TasselNetV3: Explainable Plant Counting With Guided Upsampling and Background Suppression

Hao Lu, Liang Liu, Ya-Nan Li, Xiao-Ming Zhao, Xi-Qing Wang, and Zhi-Guo Cao, *Member, IEEE*

***Abstract*— Fast and accurate plant counting tools affect revolution in modern agriculture. Agricultural practitioners, however, expect the output of the tools to be not only accurate but also explainable. Such explainability often refers to the ability to infer which instance is counted. One intuitive way is to generate a bounding box for each instance. Nevertheless, compared with counting by detection, plant counts can be inferred more directly in the local count framework, while one thing reproaching this paradigm is its poor explainability of output visualization. In particular, we find that the poor explainability becomes a bottleneck limiting the counting performance. To address this, we explore the idea of guided upsampling and background suppression where a novel upsampling operator is proposed to allow count redistribution, and segmentation decoders with different fusion strategies are investigated to suppress background, respectively. By integrating them into our previous counting model TasselNetV2, we introduce TasselNetV3 series: TasselNetV3-Lite and TasselNetV3-Seg. We validate the TasselNetV3 series on three public plant counting data sets and a new unmanned aircraft vehicle (UAV)-based data set, covering maize tassels counting, wheat ears counting, and rice plants counting. Extensive results show that guided upsampling and background suppression not only improve counting performance but also enable explainable visualization. Aside from state-of-the-art performance, we have several interesting observations: 1) a limited-receptive-field counter in most cases outperforms a large-receptive-field one; 2) it is sufficient to generate empirical segmentation masks from dotted annotations; 3) middle fusion is a good choice to integrate foreground–background *a priori* knowledge; and 4) decoupling the learning of counting and segmentation matters.**

***Index Terms*— Explainable counting, local count network, maize tassel, phenotyping, plant counting, regression, rice plant, segmentation, unmanned aircraft vehicle (UAV), wheat ear.**

Manuscript received August 19, 2020; revised December 20, 2020; accepted February 9, 2021. Date of publication February 26, 2021; date of current version December 6, 2021. This work was supported by the Natural Science Foundation of China under Grant 61876211 and Grant 61906139. *(Corresponding author: Zhi-Guo Cao.)*

Hao Lu, Liang Liu, and Zhi-Guo Cao are with the Key Laboratory of Image Processing and Intelligent Control, Ministry of Education, Huazhong University of Science and Technology, Wuhan 430074, China, and also with the School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: hlu@hust.edu.cn; wings@hust.edu.cn; zgcao@hust.edu.cn).

Ya-Nan Li is with the School of Computer Science and Engineering, Wuhan Institute of Technology, Wuhan 430205, China (e-mail: yananli@wit.edu.cn).

Xiao-Ming Zhao and Xi-Qing Wang are with Center for Crop Functional Genomics and Molecular Breeding, College of Biological Science, China Agricultural University, Beijing 100193, China (e-mail: zhaoxiaoming@cau.edu.cn; wangxq21@cau.edu.cn).

Digital Object Identifier 10.1109/TGRS.2021.3058962

I. Introduction

PLANT counting has wide applications in agricultural breeding, plant phenotyping, crop monitoring, and yield forecasting [1]. Phenotyping geometrical traits, such as leaves/organs, allows breeders to identify desirable crop varieties [2]. The emergence rate of plants/ears is a key index in monitoring critical crop growth stages [3], [4]. Plant/ear density and fruit population are also closely related to yield [5], [6]. Acquiring all these indices requires plant counting.

A ruler and a scale are typical phenotyping tools in traditional agriculture such that many field technicians are required for manual measurement. This is particularly true for plant counting where this task is often implemented by random sampling and naked eyes. Manual counting, however, is labor-intensive, tedious, costly, time-consuming, and error-prone. It cannot meet the need of modern high-throughput phenotyping, which aims to monitor a substantial number of plots composed of versatile varieties over consecutive space and time [7]. Automating plant counting has become increasingly important in modern agriculture.

