional annotating effort at train time.

While a lot of previous work has addressed wheat spike counting [32, 52, 17, 33, 14, 2, 55] we are not familiar with work attempting to solve the spikelet-count-per-spike or the banana-per-bunch problems we face here. Global spikelet count was addressed in the recent papers [37, 3]. In [37] both spike and spikelets were counted in glassdoor conditions, where background is surpressed and each image contains a few large spikes. A density estimation approach was used, with very good results of relative error $< 1\%$ for spikelets and $< 5\%$ for spikes. While in principal the average spikelet-per-spike statistic can be estimated from the two global counts, we show in our experimental results that this method provides inferior accuracy in field images. In [3] only spikelets are counted with density estimation, but in challenging field conditions as we address in our problem.

3 Method

The suggested network is comprised of several components. First is a detection section responsible for detecting the objects of interest. These are passed to the

RoI-Align module that crops and resizes the detected objects from the original image. The crops are passed to the counting section, in which each crop is treated as an independent input image. It is assumed that each crop contains a single detected object, and the counting section outputs the parts' count for that object. The assumption's correctness naturally depend on the performance of the object detection section. The basic composition of the detection and counting network is demonstrated in Fig. 2.

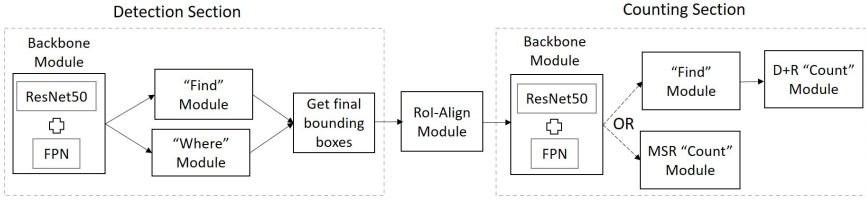

Fig. 2: The Detection and Counting Network. The network includes two separate sections for object detection and part counting, connected by an object re-sampling layer .

The detection section is a re-implementation of the RetinaNet architecture, composed of 'Backbone', 'Find' and 'Where' modules, whose details are explained in Section 3.1. The RoI-Align module receives as input the original image and a set of rectangle coordinates of the detected objects. By re-sampling the image according to the given coordinates, it outputs a tensor of size $s \times s \times n_d$ of image crops, where s is the new object size (empirically set to 640) and n_d is the number of detected objects. We use the RoI-Align implementation of Mask-R-CNN [19] which currently does not support gradient transfer. The counting section includes re-implementations of the counters proposed by [21], composed using 'Backbone', 'Find' and 'Count' modules. Modules re-used in different sections ('Backbone' and 'Find' appear in both the detector and counter sections) share code and architecture, but not parameters. The counters and their composition using the modules are described at Section 3.2.

Both the Detection and the counting sub-networks have their own separate losses, which are learned separately. The training methodology of the whole network is discussed at Section 4.2.

3.1 The detection section

The relevant modules for the detection section of the network are the following:

"Backbone": A module used for creating a dense feature-rich representation of the image. It is based on applying ResNet-50 [20] as a dense convolutional network followed by FPN [27] on top of it. The FPN architecture generates a rich 5-scale feature pyramid (P3-P7) representation of the input image. These tensors include similar representations of the original image in multiple octaves, with P_i twice smaller than P_{i-1} in its spatial dimensions. Therefore they enables object detection at multiple scales. In all experiments we have used the ResNet-50 network with weights pre-trained on ImageNet and fine-tuned them.

"Find" (for detection): The module includes 4 convolutional layers and it is trained for spatially locating objects in an input tensor representation. It supports two output modes with the first used for the detection section, the second for the counting section. For the detection pipeline, it implements the fully convolutional classification sub-network of RetinaNet that predicts the probability of object presence at each spatial position of an input tensor. At each position, this probability is estimated for nine possible anchor rectangles of predefined sizes and aspect ratios. The output tensor is thus with the same spatial dimensions as the input, but with nine score maps. The 'Find' module processes independently all the five pyramid tensors produced by the Backbone's FPN. It is trained with the Focal loss, designed to address possible imbalance between foreground and background classes during training [28].

