CentroidNet: A Deep Neural Network for Joint Object Localization and Counting

K. Dijkstra[1,2(✉)], J. van de Loosdrecht[1], L. R. B. Schomaker[2], and M. A. Wiering[2]

[1] Centre of Expertise in Computer Vision and Data Science, NHL Stenden University of Applied Sciences, Leeuwarden, Netherlands
k.dijkstra@nhl.nl

[2] Department of Artificial Intelligence, Bernoulli Institute, University of Groningen, Groningen, Netherlands

Abstract. In precision agriculture, counting and precise localization of crops is important for optimizing crop yield. In this paper CentroidNet is introduced which is a Fully Convolutional Neural Network (FCNN) architecture specifically designed for object localization and counting. A field of vectors pointing to the nearest object centroid is trained and combined with a learned segmentation map to produce accurate object centroids by majority voting. This is tested on a crop dataset made using a UAV (drone) and on a cell-nuclei dataset which was provided by a Kaggle challenge. We define the mean Average F1 score (mAF1) for measuring the trade-off between precision and recall. CentroidNet is compared to the state-of-the-art networks YOLOv2 and RetinaNet, which share similar properties. The results show that CentroidNet obtains the best F1 score. We also explicitly show that CentroidNet can seamlessly switch between patches of images and full-resolution images without the need for retraining.

1 Introduction

Crop-yield optimization is an important task in precision agriculture. This agricultural output should be maximized while the ecological impact should be minimized. The state of crops needs to be monitored constantly and timely interventions for optimizing crop growth should be applied. The number of plants is an important indicator for the predicted yield. For a better indication of crop-yield the plants should be localized and counted during the growth season. This can potentially be done by using Unmanned Aerial Vehicles (UAVs) to record images and use Deep Learning to locate the objects.

Deep neural networks are trained using annotated image data to perform a specific task. Nowadays Convolutional Neural Networks (CNNs) are mainly used and have shown to achieve state-of-the-art performance. A wide range of applications benefit from deep learning and several general image processing tasks have emerged. A few examples are segmentation, classification, object detection

U. Brefeld et al. (Eds.): ECML PKDD 2018, LNAI 11053, pp. 585–601, 2019.
https://doi.org/10.1007/978-3-030-10997-4_36

[1] and image tag generation [11]. Recent methods focus on counting and localization [4]. In other cases, object counting is regarded as an object detection task with no explicit focus on counting or as a counting task with no explicit focus on localization [2].

Many object detection architectures exist which vary in several regards. In two-stage detectors like Faster R-CNN [15], a first-stage network produces a sparse set of candidate objects which are then classified in a second stage. One-stage detectors like SSD [8], YOLOv2 [14] and RetinaNet [7] use a single stage to produce bounding boxes in an end-to-end fashion. If an object detection network is fully convolutional it can handle images of varying sizes naturally and is able to adopt a Fully Convolutional Network (FCN) as a backbone [9]. The aforementioned networks are all regarded to be fully convolutional, but rely on special subnetworks to produce bounding boxes.

This paper introduces CentroidNet which has been specifically designed for joint object counting and localization. CentroidNet produces centroids of image objects rather than bounding boxes. The key idea behind CentroidNet is to combine image segmentation and centroid majority voting to regress a vector field with the same resolution as the input image. Each vector in the field points to its relative nearest centroid. This makes the CentroidNet architecture independent of image size and helps to make it a fully-convolutional object counting and localization network. Our idea is inspired by a random-forest based voting algorithm to predict locations of body joints [12] and detect centroids of cells in medical images [5]. By binning the summation of votes and by applying a non-max suppression our method is related to the Hough transform which is known to produce robust results [10].

CentroidNet is a fully-convolutional one-stage detector which can adopt an FCN as a backbone. We choose a U-Net segmentation network as a basis because of its good performance [16]. The output of U-Net is adapted to accommodate CentroidNet. Our approach is compared with state-of-the-art on-stage object-detection networks. YOLOv2 is chosen because of its maturity and popularity. RetinaNet is chosen because it outperforms other similar detectors [7].

A dataset of crops with various sizes and heavy overlap has been created to compare CentroidNet to the other networks. It is produced by a low-cost UAV with limited image quality and limited ground resolution. Our hypothesis is that a small patch of an image of a plant naturally contains information about the location of its centroid which makes a convolutional architecture suitable for the task. Because the leaves of a plant tend to grow outward they implicitly point inward to the location of the centroid of the nearest plant. Our rationale is that this information in the image can be exploited to learn the vectors pointing to the center of the nearest plant. Centroids are calculated from these vectors.

An additional dataset containing microscopic images of cell nuclei for the Kaggle Data Science Bowl 2018[1] is used for evaluating the generality and fully convolutional properties of CentroidNet.

[1] https://www.kaggle.com/c/data-science-bowl-2018.

Object detection networks are generally evaluated by the mean Average Precision (mAP) which focuses mainly on minimizing false detections and therefore tends to underestimate object counts. In contrast to this, the mean Average Recall (mAR) can also be used, but tends to overestimate the number of counted objects because of the focus on detecting all instances. We define the mean average F1 (mAF1) score to determine the trade-off between overestimation and underestimation. Similarly to the regular F1 score the mAF1 score is defined as the harmonic mean between the mAP and mAR. For convenience throughout this paper the terms Precision (P), Recall (R) and F1 score are used instead of mAP, mAR and mAF1.