Due to the popularization of flexible plant phenotyping platforms, such as unmanned aircraft vehicle (UAV), yearly increased computing power, and the third wave of artificial intelligence—deep learning, there has been a growing interest in developing automated plant counting tools based on digital imagery. In the open literature, many of these tools are motivated by the success of deep convolutional neural networks (CNNs) [8]. While existing plant counting networks have reported remarkable performance [9]–[12], agricultural practitioners expect not only accurate but also explainable counting output. Here, explainability refers to the ability to infer which instance is counted. Since output explainability closely relates to output visualization, it drives how practitioners choose a certain algorithmic framework.

One intuitive visualization is to draw a bounding box onto an instance or to delineate the contour. This may explain why plant counting is often formulated as an object detection [5], [13] or an image segmentation [14], [15] problem. Another reason why the detection pipeline is followed can boil down to its potential to achieve panoptic segmentation [16] to have a full description of the scene. Albeit intuitive, their effectiveness comes at the cost of the expensive bounding box and pixel-level annotations. To robustize a plant detector that works in reality, manual annotation

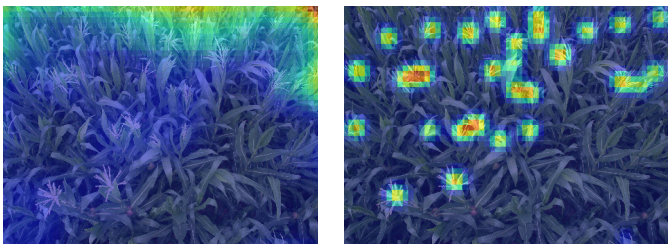

Fig. 1. Output visualization of TasselNetV2 and TasselNetV3 in MTC. Compared with TasselNetV2, one can see more clearly from the visualization of TasselNetV3 which instance is counted.

often has to repeat to supplement more training data to cover different cultivars and different environments. This distracts scientists from real scientific problems that should be focused on. To popularize the use of a plant counting algorithm, it is important to reduce the human involvement in the loop [6].

We remark that, counting instances does not have to detect or segment instances first. The transductive principle suggests that, *when solving a problem of interest, one should not solve a more general (harder) problem as an intermediate step but pursue the answer that really needs* [17]. In computer vision, object detection and image segmentation are considered harder than object counting. Detection or segmentation should not be a prerequisite to counting. Instead, counting can be addressed more directly in the framework of object counting with cheap dotted annotations.

Object counting in computer vision has been considered an independent research area since 2008 when counting by regression is introduced [18]. In this paradigm, three subbranches exist. The first is to regress the global image count [18]. It amounts to optimize the image-level mean absolute error (MAE). However, this regression target is later found difficult to optimize because of large variations in images and the open-set nature of count values [11]. A good surrogate (the second branch) is to regress the density map introduced by Lempitsky and Zisserman [19]. The density map is a pointwise regression target generated from dotted annotations with Gaussian smoothing. It inspires a majority of deep counting networks, such as MCNN [20] and CSRNet [21]. The third branch is a blockwise regression target, local counts, first appeared in [22]. This idea is recently deeplized by Lu *et al.* [9] and Cohen *et al.* [23]. In our previous work [9], we have shown that this local count framework is particularly suitable for plant counting because of its robustness to local size variations. Such robustness is useful because plants are self-changing by nature. Particularly, it has been shown that local counts provide a tighter surrogate of image-level MAE than the density map [24].

Albeit effective, one thing reproaching the local count framework is its poor explainability of output visualization. From Fig. 1, the visualization of a typical local count model TasselNetV2 [10] only produces coarse responses, which is hard to infer which instance is counted. The poor explainability also limits the counting performance: supervision signals can only be injected after the low-resolution feature map; failure patterns are difficult to diagnose. Inspired by guided upsampling [25], we propose a novel upsampling operator to enable count redistribution and additional supervision to improve output visualization. Improved visualization further contributes to the performance increase, allowing diagnosis of failure cases, such as false responses appearing in visually similar backgrounds (see Fig. 6). We, thus, investigate several segmentation decoders with different fusion strategies for background suppression. By incorporating the upsampling operator and the decoder in a unified framework, we extend our previous local count model TasselNetV2 and propose TasselNetV3 series: 1) TasselNetV3-Lite, a lightweight model that inherits the vein of TasselNetV2 but with an enhanced feature encoder; 2) TasselNetV3-Seg†, a single-decoder model with a segmentation branch and a limited-receptive-field counter; and 3) TasselNetV3-Seg‡, a double-decoder version that enlarges the receptive field of the counter.