"Where": This module implements the bounding box regression sub-network of RetinaNet. Accepting a representation tensor as input, it predicts for every position and possible anchor (among the nine considered) a 4-dimensional bounding box refinement vector (so the output includes 36 maps overall). The vector includes corrections required for the bounding box to better match the object in its $(x, y, width, height)$ parameters. It is trained by propagating an smooth-$L1$-regression loss matching the predicted values to ground-truth rectangles - only for anchors with relevant objects. Like the 'Find' module, the module processes all the pyramid tensors $P3 - P7$, and produces 5 output tensors.

"Get final bounding boxes": This function accepts the output tensors of the 'Find' and 'Where' modules and creates a refined list of predicted boxes containing presumed objects. It filters boxes with predicted object probability higher than 0.7 from the 'Find' module, and decodes the refined boxes for them according to the 'Where' module. The predictions from all pyramid levels are merged and a non-maximum suppression procedure with a threshold of 0.3 is applied to yield the final detections list.

3.2 The counting section

Given an input image with a single object, the counting section outputs the count value of its parts. It has two variations, following the proposed architectures by [21]. The first variation uses the implementation of the Multiple Scale Regression (MSR) algorithm, that directly regresses the count value. The second variation is a re-implementation of the Detection+Regression (D+R) algorithm which predicts heat maps of the parts centers in addition to counting them, and thus requires parts location annotations in training. In both cases, the counting section starts with a second 'Backbone' module receiving as input detected region objects. Figure 3 include box-diagrams of the two Counting methods.

MSR "Count": In this module, the representation tensors P3-P7 generated by the 'Backbone', or a subset of them, are each sent to a direct regression module, so multiple count estimations are produced based on different resolutions. This regression module includes several convolutional and fully connected layers culminating in a two outputs neurons: the first predicts the expected count, and the second estimates the variance of the error expected in prediction, using the

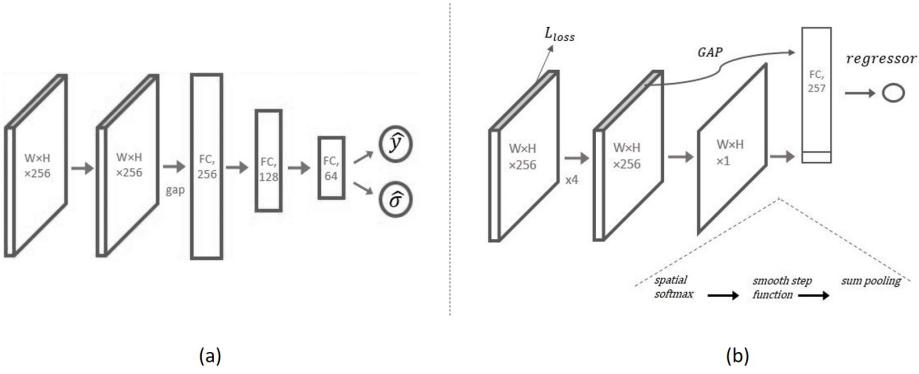

Fig. 3: Architectures of the proposed counters. **a:** The MSR 'Count' module architecture, **b:** The D+R 'Find' and 'Count' modules.

loss function suggested in [23]. The count estimates are then fused as a weighted average with weights based on the variance values.

For the D+R variation, the count section includes a 'Find' module variation described next, and then a different 'Count' module.

"Find" (for counting): This module has similar architecture to the "Find" module used in detection, but it is used to 'Find' center points instead of bounding box rectangles. The module operates only on the high-resolution pyramid scale (P3), and a single output map of the same resolution is created, stating the probability of object presence at each location. To learn this network, a ground truth heat map is created in training, with a Gaussian kernel placed around each parts' center. Like the detection "Find" module, the module contains four convolutional layers, but after that a single final map is predicted. It is trained to mimic the ground truth heat map using a dense L_1 regression loss.

Following [21], we tested the option of guiding internal layers by predicting an intermediate heat map after each convolutional layer, and forcing it to mimic the ground truth heat map with additional regression losses. When this is done, the intermediate guiding heat maps are created with decreasing kernel size, so the heat map regression task is cruder and simpler at initial layers.