The remainder of this paper is structured as follows. Section 2 introduces the datasets used in the experiments. In Sect. 3, the CentroidNet architecture is explained in detail. Sections 4 and 5 present the experiments and the results. In Sect. 6 the conclusion and directions for future work are discussed.

2 Datasets

The crops dataset is used for comparing CentroidNet to the state-of-the-art networks. The nuclei dataset is used to test the generality of CentroidNet. Both sets will be used to test the fully-convolutional properties of CentroidNet.

2.1 Crops

This dataset contains images of potato plants. It was recorded by us during the growth season of 2017 in the north of the Netherlands. A Yuneec Typhoon 4K Quadrocopter was used to create a video of crops from a height of approximately 10 m. This produced a video with a resolution of 3840×2160 pixels at 24 fps. From this original video, 10 frames were extracted which contain a mix of overlapping plants, distinct plants and empty patches of soil. The borders of each image were removed because the image quality close to borders is quite low because of the wide angle lens mounted on the camera. The cropped images have a resolution of 1800×1500 pixels. Two domain experts annotated bounding boxes of potato plants in each of the images. This set is split into a training and validation set each containing 5 images and over 3000 annotated potato plant locations. These sets are referred to as 'crops-full-training' and 'crops-full-validation'.

The networks are trained using small non-overlapping patches with a resolution of 600×500 pixels. Patches are used because these neural networks use a large amount of memory on the GPU and reducing the image size will also reduce the memory consumption. In our case the original set of 10 images is subdivided into a new set of 90 images. It is well known that by training on more images the neural networks generalize better. Also more randomness can be introduced when drawing mini-batches for training because the pool of images to choose from is larger. An additional advantage of using small patches is that CentroidNet can be trained on small patches and validated on the full-resolution

images to measure the impact of using CentroidNet as a fully convolutional network. The sets containing the patches are referred to as 'crops-training' and 'crops-validation'. In Fig. 1 some example images of these datasets are shown. To provide a fair comparison between networks and to reduce the amount of detection errors for partially-observed plants the bounding boxes that are too close to the borders have been removed.

Fig. 1. Three images of the 'crops-training' and 'crops-validation' dataset. The images show an overlay with bounding boxes annotated by one of the experts. The annotations are filtered by removing objects too close to the border.

The bounding-box annotations are converted into a three-channel image that will be used as a target image for training. The first two channels contain the x and y components of vectors pointing to the nearest centroid of a crop (center of its bounding box). The third channel is generated by drawing binary ellipses in the annotated bounding-boxes. More binary maps can be added if more classes are present in the image. This means that the target image contains information about the centroid locations in each pixel and the class of each pixels is known. These three channels help CentroidNet to be a robust centroid detector.

2.2 Kaggle Data Science Bowl 2018

The generality of CentroidNet will be tested on the dataset for the Data Science Bowl 2018 on Kaggle.com. This set is referred to as 'nuclei-full'. The challenge is to create an algorithm that automates the detection of the nucleus in several microscopic images of cells. This dataset contains 673 images with a total of 29,461 annotated nuclei (see Fig. 2). The images vary in resolution, cell type, magnification, and imaging modality. These properties make this dataset particularly interesting for validating CentroidNet. Firstly the variation in color and size of the nuclei makes it ideal for testing a method based on deep learning. Secondly the image sizes vary to a great extent making it suitable for testing our fully-convolutional network by training on smaller images and validation on the full-resolution images.

Because fixed-size tensors are more suitable for training on a GPU the 'nuclei-full' dataset is subdivided into patches of 256×256 pixels. Each patch overlaps with the neighboring patches by a border of 64 pixels in all directions.

This dataset of patches is split into 80% training and 20% validation. These datasets are referred to as 'nuclei-training' and 'nuclei-validation'. The fusion of the results on these patches is not required because the trained network is also applied to the original images in 'nuclei-full' to produce full-resolution results.

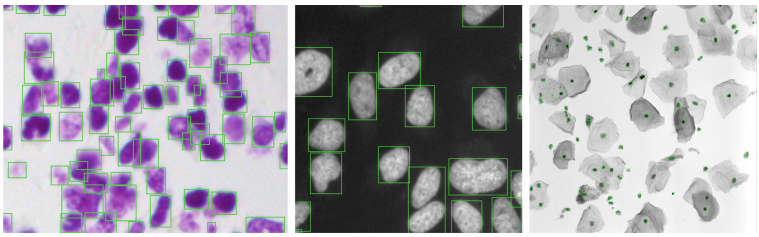

Fig. 2. Three images from 'nuclei-full' set. These images give an indication of the variation encountered in this dataset. The green overlay shows annotated bounding boxes. The right image shows that these bounding boxes can be very small.

While this nuclei-detection application is ideal to validate our method it is difficult to directly compare our approach to the other participants. The original challenge is defined as a pixels-precise segmentation challenge. We redefine this problem as an object localization and counting challenge. This means that the size of the bounding boxes needs to be fixed and that the mean F1 score will be used as an evaluation metric.