To demonstrate the generality of TasselNetV3, we validate TasselNetV3 on three public plant counting data sets, i.e., the maize tassels counting (MTC) data set [9], the wheat ears detection (WED) data set [5], and the rice plants counting (RPC) data set [12], as well as a new UAV-based maize tassels data set termed MTC-UAV, with over 400 maize cultivars and 70 870 manually labeled instances. Images in the four data sets are captured at different imaging platforms, under varying in-field environments, from distinct imaging views and with changing imaging heights. Results demonstrate that guided upsampling and background suppression lead to improved counting performance, TasselNetV3 series consistently outperforms TasselNetV2 by large margins, and TasselNetV3 series also outperforms or at least are on par with other recent deep counting networks. More importantly, compared with TasselNetV2, the visualization of TasselNetV3 is more explainable, as shown in Fig. 1. Aside from reporting state-of-the-art results, we have also several interesting observations: 1) plant counting is a local visual problem: a limited-receptive-field counter in most cases delivers counting performance better than a large-receptive-field counter; 2) by knowing empirically the size of the plants, segmentation can benefit counting performance; 3) a fusion strategy matters: middle fusion outperforms early/late fusion when integrating foreground–background prior knowledge; and 4) an appropriate supervision is essential: it is crucial to decouple the learning of counting and segmentation.

Our main contributions include the following.

1) A novel upsampling operator designed for local count networks, which enables explainable visualization and additional supervision.
2) An empirical study on how to integrate foreground–background *a priori* knowledge given dotted annotations.
3) *TasselNetV3 Series:* Three plant counting models with state-of-the-art performance on four plant counting benchmarks.
4) *MTC-UAV:* A new UAV-based plant counting data set, with over 400 maize cultivars and 70 870 annotated instances.[1]

[1]The UAV-based plant counting data set is made available at https://git.io/mtc-uav

II. Plant Counting: A Recap

In early vision applications in agriculture, plant counting is mainly addressed by color-based segmentation [26] because plants have strong color cues. The color cues, however, are considered unstable when the illumination changes. Some attempts integrate appearance and texture features [27]. Some approach the problem with a detection idea [28]. Nevertheless, counting performance is not ready for practical applications due to weak feature representation. This issue is largely alleviated when deep CNNs emerge.

In deep learning era, many methods still prefer to build a segmentation [29] or a detection [5], [30], [31] pipeline. One possible reason is that both detection and segmentation have well-known, robust, and easy-to-use frameworks, such as Faster R-CNN [32] and fully convolutional networks (FCNs) [33]; another plausible explanation is that practitioners aim to not only count the number of plants but also estimate their sizes. However, we remark that size estimation is not reliable in 2-D imagery due to camera distortions and pose variations. This is a problem that should be addressed in the 3-D domain [34].

Compared with counting by detection and segmentation, regression-based plant counting is less studied. Giuffrida *et al.* [2] proposed to count leaves based on log-polar representation and support vector regression. Rahnemoonfar and Sheppard [35] presented a ResNet-like [36] density regression model to count tomatoes based on simulated data. Aich and Stavness [37] introduced an encoder–decoder network to regress the leaf count. While promising results are reported, experiments are limited on potted plants or under a controlled environment. Recently, Wu *et al.* [38] propose a combined network that integrates density map regression and image segmentation for rice seedlings counting. The segmentation map and the density map are fused at the decision level, while we show that middle fusion is more effective than late fusion in plant counting. Osco *et al.* [39] regressed an unnormalized version of the density map and predicts dots with nonmaximum suppression for citrus-trees counting and locating. Dotted visualization is also a good idea to promote explainability, but, to the best of our knowledge, dot prediction generally requires a heuristic parameter to define a so-called peak point. It may not generalize well to varying densities.