D+R "Count": This module gets from the 'Find' module the predicted heat map and the feature tensor preceding it, and provides a count estimate based on them. It is an implementation of the "Counting sub-network" of the D+R pipeline of [21]. The heat map is subjected to a smooth non-maxima suppression operation, keeping activity of the most active points close to ones while all others are zeroed. These remaining active points are object center predictions, and a global sum operation now gives an initial detection-based estimator of the count. In addition to this path, the tensor preceding the map is globally summed to extract additional useful features, and the final count estimate is computed as linear regression from these features and the detection-based estimate.

Dataset	Train (Images/Objects/Parts)	Validation (Images/Objects/Parts)	Test (Images/Objects/Parts)
Wheat	44/223/3523	27/102/1779	30/135/2347
Banana	72/82/6822	34/41/3372	35/44/3432

Table 1: Datasets sizes

4 Experiments and results

We describe the datasets used and performance evaluation methodology in Section 4.1. Results in the part counting tasks and experiments measuring the relation between visible and actual number of parts for banana bunches are presented in Section 4.2.

4.1 Experimental setup

Wheat dataset: The collected dataset includes 101 RGB images taken in an agricultural facility in the Central District of Israel. High resolution images of several wheat varieties in field conditions were taken using a commercial DSLR camera. Since the original images included hundreds of wheat spikes, several regions (usually 4-5) in focus were handpicked in each image. Theses areas were cropped and treated as separate images that were passed for annotation of well defined, measurable spikes.

Bananas dataset: 141 RGB images were captured in a facility in the Northern District of Israel. Images included banana bunches of different varieties and stages of the reproductive phase. These were taken using commercial 9MP-12MP mobile device cameras and a digital point-and-shoot cameras in field conditions. The common resolutions were 2340×4160 and 3024×4032.

All images were annotated with bounding boxes for countable objects, and with center dot annotation for each object part. Countable objects were defined as "objects for which a human annotator can visually count the parts". At least for wheat, the distinction between countable and non-countable spikes is slightly vague, and depends to some extent on the annotators' judgement. The number of images, objects and parts used is listed in Table 1.

Detection performance is measured by the Average Precision (AP) metric [12]. An object is considered to be found if its Intersection over Union (IoU) is at least 0.5 with a ground truth object annotation. For counting, denote the set of ground truth counts of the test set by $\{y_i\}_{i=1}^N$ and the set of count prediction of the model by $\{\hat{y}_i\}_{i=1}^N$. We estimate counting accuracy using the Mean Relative Deviation (MRD) and $1 - FVU$ statistics defined as follows:

$$MRD = \sum_{i=1}^{N} [\frac{|\hat{y}_i - y_i|}{y_i}] \quad 1 - FVU = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})} \tag{1}$$

Where $\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$ is the mean ground truth count. The relative deviation states the count deviation as a fraction of the total count (the count deviation percentage). $1 - FVU$, known as the fraction of explained variance, checks if the quality of the predictor we use (its least squares error in the nominator) is significantly better than the trivial predictor of guessing that y_i is always equal to the mean $\bar{y}$ (the least square error of this predictor is the denominator - it is also the variance of y).

For each dataset, counting networks were trained for 300 epochs. The best epoch was chosen as the one with the lowest mean relative deviation, measured on the validation set, and averaged across IoU thresholds of 0.3, 0.5 and 0.7. This model was then tested on the test set images, providing the reported results.

4.2 Results

Based on the stages order of the suggested method, we start by describing object detection results. Then we report part counting results obtained with various counters, and consider the effect of training procedure on the obtained accuracy. Next, we compare the results to an alternative approach, in which objects and parts are counted independently, and report experiments predicting the actual physical count for banana bunchs.

Detection results: High confidence thresholds were chosen for the trained object detectors, to provide high precision. This ensures that at least 90% of the detections ares countable objects, with some cost in recall rates. Table 2 shows the recall and precision of the detectors used on the test set. As can be observed from the recall statistics, for wheat the detection problem is more difficult, due to the fine distinction required between countable and non-countable spikes.