A three-channel target image is created for training CentroidNet. The creation method is identical to the one described in the previous subsection.

3 CentroidNet

The input image to CentroidNet can be of any size and can have an arbitrary number of color channels. The output size of CentroidNet is identical to its input size and the number of output channels is fixed. The first two output channels contain the x and the y component of a vector for each pixel. These vectors each point to the nearest centroid of an object and can thus be regarded as votes for where an object centroid is located. All votes are aggregated into bins represented by a smaller image. A pixel in this voting map has a higher value if there is greater chance of centroid presence. Therefore a local maximum represents the presence of a centroid. The remaining channels of the output of CentroidNet contain the logit maps for each class. This is identical to the per-pixel-one-hot output of a semantic segmentation network [9]. For the crops and the nuclei datasets only one such map exists because there is only one class (either object or no object). By combining this map with the voting map an accurate and robust estimation can be made of the centroids of the objects.

In our experiments we use a U-Net implemented in PyTorch as a basis[2] for CentroidNet. In the downscaling pathway the spatial dimensions (width and height) of the input are iteratively reduced in size and the size of the channel dimension is increased. This is achieved by convolution and max-pooling operations. Conversely, in the upscaling pathway the tensor is restored to its original size by deconvolution operations. The intermediate tensors from the downscaling pathway are concatenated to the intermediate tensors of the upscaling pathway to form "horizontal" connections. This helps to retain the high-resolution information.

Theoretically any fully-convolutional network can be used as a basis for CentroidNet as long as the spatial dimensions of the input and the output are identical. However there are certain advantages to employing this specific CNN architecture. By migrating information from spatial dimensions to spectral dimensions (the downscaling pathway) an output vector should be able to vote more accurately over larger distances in the image which helps voting robustness. By concatenating tensors from the downscaling pathway to the upscaling pathway a sharper logit map is created. This also increases the accuracy of the voting vectors and makes the vector field appear sharper which results in more accurate centroid predictions.

The next part explains the details of the CNN architecture. The first part explains the downscaling pathway of CentroidNet and then the upscaling pathway is explained. The final part explains the voting algorithm and the method used to combine the logit map and the voting result to produce final centroid locations.

A CNN consists of multiple layers of convolutional-filter banks that are applied to the input tensor (or input image). A single filter bank is defined as a set of convolutional filters:

$$\mathcal{F}_n^{t\times t} = \{\mathbf{F}_1^{t\times t}, \mathbf{F}_2^{t\times t}, \ldots, \mathbf{F}_n^{t\times t}\} \tag{1}$$

where $\mathcal{F}_n^{t\times t}$ is a set of n filters with a size of $t \times t$. Any convolutional filter in this set is actually a 3-d filter with a depth equal to the number of channels of the input tensor.

The convolutional building block of CentroidNet performs two 3×3 convolution operations on the input tensor and applies the Rectified Linear Unit (ReLU) activation function after each individual operation. The ReLU function clips values below zero and is defined as $\psi(\mathbf{X}_{yx}) = \max(0, \mathbf{X}_{yx})$ where $\mathbf{X}$ is the input tensor. We found that input scaling is not required if ReLU is used as an activation function. Scaling is required if a saturating function like the hyperbolic tangent would be used as an activation function. The convolution operator $\otimes$ takes an input tensor on the left-hand side and a set of convolution filters on the right-hand side. The convolution block is defined by

$$\mathrm{conv}(\mathbf{X}, c) = \psi(\psi(\mathbf{X} \otimes \mathcal{F}_c^{3\times 3}) \otimes \mathcal{F}_c^{3\times 3}) \tag{2}$$

[2] https://github.com/jaxony/unet-pytorch.

where $\mathbf{X}$ is the input tensor, c is the number of filters in the convolutional layer, and $\otimes$ is the convolution operator.

The downscaling pathway is defined as multiple $\text{conv}(\cdot,\cdot)$ and max-pooling, $\text{pool}(\cdot)$, operations. The initial convolutional operator increases the depth of the input image from 3 (RGB) to 64 channels and reduces the height and width by a max-pooling operation of size 2×2. In subsequent convolutional operations the depth of the input tensor is doubled by increasing the amount of convolutional filters and the height and width of the input tensor is reduced by 1/2 with a max-pooling operation. This is a typical CNN design pattern which results in converting spatial information to semantic information. In CentroidNet this also has the implicit effect of combining voting vectors from distant parts of the image in a hierarchical fashion. The downscaling pathway is mathematically defined as:

$$\mathbf{C}_1 = \text{conv}(\mathbf{X}, 64) \tag{3}$$
$$\mathbf{D}_1 = \text{pool}(\mathbf{C}_1) \tag{4}$$
$$\mathbf{C}_2 = \text{conv}(\mathbf{D}_1, 128) \tag{5}$$
$$\mathbf{D}_2 = \text{pool}(\mathbf{C}_2) \tag{6}$$
$$\mathbf{C}_3 = \text{conv}(\mathbf{D}_2, 256) \tag{7}$$
$$\mathbf{D}_3 = \text{pool}(\mathbf{C}_3) \tag{8}$$
$$\mathbf{C}_4 = \text{conv}(\mathbf{D}_3, 512) \tag{9}$$
$$\mathbf{D}_4 = \text{pool}(\mathbf{C}_4) \tag{10}$$
$$\mathbf{D}_5 = \text{conv}(\mathbf{D}_4, 1024) \tag{11}$$

where $\mathbf{X}$ is the input tensor, $\mathbf{C}_x$ are the convolved tensors, and $\mathbf{D}_x$ are the downscaled tensors. The convolved tensors are needed for the upscaling pathway. The final downscaled tensor $\mathbf{D}_5$ serves as an input to the upscaling pathway.