In our previous work [9], we propose TasselNet, the first local count network applied to plants, to count in-field maize tassels. Later, we extend TasselNet to TasselNetV2 [10] to count wheat ears. TasselNetV2 is motivated by an observation that weak context matters for nonrigid plants, such as maize tassels and wheat ears. However, both TasselNet and TasselNetV2 do not generate human-interpretable output visualization. We aim to address this problem in TasselNetV3.

III. Image Acquisition and Data Sets

A. Experimental Site and Image Acquisition

We collect maize tassel images captured by a UAV. The experimental site is located at China Agriculture University, Beijing (China, 30.48° latitude North, 114.35° longitude East). The experimental field covers around 1 ha. Over 400 maize cultivars are sown in the 2019 spring growing season. Different cultivars are sown randomly in different microplots, and each cultivar has six repetitions. Each microplot is of 5-m length and 0.6-m width. For the UAV, we use the DJI Mavic 2 Pro. The flying height is 12.5 m above the ground. The camera has a focal length of 28 mm such that the ground sampling resolution is about 0.3 cm/pixel. The flying speed is fixed to 1.1 m/s. Images are captured with 80% overlaps along the flying direction between every two consecutive shots.

B. Plant Counting Data Sets

Here, we introduce the new MTC-UAV data set and the other three plant counting data sets. We also summarize data set characteristics and key challenges in plant counting.

The MTC-UAV data set addresses MTC from aerial imagery. To avoid overlaps between images, we choose one image from every five consecutive shots. Three hundred and six nonoverlapping images of resolution 5472×3648 are consequently chosen to compose the MTC-UAV data set. Among them, 200 images are randomly sampled to be the training set, and the testing set takes the rest. The number of tassels in each image varies from 36 to 550. Following the standard practice [19], each tassel is manually annotated with a dot. Instances of 70 870 in all are finally annotated. We remark that, compared with expensive bounding-box and pixel-level annotation, dotted annotations are affordable. Indeed, dotting is the most natural way to count instances.

The MTC data set [9] is also about counting maize tassels. It includes 361 images collected from four experimental fields across China between 2010 and 2015 with six maize cultivars. Images are captured from 4/5-m-height phenopoles with a charge-coupled device (CCD) digital camera (E450 Olympus). Details about the imaging device can refer to [40]. The image resolutions are 3648×2736, 4272×2848, and 3456×2304. Among 361 images, 186 images are used for training, and the other 175 images are used for testing. The number of maize tassels in each image varies from 0 to around 100. This data set is also annotated with dots.

The WED data set [5] targets wheat ears counting. It is collected in 2017 in France with a movable phenotyping platform using a Sony ILCE-6000 digital camera. Experimental fields consist of a trial of 120 2×10 m^2 microplots with 20 different genotypes. The imaging distance is 2.9 m above the ground. The image resolution is 6000×4000. The number of ears in each image varies from 80 to 170. The data set includes 236 images, where 165 and 71 images are used for training and testing, respectively. Bounding box annotations are provided in this data set, while we only use the center point of each box in experiments.

The RPC data set [12] concerns RPC at early growth stages. It consists of 382 high-resolution images of size 4272×2848. Images are captured between 2010 and 2013 in China with the same device and imaging conditions akin to the MTC data set. The number of rice plants in each image ranges from 182 to 1330. Two hundred and thirty images are used for training, and the other 152 images are employed for testing. Dotted annotations are provided.

Data Set Characteristics and Open Challenges: Some characteristics of the four data sets are summarized in Table I.