Dataset	Number of Objects	Recall	Precision
Wheat spike	135	45.9%	93.9%
Banana bunch	43	81.4%	100.0%

Table 2: Test set recall and precision of the object detectors.

Part counting accuracy: We have experimented with variations of the two counting architectures suggested in [21], described in Section 3. Counting modules were trained while keeping the detection section fixed. Results of the tested counters are shown in Table. 3. It can be seen that the best counters are able to achieve relative deviation close to 11% in both tasks. The best methods for the two tasks are not the same, though. On the wheat dataset, the direct regression MSR performs better, while for the banana dataset detection based counting is superior. We believe this lack of consistency across tasks can be attributed to a simple fact: detection of bananas is much easier than detection of wheat spikelets, which are smaller and sometimes confused with spikelets of neighboring spikes. Since explicit detection of the relevant parts is more difficult for wheat, the D+R method based on such explicit detection is weaker. It can be seen in table 3 that for the wheat task, the results improve when less attempts

Counting Approach	Wheat		Banana	
	Mean Relative Deviation	1-FVU	Mean Relative Deviation	1-FVU
MSR (p3)	11.4%	0.718	13.5%	0.341
MSR (p3 - p5)	10.8%	0.745	13.4%	0.349
MSR (p3 - p7)	11.6%	0.652	12.7%	0.375
D+R	14.1%	0.641	12.1%	0.461
D+R (same radii)	12.9%	0.684	11.5%	0.581
D+R (no intermediate losses)	11.0%	0.759	13.1%	0.426

Table 3: Part counting results for several counter ablations. The first three rows show the results obtained by the MSR approach, when integrating a single resolution (P_3), 3 resolutions ($P_3 - P_5$), or the originally proposed 5 resolutions ($P_3 - P_7$). The last three rows show results of the D+R method. The original D+R method is compared to a variation in which the learned heat maps are generated with fixed kernel size (same radii), and to a variation in which no intermediate losses are used.

are made to "educate" the network toward explicit detection (as we move from the original D+R method toward complete lack of intermediate detection losses).

The part detection results of the D+R method, presented at Fig. 4, further support the above mentioned view. It can be seen that banana fruits are much easier to detect, with AP of 0.89 obtained, while for spikelets it is only 0.61. A demonstration of the networks D+R pipeline operation can be seen in Fig. 5.

Training methodology. In the training procedure described above, the detector is kept fixed while the counter is training. However, several other options are possible. One appealing option is to use a 'perfect' detector in training, which provides the counter with ground truth object bounding boxes, thus enabling it to train in 'ideal' conditions. An opposite intuition is that the counter should train in far from ideal, noisy conditions, so training should include detection noise. We have experimented with several alternatives using the best counter found for the wheat dataset (MSR using resolutions P3-P5), and the results are shown in Table 4. The main trend visible is that training with a perfect oracle detector provides inferior results - the counter needs to see imperfect bounding boxes while training. However, obtaining such imperfect bounding-boxes can be done either by using the imperfect detector, or/and by adding explicit detection noise. Training the detector simultaneously with the counter reduced accuracy a bit for the wheat problem, but it can be considered if training time is an issue (as training simultaneously is faster than stage-wise training). Moreover using a similar simultaneous training approach helped us to improve the relative deviation of the D+R method on the banana dataset to 10.1% (with 1-FVU = 0.584)

Image average counts and comparison to independent object and part counting. In the wheat task, the statistic of interest for the breeder is not the spikelet count per specific spikes, but the *average* of this count over the entire piece of land captured in the image. Hence an alternative approach