The upscaling pathway incrementally restores the tensor back to the original image size by deconvolution operations. This is needed because the output tensor should have the same size as the input image to be able to produce one voting vector per pixel. The $\text{up}(\cdot)$ operation first performs the $\text{conv}(\cdot,\cdot)$ operation defined in Eq. 2 and then performs a deconvolution $\oslash$ operation with a filter of 2×2 that doubles the height and width of the input tensor.

$$\text{up}(\mathbf{X}, c) = \psi(\mathbf{X} \oslash \mathcal{F}_c^{2\times 2}) \tag{12}$$

where $\mathbf{X}$ is the input tensor, c is the number of filters in the deconvolutional layer, and $\oslash$ is the deconvolution operator.

The final part of the CNN is constructed by subsequently upscaling the output tensor of the downscaling pathway $\mathbf{D}_5$. The width and height are doubled and the depth is halved by reducing the amount of convolution filter in the filter bank. The upscaling pathway is given additional high-resolution information in "horizontal" connections between the downscaling and upscaling pathways. This should result in more accurate voting vectors. Before each tensor was subsequently downscaled it was stored as $\mathbf{C}_x$. These intermediate tensors are concatenated to tensors of the same size produced by the upscaling pathway. The operator $\oplus$ concatenates the left-hand-side tensor to the right-hand-side tensor over the depth axis.

$$\mathbf{U}_1 = \text{conv}(\text{up}(\mathbf{D}_5, 512) \oplus \mathbf{C}_4, 512) \tag{13}$$

$$\mathbf{U}_2 = \text{conv}(\text{up}(\mathbf{U}_1, 256) \oplus \mathbf{C}_3, 256) \tag{14}$$

$$\mathbf{U}_3 = \text{conv}(\text{up}(\mathbf{U}_2, 128) \oplus \mathbf{C}_2, 128) \tag{15}$$

$$\mathbf{U}_4 = \text{conv}(\text{up}(\mathbf{U}_3, 64) \oplus \mathbf{C}_1, 64) \tag{16}$$

$$\mathbf{Y} = \mathbf{U}_4 \otimes \mathcal{F}_3^{1\times 1} \tag{17}$$

where $\mathbf{D}_5$ is the smallest tensor from the downscaling pathway, $\mathbf{U}_x$ are the upscaled tensors, $\oplus$ is the tensor concatenation operator, and the final tensor with its original width and height restored and its number of channels set to 3 is denoted by $\mathbf{Y}$.

The concatenation operator fails to function properly if the size of the input image cannot be divided by 2^5 (e.g. the original input image size cannot be divided by two for five subsequent times). To support arbitrary input-image sizes the tensor concatenation operator $\oplus$ performs an additional bilinear upscaling if the dimensions of the operands are not equal due to rounding errors.

The final 1×1 convolution operation in Eq. 17 is used to set the number of output channels to a fixed size of three (x and y vector components and an additional map of class logits for the object class). It is important that no ReLU activation function is applied after this final convolutional layer because the output contains relative vectors which should be able to have negative values (i.e. a centroid can be located anywhere relative to an image coordinate).

To produce centroid locations from the outputs of the CNN the votes have to be aggregated. The algorithm for this is shown in Algorithm 1. A centroid vote is represented by a relative 2-d vector at each image location. First, all relative vectors in the CNN output $\mathbf{Y}$ are converted to absolute vectors by adding the absolute image location to the vector value. Then for each vote the bin is calculated by performing an integer division of the vector values. In preliminary research we found a bin size of 4 to perform well. By binning the votes the result is more robust and the influence of noise in the vectors is reduced. Finally, the voting matrix $\mathbf{V}$ is incremented at the calculated vector location. When an absolute vector points outside of the image the vote is discarded. The voting map is filtered by a non-max suppression filter which only keeps the voting maxima (the peaks in the voting map) in a specific local neighborhood.

Algorithm 1. Voting algorithm

$\mathbf{Y} \leftarrow$ Output of CNN ▷ The 1st and 2nd channel contain the voting vectors
$h, w \leftarrow$ height, width of $\mathbf{Y}$
$\mathbf{V} \leftarrow$ zero filled matrix of size (h div 4, w div 4)
for $y \leftarrow 0$ **to** $h - 1$ **do**
  for $x \leftarrow 0$ **to** $w - 1$ **do**
    $y' \leftarrow (y + \mathbf{Y}[y, x, 0])$ div 4 ▷ Get the absolute-binned y component
    $x' \leftarrow (x + \mathbf{Y}[y, x, 1])$ div 4 ▷ Get the absolute-binned x component
    $\mathbf{V}[y', x'] \leftarrow \mathbf{V}[y', x'] + 1$ ▷ Aggregate vote
  end for
end for