TABLE I
CHARACTERISTICS OF DIFFERENT PLANT COUNTING DATA SETS

Dataset	Imaging platform	Imaging view	Imaging height	# images	# labeled instances
MTC–UAV	UAV	nadir	12.5 m	306	70,870
MTC	phenopole	oblique	4 m / 5 m	361	13,564
WED	movable boom	nadir	2.9 m	236	30,729
RPC	phenopole	oblique	5 m	382	211,971

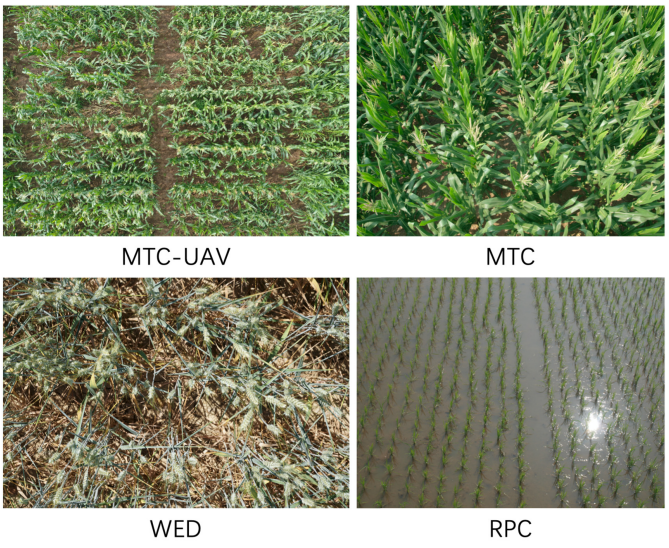

Fig. 2. Example images in the four plant counting data sets.

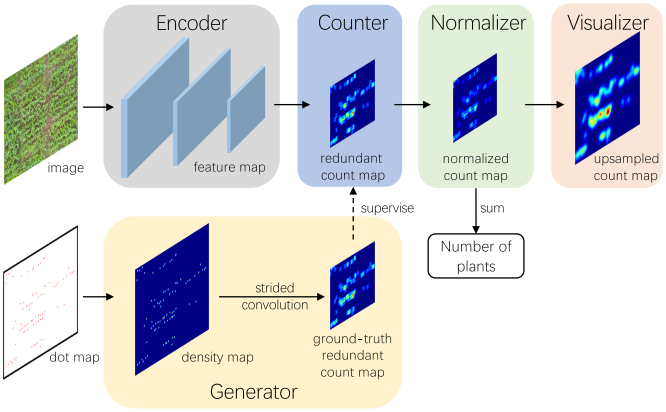

Fig. 3. Local count framework. A naive local count model includes an encoder used to encode image features, a counter used to map the features to local counts, a normalizer used to normalize redundant local counts, a generator used to generate supervision signals, and a visualizer for output visualization.

Example images are shown in Fig. 2. The four data sets cover different plant varieties, and images are captured at different imaging platforms, under varying in-field environments, from distinct imaging views, and with changing imaging heights. We believe that evaluations on these data sets can, to a great extent, reveal the generality of a plant counting tool in reality.

Plant counting is challenging, especially in a field-based environment. Some plant counting-related workshops are organized to showcase the challenges, e.g., a wheat head detection challenge is recently organized in conjunction with ECCV 2020 [13]. Some challenges are general, including: 1) extrinsic variations engendered by the environment, such as changing illumination, pose variances, image degradation, occlusions, and cluttered background and 2) intrinsic variations due to plants per se—size, color, and texture—will change during plant growth. Some challenges, however, are exclusive. For instance, rice grows in a field covered by water such that strong sunlight reflection may occur; plant sizes change significantly in images when capturing from an oblique view, even if they have similar physical sizes in reality; and wine grapes are perennial plants, so counting must be on-site and from a side view with significant occlusions.

IV. FROM TASSELNETV2 TO TASSELNETV3

A. Local Count Framework: A Generalized View

Here, we first provide a generalized view of the local count framework and revisit the TasselNetV2 baseline.

Local count modeling aims to learn a mapping from an image patch of size $p \times p$ to a local count value. As shown in Fig. 3, basic components of a local count model include an encoder used to encode image features, a counter used to map the features to local counts, a normalizer used to normalize redundant local counts, a generator used to generate ground truths, and a visualizer used to visualize the output.