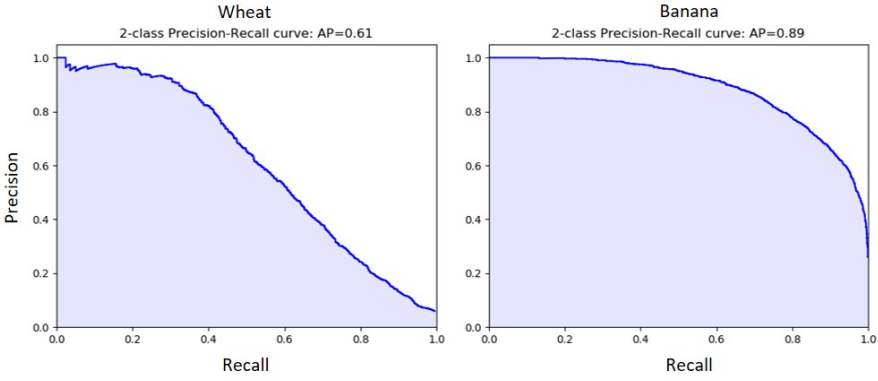

Fig. 4: Precision-Recall curves for parts detection, obtained by the D+R method. Unlike the direct regression, this method provides explicit localization of the detected parts, evaluated in these curves. In order to determine if a part detection is a hit or a miss we use the Percentage of Correct Keypoints (PCK) criterion, introduced in [53] and applied as in [21]. **Left:** Wheat spikelets. **Right:** Banana fruits in a bunch.

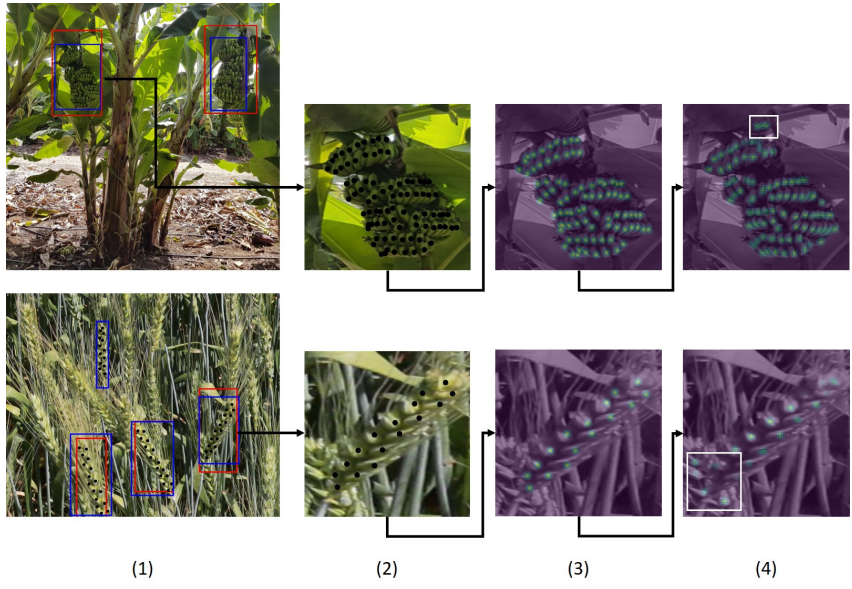

Fig. 5: Example of the network's stages performance with the D+R pipeline. **Top Row:** Banana, **Bottom Row:** Wheat. **(1).** Object detection (blue-ground truth, red-predictions),**(2).** Getting image crops of the detected objects, **(3)**. Ground truth Gaussisan heat map of the objects' parts, **(4).** The final predicted heat map of the parts' centers. White box shows the detections of unannotated parts, found by the predicted map.

Training Regime	Description	Mean Relative Deviation	1-FVU
T1	Fixed Test Detector	10.8%	0.745
T1 w. ST	Fixed Test Detector + scale and translation augmentation	10.7%	0.649
T1+	Test Detector at t=0 + subsequent learning	11.4%	0.588
TP	Perfect Detector	13.8%	0.479
TP w. ST	Perfect Detector + scale and translation augmentation	10.6%	0.637
L	Learning detector together with counter	11.6%	0.600

Table 4: Results of the MSR with P3-P5 counter ("fusing" three resolutions) trained with several procedures varying w.r.t the training detector. All methods were tested under the same conditions as in previous experiment, i.e. using the wheat object detector reported in Table 2. In bounding box noise addition, boxes provided by the detector were translated horizontally and vertically and scaled by random factors, drawn from the uniform distributions over $[-10:10]$, $[0.9:1.1]$ respectively (translation offsets are in pixels)

seems reasonable: counting the number of objects in the image (spike count), independently count the parts (spikelets), and then divide the two counts to get the average of interest. While this approach is simple, it suffers from two disadvantages relative to the two staged approach. First, it does not include a resize mechanism enlarging the relevant objects, hence detection of the small spikelets is more difficult. Second, not all spikes are 'countable' (due to scale and occlusion), and in many spikes only part of the spikelets can be observed. Independent counting of spikelets is therefore likely to count spikelets which do not belong to countable spikes, producing an over-estimation.