Finally the voting map is thresholded to select high votes. This results is a binary voting map. Similarly the output channel of the CNN which contains the class logits is also thresholded (because we only have one class this is only a single channel). This results in the binary segmentation map. Both binary maps are multiplied to only keep centroids at locations where objects are present. We found that this action reduces the amount of false detections strongly.

$$\mathbf{V}_{y,x} = 1 \text{ if } \mathbf{V}_{y,x} \geq \theta, 0 \text{ otherwise.} \tag{18}$$

$$\mathbf{S}_{y,x} = 1 \text{ if } \mathbf{Y}_{y,x,3} \geq \gamma, 0 \text{ otherwise.} \tag{19}$$

$$\mathbf{V} = \text{enlarge}(\mathbf{V}, 4) \tag{20}$$

$$\mathbf{C}_{y,x} = \mathbf{V}_{y,x} \times \mathbf{S}_{y,x} \tag{21}$$

where $\mathbf{Y}$ is the output of the CNN, $\mathbf{V}$ is the binary voting map, $\mathbf{S}$ is the binary segmentation map, and θ and γ are two threshold parameters. The final output $\mathbf{C}$ is produced by multiplying each element of $\mathbf{V}$ with each element of $\mathbf{S}$ to only accept votes at object locations. The vote image needs to be enlarged because it was reduced by the binning of the votes in Algorithm 1.

4 Experiments

CentroidNet is compared to state-of-the-art object detection networks that share their basic properties of being fully convolutional and the fact that they can be trained in one stage. The YOLOv2 network uses a backbone which is inspired by GoogleNet [13,17] (but without the inception modules). It produces anchor boxes as outputs that are converted into bounding boxes [14]. A cross-platform implementation based on the original DarkNet is used for the YOLOv2 experiments[3]. RetinaNet uses a Feature Pyramid Network [6] and ResNet50 [3] as backbone. Two additional convolutional networks are stacked on top of the backbone to produce bounding boxes and classifications [7]. A Keras implementation of RetinaNet is used in the experiments[4].

The goal of this first experiment is to compare networks. The compared networks are all trained on the 'crops-training' dataset which contains 45 image patches containing a total of 996 plants. The networks are trained to convergence. Early stopping was applied when the loss on the validation set started to increase. The 'crops-validation' set is used as a validation set and contains 45 image patches containing a total of 1090 potato plants. CentroidNet is trained using the Euclidean loss function, the Adam optimizer and a learning rate of 0.001 for 120 epochs. For the other networks mainly the default settings were used. RetinaNet uses the focal loss function proposed in their original paper [7] and an Adam optimizer with a learning rate of 0.00001 for 200 epochs. YOLOv2 was trained with stochastic gradient descent and a learning rate of 0.001 for 1000 epochs. RetinaNet uses pretrained weights from ImageNet and YOLOv2 uses pretrained weights from the Pascal VOC dataset.

[3] https://github.com/AlexeyAB/darknet.
[4] https://github.com/fizyr/keras-retinanet.

The goal of the second experiment is to test the full-convolutional properties of CentroidNet. What is the effect when the network is trained on image patches and validated on the set of full-resolution images without retraining? In the third experiment CentroidNet is validated on the nuclei dataset. Training is performed using the 'nuclei-training' dataset which contains 6081 image patches. The 'nuclei-validation' which contains 1520 image patches is used to validate the results. CentroidNet is also validated using all 637 full-resolution images from the 'nuclei-full' dataset without retraining. In this case CentroidNet was trained to convergence with the Adam optimizer with a learning rate of 0.001 during 1000 epochs. The Euclidean loss was used during training. Early stopping was not required because the validation results were stable after 1000 epochs.

The next part explains how the bounding-boxes produced by the networks are compared to the target bounding boxes. The Intersection of Union (IoU) represents the amount of overlap between two bounding boxes, where an IoU of zero means no overlap and a value of one means 100% overlap:

$$\mathrm{IoU}(\mathcal{R}, \mathcal{T}) = \frac{\mathcal{R} \cap \mathcal{T}}{\mathcal{R} \cup \mathcal{T}} \tag{22}$$

where $\mathcal{R}$ is a set of pixels of a detected object and $\mathcal{T}$ is a set of target object pixels. Note that objects are defined by their bounding box, which allows for an efficient calculation of the IoU metric.

When the IoU for a target and a result object have sufficiently high overlap, the object is greedily assigned to this target and counted as a True Positive (TP). False Positives (FP) and False Negatives (FN) are calculated as follows:

$$\mathrm{TP} = \mathrm{count\_if}(\mathrm{IoU}(\mathcal{R}, \mathcal{T}) > \tau) \qquad \forall \mathcal{R}, \mathcal{T} \tag{23}$$

$$\mathrm{FP} = \#\mathrm{results} - \mathrm{TP} \tag{24}$$

$$\mathrm{FN} = \#\mathrm{targets} - \mathrm{TP} \tag{25}$$

where #results and #targets are the number of result and target objects, and τ is a threshold parameters for the minimum IoU value.