1) Encoder: Given an image I of size $h \times w \times 3$, the encoder defines a transformation $\mathcal{E}_I : \mathbb{R}^{h \times w \times 3} \longrightarrow \mathbb{R}^{(h/s_e) \times (w/s_e) \times m}$ to encode I and generate the feature map X, where h and w are image height and image width, respectively, and m is the number of feature channels. s_e is the downsampling rate of the encoder, e.g., if the encoder has three downsampling stages with a downsampling rate of 2 for each stage, $s_e = 2^3 = 8$.

2) Counter: Given X, the counter further defines a transformation $\mathcal{C}_X : \mathbb{R}^{(h/s_e) \times (w/s_e) \times m} \longrightarrow \mathbb{R}^{(h/o) \times (w/o)}$, where $o = s_e \times s_c$ is also known to be the output stride, and s_c is the downsampling rate of the counter (s_c is often set to 1 in literatures). It generates a (redundant) count map C where each element represents a local count of a local region.

3) Normalizier: The key idea of the normalizer is to compute how many times each position is counted and normalize the redundant local counts accordingly. Note that, it is only required when $o < p$. $o < p$ suggests that overlaps exist between adjacent image patches. When $o = p$, the normalizer is unnecessary. Given C, the normalizer defines a transformation $\mathcal{N}_C : \mathbb{R}^{(h/o) \times (w/o)} \longrightarrow \mathbb{R}^{(h/o) \times (w/o)}$. It produces a normalized count map Z without changing the spatial resolution. The sum of Z equals to the global image count. Further details with respect to the normalizer can be found in [9] and [10].

4) Generator: The generator generates a ground-truth (redundant) count map $\hat{C}$ to supervise the learning of the encoder and the counter. $\hat{C}$ is generated from a density map D [9], and D is generated from a dot map $T \in \mathbb{R}^{h\times w}$ with Gaussian smoothing, which takes the form

$$D = T * G_\sigma \tag{1}$$

where G_σ is a Gaussian kernel parameterized by σ. If $\mathbb{1}_p$ is denoted by a $p \times p$ full-one matrix, $\hat{C}$ is defined by

$$\hat{C} = D *_o \mathbb{1}_p \tag{2}$$

where $*_o$ is an o-stride convolution operator. $\hat{C}$ and C are of the same size. This process defines a transformation $\mathcal{G}_T : \mathbb{R}^{h\times w} \longrightarrow \mathbb{R}^{(h/o)\times(w/o)}$. While σ is adjustable, it has been shown that $\hat{C}$ is not sensitive to σ [41].

5) Visualizer: The visualizer projects the normalized count map Z back to a $h \times w$ high-resolution count map, characterized by a transformation $\mathcal{V}_Z : \mathbb{R}^{(h/o)\times(w/o)} \longrightarrow \mathbb{R}^{h\times w}$. This is a typical upsampling process. Without any auxiliary information, upsampling is blind. This is why nearest-neighbor-like interpolation is used in TasselNet such that a local count is averaged into a local region, resulting in ambitious visualization and poor explainability (see Fig. 1). However, it is possible to perform guided upsampling by introducing auxiliary information [25], [42]. A key contribution of this work is how we generate such auxiliary information.

6) TasselNetV2 Revisited: TasselNetV2 is developed in this local count framework where the transformations above are modeled by a CNN. The encoder is implemented by a plain network, denoted by $\mathbf{L}_3^1(16)$-$\mathbf{M}_2^2$-$\mathbf{L}_3^1(32)$-$\mathbf{M}_2^2$-$\mathbf{L}_3^1(64)$-$\mathbf{L}_3^1(64)$-$\mathbf{L}_3^1(64)$-$\mathbf{M}_2^2$, where $\mathbf{L}_k^s(m)$ is a convolutional layer parameterized by a m-channel s-stride $k \times k$ kernel with zero paddings, followed by batch normalization (BN) and rectified linear unit (ReLU), and $\mathbf{M}_k^s$ is a max pooling layer with a s-stride $k \times k$ kernel. The counter is defined by $\mathbf{L}_8^1(128)$-$\mathbf{L}_1^1(128)$-$\mathbf{L}_{\mathbf{n}}{}_1^1(1)$ where $\mathbf{L_n}$ is a convolution layer without BN and ReLU. According to this definition, each local count is mapped from a 64×64 image patch. Since the output stride $o = 8 < p = 64$, the resulting count map C is redundant. A normalizer is, thus, required. We refer readers to implementation details in [10]. Training TasselNetV2 requires minimizing the ℓ_1 loss between C and $\hat{C}$, i.e., $\min(1/n)\sum_{i=1}^n |C_i - \hat{C}_i|$, where n is the number of training images. To visualize the count map, normalized local counts are further upsampled.