We empirically tested the subject by training a detector for independent detection of spikes and spikelets. An output example of this detector is presented in Fig. 6. For each image, an estimated value $\hat{y}_d$ of the average spikelets per spike was computed as the ratio of the detected spikelets count and the detected spikes count. To enable fair comparison with the two-stage method, which has internal ability for linear regression, a regressor of the form $\hat{y} = a\hat{y}_d + b$ was trained. The coefficients a, b were chosen to minimize the squared error expectation over the validation set $E_v[\hat{y} - y_{gt})^2]$. For the two stage method, an estimate of the average (over image) spikelets-per-spike was computed by simple averaging of the spikelet count over all detected spikes. Comparison between the two methods, presented in table 5, shows that the 2-stage method is significantly more accurate.

Physical Counting. While for countable wheat spikes the vast majority of the spikelets are visible, banana bunches are round objects, where typically at least half of the bananas are occluded. An additional dataset was collected to investigate the relation between actual and visual banana count, containing 30 banana bunches with known actual count, photographed from 3 viewpoints

Fig. 6: Ground truth (blue) and predicted (red) bounding boxes for spike and spikelets detected independently. The lack of object-part correspondence is clearly visible.

Method	Mean Relative Deviation	Pearson Correlation Coefficient
Two Stage	11.50%	0.937
Independent	17.82%	0.565

Table 5: Accuracy comparison between the two-stage and the independent detection methods.

Predicting from	Mean Relative Deviation	1-FVU
Visual ground truth	12.42%	0.554
Network predictions	12.42%	0.541

Table 6: Prediction of actual banana fruits count per bunch based on visual ground truth count, and based on network predictions.

each. Linear regression deviation was then estimated using a Leave-one-out cross validation procedure. This was done for inference of the actual count from the visual ground truth count, and (separately) from the network predicted count. The accuracy obtained is reported in table 6.

5 Conclusions and future work

We presented a two-staged network that successfully performs per-object part counting in terms of relative count deviation and 1-FVU measurements. It was demonstrated on two real world agricultural problems, counting banana fruits in a bunch and spikelets in a spike, with images taken in field conditions. Several technical improvements should be considered in future work, including replacing the non-differentiable RoI-Align module with a differentiable one, or replacing the ResNet based backbone with an improved alternative such as EfficientNet [46]. Beyond these, it would be interesting to test the suggested approach in additional part counting tasks.

References

1. Aich, S., Josuttes, A., Ovsyannikov, I., Strueby, K., Ahmed, I., Duddu, H.S., Pozniak, C., Shirtliffe, S., Stavness, I.: Deepwheat: Estimating phenotypic traits from crop images with deep learning. In: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 323–332. IEEE (2018)
2. Alharbi, N., Zhou, J., Wang, W.: Automatic counting of wheat spikes from wheat growth images (2018)
3. Alkhudaydi, T., Zhou, J., et al.: Spikeletfcn: Counting spikelets from infield wheat crop images using fully convolutional networks. In: International Conference on Artificial Intelligence and Soft Computing. pp. 3–13. Springer (2019)