All metrics are defined in terms of these basic metrics. Precision (P), Recall (R) and F1 score are defined as

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{26}$$

$$\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{27}$$

$$\mathrm{F1} = 2 \times \frac{\mathrm{P} \times \mathrm{R}}{\mathrm{P} + \mathrm{R}} \tag{28}$$

In Eq. 28 it can be seen that the F1 score gives a trade-off between precision and recall, and thus measures the equilibrium between the overestimation and the underestimation of object counts. The size of the bounding boxes of both the target and the result objects will be set to a fixed size of roughly the size of an object. This fixed size of the bounding box can be interpreted as the

size of the neighborhood around a target box in which a result bounding box can still be counted as a true positive. This could possibly be avoided by using other metrics like absolute count and average distance. However, absolute count does not estimate localization accuracy and average distance does not measure counting accuracy. Therefore the F1 score is used to focus the validation on joint object localization and on object counting.

For the crops dataset the bounding-box size is set to 50×50 pixels. For the nuclei dataset a fixed bounding box size of 60×60 pixels is chosen. This size is chosen so that there is not too much overlap between bounding boxes. When boxes are too large the greedy IoU matching algorithm in Eq. 23 could assign result bounding boxes to a target bounding box which is too far away.

The best voting threshold θ (Eq. 18) is determined using the training set. The segmentation threshold γ in Eq. 19 is set to zero for all experiments (result values in the segmentation channel can be negative), and the performance metrics for several IoU thresholds τ (Eq. 23) will be reported for each experiment.

5 Results

This section discusses the results of the experiments. The first subsection shows the comparison between CentroidNet, YOLOv2 and RetinaNet on the 'crops' dataset. The second subsection explores the effect of enlarging the input image size after training CentroidNet on either the 'crops' or the 'nuclei' dataset.

5.1 Comparison with the State-of-the Art on the Crops Dataset

The results for the best F1 score with respect to the voting threshold θ are shown in Table 1. The results show that CentroidNet achieves a higher F1 score on the 'crops-validation' set regardless of the chosen IoU and regardless of the expert (90.4% for expert A, and 93.4% for expert B). For both experts RetinaNet and YOLOv2 obtain lower F1 scores. Interestingly the performance measured for expert B is higher compared to expert A. This is probably because of the higher quality of the annotations produced by expert B. There is an optimal IoU threshold. When the IoU threshold is chosen too low the validation results are adversely affected. This is probably due to the greedy matching scheme involved in calculating the F1 score. Therefore the intuition that a smaller IoU threshold yields higher validation scores seems unfounded.

Table 1. F1 score for both experts on the 'crops-validation' dataset

Using annotations of expert A and using annotations of expert B

IoU (τ)	0.1	0.2	0.3	0.4	0.5
CentroidNet	90.0	**90.4**	90.0	89.1	85.7
RetinaNet	88.3	89.1	**89.4**	87.7	83.1
YOLOv2	87.1	88.4	**88.8**	87.5	82.0

IoU (τ)	0.1	0.2	0.3	0.4	0.5
CentroidNet	93.0	93.2	**93.4**	92.7	90.0
RetinaNet	90.5	90.9	**91.1**	90.7	89.2
YOLOv2	88.3	88.9	**89.1**	89.0	87.2

The results can be further analyzed by the precision-recall graph shown in Fig. 3. The red curve of CentroidNet is generated by varying the voting threshold θ between 0 and 1024. The curves for YOLOv2 (Blue) and RetinaNet (Green) have been generated by varying the confidence threshold between 0 and 1. The curve of CentroidNet passes the closest to the right-top corner of the graph with a precision of 91.2% and a recall of 95.9% for expert B.

When using the annotated set of expert B (Fig. 3-right), RetinaNet and YOLOv2 show similar recall values when exiting the graph at the most left. CentroidNet and RetinaNet show a similar precision when exiting the graph at the bottom. The precision-recall graph for Expert A (Fig. 3-left) shows that RetinaNet has better precision at the cost of low recall, but the best precision-recall value is observed for CentroidNet.

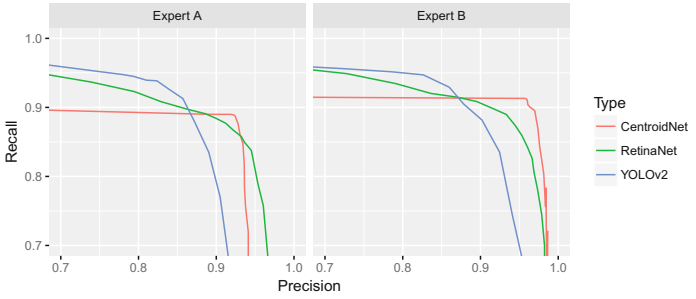

Fig. 3. Precision-recall graphs for expert A (left) and expert B (right). These graphs show that CentroidNet performs best on the trade-off between precision and recall. (Color figure online)

An image from the 'crops-validation' set is used to show detailed results of the inner workings of CentroidNet. The images for the regression output of CentroidNet are shown in the top row of Fig. 4. The left image shows the input image which gives an idea of the challenge this image poses with respect to the image quality and overlap between plants. The middle image of the top row of Fig. 4 shows the magnitude of the target voting vectors (dark is shorter). The right image of the top row shows the magnitude of the learned voting vectors. Important aspects like the length of the vectors and the ridges between objects can be observed in the learned vectors. Interestingly, CentroidNet is also able to learn large vectors for locations which do not contain any green plant pixels.