4. Arad, B., Balendonck, J., Barth, R., Ben-Shahar, O., Edan, Y., Hellström, T., Hemming, J., Kurtser, P., Ringdahl, O., Tielen, T., et al.: Development of a sweet pepper harvesting robot. Journal of Field Robotics (2020)
5. Baharav, T., Bariya, M., Zakhor, A.: In situ height and width estimation of sorghum plants from 2.5 d infrared images. Electronic Imaging **2017**(17), 122–135 (2017)
6. Bargoti, S., Underwood, J.P.: Image segmentation for fruit detection and yield estimation in apple orchards. Journal of Field Robotics **34**(6), 1039–1060 (2017)
7. Bell, J., Dee, H.: Aberystwyth leaf evaluation dataset. URL: https://doi.org/10.5281/zenodo **168158**(17-36), 2 (2016)
8. Berenstein, R., Shahar, O.B., Shapiro, A., Edan, Y.: Grape clusters and foliage detection algorithms for autonomous selective vineyard sprayer. Intelligent Service Robotics **3**(4), 233–243 (2010)
9. Cholakkal, H., Sun, G., Khan, F.S., Shao, L.: Object counting and instance segmentation with image-level supervision. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
10. Dias, P.A., Tabb, A., Medeiros, H.: Apple flower detection using deep convolutional networks. Computers in Industry **99**, 17–28 (2018)
11. Dobrescu, A., Valerio Giuffrida, M., Tsaftaris, S.A.: Leveraging multiple datasets for deep leaf counting. In: Proceedings of the IEEE International Conference on Computer Vision Workshops. pp. 2072–2079 (2017)
12. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International Journal of Computer Vision **88**(2), 303–338 (Jun 2010)
13. Farjon, G., Krikeb, O., Hillel, A.B., Alchanatis, V.: Detection and counting of flowers on apple trees for better chemical thinning decisions. Precision Agriculture pp. 1–19 (2019)
14. Fernandez-Gallego, J.A., Kefauver, S.C., Gutiérrez, N.A., Nieto-Taladriz, M.T., Araus, J.L.: Wheat ear counting in-field conditions: high throughput and low-cost approach using rgb images. Plant Methods **14**(1), 22 (2018)
15. Fuentes, A., Yoon, S., Kim, S.C., Park, D.S.: A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition. Sensors **17**(9), 2022 (2017)
16. Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 1440–1448 (2015)
17. Hasan, M.M., Chopin, J.P., Laga, H., Miklavcic, S.J.: Detection and analysis of wheat spikes using convolutional neural networks. Plant Methods **14**(1), 100 (2018)
18. Haug, S., Ostermann, J.: A crop/weed field image dataset for the evaluation of computer vision based precision agriculture tasks. In: European Conference on Computer Vision. pp. 105–116. Springer (2014)
19. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017)
20. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
21. Itzhaky, Y., Farjon, G., Khoroshevsky, F., Shpigler, A., Bar-Hillel, A.: Leaf counting: Multiple scale regression and detection using deep cnns. In: BMVC. p. 328 (2018)

22. Kamilaris, A., Prenafeta-Boldú, F.X.: A review of the use of convolutional neural networks in agriculture. The Journal of Agricultural Science **156**(3), 312–322 (2018)
23. Kendall, A., Gal, Y.: What uncertainties do we need in bayesian deep learning for computer vision? In: Advances in neural information processing systems. pp. 5574–5584 (2017)
24. Kurtser, P., Ringdahl, O., Rotstein, N., Berenstein, R., Edan, Y.: In-field grape cluster size assessment for vine yield estimation using a mobile robot and a consumer level rgb-d camera. IEEE Robotics and Automation Letters **5**(2), 2031–2038 (2020)
25. Le, T.T., Lin, C.Y., et al.: Deep learning for noninvasive classification of clustered horticultural crops–a case for banana fruit tiers. Postharvest Biology and Technology **156**, 110922 (2019)
26. Lempitsky, V., Zisserman, A.: Learning to count objects in images. In: Advances in neural information processing systems. pp. 1324–1332 (2010)
27. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2117–2125 (2017)
28. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision. pp. 2980–2988 (2017)
29. Linker, R.: A procedure for estimating the number of green mature apples in nighttime orchard images using light distribution and its application to yield estimation. Precision Agriculture **18**(1), 59–75 (2017)
30. Liu, T., Wu, W., Chen, W., Sun, C., Zhu, X., Guo, W.: Automated image-processing for counting seedlings in a wheat field. Precision agriculture **17**(4), 392–406 (2016)
31. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: European conference on computer vision. pp. 21–37. Springer (2016)
32. Lu, H., Cao, Z., Xiao, Y., Zhuang, B., Shen, C.: Tasselnet: counting maize tassels in the wild via local counts regression network. Plant methods **13**(1), 79 (2017)
33. Madec, S., Jin, X., Lu, H., De Solan, B., Liu, S., Duyme, F., Heritier, E., Baret, F.: Ear density estimation from high resolution rgb imagery using deep learning technique. Agricultural and forest meteorology **264**, 225–234 (2019)
34. Minervini, M., Fischbach, A., Scharr, H., Tsaftaris, S.A.: Finely-grained annotated datasets for image-based plant phenotyping. Pattern recognition letters **81**, 80–89 (2016)
35. Neupane, B., Horanont, T., Hung, N.D.: Deep learning based banana plant detection and counting using high-resolution red-green-blue (rgb) images collected from unmanned aerial vehicle (uav). PloS one **14**(10) (2019)
36. Paul Cohen, J., Boucher, G., Glastonbury, C.A., Lo, H.Z., Bengio, Y.: Count-ception: Counting by fully convolutional redundant counting. In: Proceedings of the IEEE International Conference on Computer Vision Workshops. pp. 18–26 (2017)
37. Pound, M.P., Atkinson, J.A., Wells, D.M., Pridmore, T.P., French, A.P.: Deep learning for multi-task plant phenotyping. In: Proceedings of the IEEE International Conference on Computer Vision Workshops. pp. 2055–2063 (2017)
38. Qiongyan, L., Cai, J., Berger, B., Okamoto, M., Miklavcic, S.J.: Detecting spikes of wheat plants using neural networks with laws texture energy. Plant Methods **13**(1), 83 (2017)