The bottom row of Fig. 4 shows the final result of CentroidNet. After aggregating the vectors and thresholding the voting map the binary voting map is produced which is shown in the bottom-left image of Fig. 4. The bright dots show where most of the voting vectors point to (the binary voting map). The blue areas show the binary segmentation map which has been used to filter false detections. By converting each centroid to a fixed-size bounding box the bottom-right image is produced. It can be seen that the plants are detected even with

heavy overlap (green boxes). In this result a false positive (red box) is caused by an oddly shaped plant group. A false negative is caused by a small undetected plant (blue box).

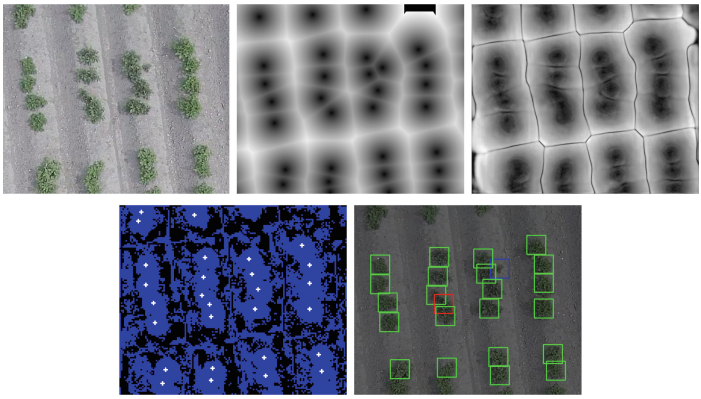

Fig. 4. Result of CentroidNet with the top row showing the input, the target vector magnitudes, and the result vector magnitudes. The bottom-left image shows the binary voting map as bright dots and the binary segmentation map as the blue area (dark). The bottom-right image shows the bounding boxes (Green = true positive, Red = false negative, Blue = false positive). Note that boxes too close to the border are not considered. (Color figure online)

5.2 Testing on Larger Images

A strong feature of CentroidNet is that it is a fully convolutional network. The CentroidNet from the previous subsection which was trained on image patches is used here. Validation is performed on the full resolution images of 1800×1500 pixels. In Table 2 the performance on the image patches in the 'crops-validation' and the full-resolution-image dataset 'crops-full-validation' are shown. The F1 score for expert A goes from 90.4% to 88.4%, which slightly less then the best performance of YOLOv2 in Table 1. The performance on the full-resolution dataset of expert B goes from 93.4% to 91.7%. This is still the best overall performance with respect to the results of the other object detection networks in Table 1. This means that the drop in performance by applying CentroidNet to the full-resolution dataset without retraining is acceptable.

The results of the performance of CentroidNet on the 'nuclei' datasets is shown in Table 3. CentroidNet is trained and validated with the patches from the 'nuclei-training' and 'nuclei-validation' sets. The network is tested with the full-resolution 'nuclei-full' dataset.

CentroidNet shows high precision at the cost of a lower recall. The highest F1 score is obtained on the full-resolution dataset (86.9%). Because the network was not trained on the full-resolution dataset this seems counter intuitive. In Fig. 5

Table 2. CentroidNet F1 score for expert A and B on the crops dataset

IoU (τ)	0.1	0.2	0.3	0.4	0.5
Expert A					
'crops-validation' (patches)	90.0	**90.4**	90.0	89.1	85.7
'crops-full-validation' (full-res)	87.6	**88.4**	87.8	84.8	77.3
Expert B					
'crops-validation' (patches)	93.0	93.2	**93.4**	92.7	90.0
'crops-full-validation' (full-res)	90.2	91.2	**91.7**	89.8	85.0

detected centroids from the 'nuclei-validation' and the 'nuclei-full' datasets are shown. The occluded nuclei at the borders are prone to becoming false detections. Because the full-resolution images have less objects at the border the higher performance is explained. This shows that CentroidNet performs well as a fully-convolutional network that can be trained on image patches and then successfully be applied to full-resolution images.

Table 3. CentroidNet F1 score, Precision and Recall on the nuclei dataset.

IoU (τ)	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
F1 score									
'nuclei-training' (patches)	81.3	82.0	82.8	83.3	83.5	**83.5**	83.4	74.0	27.8
'nuclei-validation' (patches)	79.2	79.6	79.9	**80.3**	80.1	79.5	77.7	63.4	24.4
'nuclei-full' (full-res)	84.4	85.7	86.5	87.0	86.8	**86.9**	85.7	74.7	33.0
Precision									
'nuclei-training' (patches)	97.3	98.2	99.1	99.7	99.9	**100.0**	99.9	88.6	33.3
'nuclei-validation' (patches)	94.9	95.3	95.8	**96.2**	96.0	95.3	93.1	76.0	29.3
'nuclei-full' (full-res)	94.9	96.3	97.4	**97.9**	97.6	97.8	96.5	84.0	37.1
Recall									
'nuclei-training' (patches)	69.8	70.4	71.1	71.5	71.6	**71.7**	71.6	63.5	23.8
'nuclei-validation' (patches)	68.0	68.3	68.6	**68.9**	68.7	68.2	66.7	54.4	20.9
'nuclei-full' (full-res)	75.9	77.2	77.8	**78.3**	78.2	78.2	77.1	67.3	29.7