39. Rahnemoonfar, M., Sheppard, C.: Deep count: fruit counting based on deep simulated learning. Sensors **17**(4), 905 (2017)
40. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 779–788 (2016)
41. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
42. Sa, I., Ge, Z., Dayoub, F., Upcroft, B., Perez, T., McCool, C.: Deepfruits: A fruit detection system using deep neural networks. Sensors **16**(8), 1222 (2016)
43. Santos, T.T., de Souza, L.L., dos Santos, A.A., Avila, S.: Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association. Computers and Electronics in Agriculture **170**, 105247 (2020)
44. Sindagi, V.A., Patel, V.M.: Generating high-quality crowd density maps using contextual pyramid cnns. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1861–1870 (2017)
45. Tan, M., Pang, R., Le, Q.: Efficientdet: Scalable and efficient object detection. arxiv 2019. arXiv preprint arXiv:1911.09070
46. Tan, M., Le, Q.V.: Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946 (2019)
47. Tan, M., Pang, R., Le, Q.V.: Efficientdet: Scalable and efficient object detection. arXiv preprint arXiv:1911.09070 (2019)
48. Turner, D., Mulder, J., Daniells, J.: Fruit numbers on bunches of bananas can be estimated rapidly. Scientia horticulturae **34**(3-4), 265–274 (1988)
49. Vit, A., Shani, G., Bar-Hillel, A.: Length phenotyping with interest point detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 0–0 (2019)
50. Wairegi, L., Van Asten, P., Tenywa, M., Bekunda, M.: Quantifying bunch weights of the east african highland bananas (musa spp. aaa-ea) using non-destructive field observations. Scientia horticulturae **121**(1), 63–72 (2009)
51. Wang, Z., Underwood, J., Walsh, K.B.: Machine vision assessment of mango orchard flowering. Computers and Electronics in Agriculture **151**, 501–511 (2018)
52. Xiong, H., Cao, Z., Lu, H., Madec, S., Liu, L., Shen, C.: Tasselnetv2: in-field counting of wheat spikes with context-augmented local regression networks. Plant Methods **15**(1), 150 (2019)
53. Yang, Y., Ramanan, D.: Articulated human detection with flexible mixtures of parts. IEEE transactions on pattern analysis and machine intelligence **35**(12), 2878–2890 (2012)
54. Zheng, Y.Y., Kong, J.L., Jin, X.B., Wang, X.Y., Su, T.L., Zuo, M.: Cropdeep: the crop vision dataset for deep-learning-based classification and detection in precision agriculture. Sensors **19**(5), 1058 (2019)
55. Zhou, C., Liang, D., Yang, X., Yang, H., Yue, J., Yang, G.: Wheat ears counting in field conditions based on multi-feature optimization and twsvm. Frontiers in plant science **9**, 1024 (2018)