6 Discussion and Conclusion

In this paper CentroidNet, a deep neural network for joint localization and counting, was presented. A U-Net [16] architecture is used as a basis. CentroidNet is trained to produce a set of voting vectors which point to the nearest centroid of an object. By aggregating these votes and combining the result with the segmentation mask, also produced by CentroidNet, state-of-the art performance is

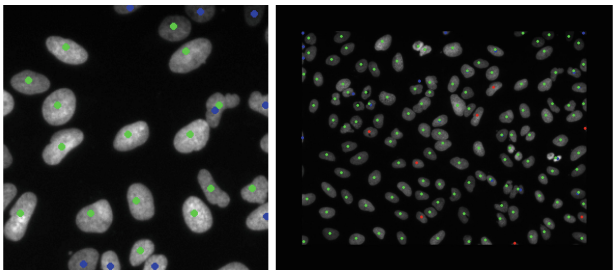

Fig. 5. The left image is a patch from the right image. The left image shows the centroid detection results from and image of the 'nuclei-validation' dataset and the right image shows the centroid detection results on an image from the 'nuclei-full' dataset. Green = true positive, Red = false negative, Blue = false positive (Color figure online)

achieved. Experiments were performed using a dataset containing images of crops made using a UAV and on a dataset containing microscopic images of nuclei. The mean Average F1 score (mAF1) which is the harmonic mean between precision and recall was used as a main evaluation metric because it gives a good indication of the trade-off between underestimation and overestimation in counting and a good estimation of localization performance.

The best performance for the joint localization and counting of objects is obtained using CentroidNet with an F1 score of 93.4% on the crops dataset. In comparison to other object detection networks with similar properties the results were 91.1% for RetinaNet and YOLOv2 obtained and F1 score of 89.1%.

CentroidNet has been tested by training on patches of images and by validating on full-resolution images. On the crops dataset the best F1 score dropped from 93.4% to 91.7%, which still made CentroidNet the best performing network. For the nuclei dataset the F1 score on the full-resolution images was highest, which can be attributed to border effects.

Generally we learned that using a majority voting scheme for detecting object centroids produces robust results with regard to the trade-off between precision and recall. By using a trained segmentation mask to suppress false detection, a higher precision is achieved, especially on the low-quality images produced by drones. A relatively small amount of images can be used for training because votes are largely independent.

Although CentroidNet is the best-performing method with respect to the posed problem. Improvements can still be made in future research. The detection of bounding boxes or instance segmentation maps can be explored, multi-class problems can be investigated or research could focus on reducing the border-effects. Future research could also focus on testing CentroidNet on larger potato-plant datasets or look into localization and counting of other types of vegetation like sugar beets, broccoli or trees using images taken with a UAV. On a more detailed scale the detection of vegetation diseases could be investigated by

detecting brown lesions on plant leaves or by looking at detection problems on a microscopic scale that are similar to nuclei detection.

References

1. Cheema, G.S., Anand, S.: Automatic detection and recognition of individuals in patterned species. In: Altun, Y., et al. (eds.) ECML PKDD 2017. LNCS (LNAI), vol. 10536, pp. 27–38. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-71273-4_3
2. Cohen, J.P., Boucher, G., Glastonbury, C.A., Lo, H.Z., Bengio, Y.: Count-ception: counting by fully convolutional redundant counting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 18–26 (2017)
3. 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 (2015)
4. Hsieh, M.R., Lin, Y.L., Hsu, W.H.: Drone-based object counting by spatially regularized regional proposal network. In: The IEEE International Conference on Computer Vision (ICCV), vol. 1 (2017)
5. Kainz, P., Urschler, M., Schulter, S., Wohlhart, P., Lepetit, V.: You should use regression to detect cells. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 276–283. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_33
6. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Conference on Computer Vision and Pattern Recognition, vol. 1, p. 4 (2017)
7. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. arXiv preprint arXiv:1708.02002 (2017)
8. Liu, W., et al.: SSD: single shot multibox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 21–37. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_2
9. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431–3440 (2014)
10. Milletari, F., et al.: Hough-CNN: deep learning for segmentation of deep brain regions in MRI and ultrasound. Comput. Vis. Image Underst. **164**, 92–102 (2017)
11. Nguyen, H.T.H., Wistuba, M., Schmidt-Thieme, L.: Personalized tag recommendation for images using deep transfer learning. In: Ceci, M., Hollmén, J., Todorovski, L., Vens, C., Džeroski, S. (eds.) ECML PKDD 2017. LNCS (LNAI), vol. 10535, pp. 705–720. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-71246-8_43
12. Pietikaäinen, M.: Computer Vision Using Local Binary Patterns. Springer, London (2011). https://doi.org/10.1007/978-0-85729-748-8
13. 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)
14. Redmon, J., Farhadi, A.: YOLO9000: Better. Stronger. arXiv preprint, Faster (2017)
15. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. **39**(6), 1137–1149 (2015)

16. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
17. Szegedy, C., et al.: Going deeper with convolutions. In: Computer Vision and Pattern Recognition (CVPR) (2015)