B. Dynamic Unfolding for Count Redistribution

We believe that it is the averaging effect in the visualizer that leads to poor explainability. Inspired by guided upsampling [25], we propose a novel upsampling operator termed *dynamic unfolding* that improves explainability with count redistribution.

Dynamic unfolding is inspired by the *spatial divisibility* of counting—a large-region count equals the sum of spatially divided subregion counts [11]. As shown in Fig. 4, the idea of dynamic unfolding is simple. Instead of blindly averaging a count c into subregions, we unfold c based on subregion

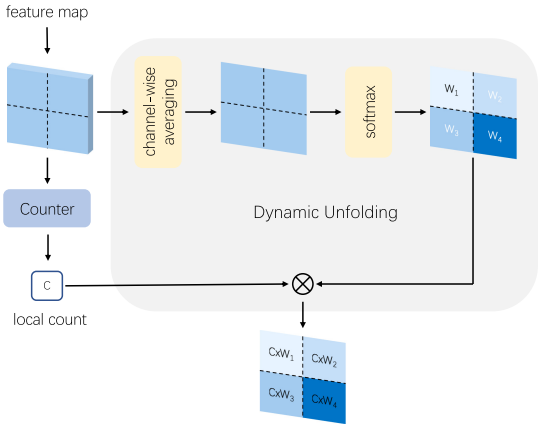

Fig. 4. Dynamic unfolding for local count redistribution.

weights to enable count redistribution, conditioned on the encoder feature map. The weights are normalized such that a weighted sum of subregion counts still equals c. This redistribution process is dynamic in which the redistribution weight is data-dependent.

For ease of exposition, we first take a look at an image patch of size $p \times p \times 3$. Hence, a local count c can be generated from the encoder feature map $X \in \mathbb{R}^{(p/s_e)\times(p/s_e)\times m}$. Given X, we further generate a weight map $W \in \mathbb{R}^{(p/s_e)\times(p/s_e)}$. In our implementation, W is generated by applying nonparametric channelwise averaging to X.[2] The weight map is further spatially normalized by a `softmax` function such that, for each $w_i \in W$, $\sum_i w_i = 1$. By multiplying the weight map with the local count c, a redistributed count map R can be generated. Compared with c that is mapped from a $p \times p$ region, each $w_i \times c$ indicates a subregion count aggregated from a $s_e \times s_e$ region. Since s_e is often smaller than p, after count redistribution, the $(1/p)$-resolution count map is transformed to the $(1/s_e)$-resolution output. This is why the resolution is risen and the explainability of visualization is enhanced.

Notice that, when processing an image of arbitrary size, say $h \times w \times 3$, the redistributed count map R is not an upsampled version of C. Instead, R has an output size of $(h/s_e) \times (w/s_e)$. If redundancy exists ($o < p$), a normalizer is also required for de-redundancy.

Aside from enhanced explainability, dynamic unfolding further enables explicit supervision to R. That is, an additional loss can be defined between R and its ground truth $\hat{R}$. Since R is closely related to W, a good prediction of W suggests good spatial responses of the encoder feature map. In what follows, we discuss how to incorporate dynamic unfolding into the TasselNetV3 series.

C. TasselNetV3-Lite

Here, we first introduce TasselNetV3-Lite, a lightweight plant counting model that inherits the vein of TasselNetV2.

[2]We also try the idea of predicting the weight map with a parametric convolutional layer or with channelwise max pooling but do not observe any improvement.

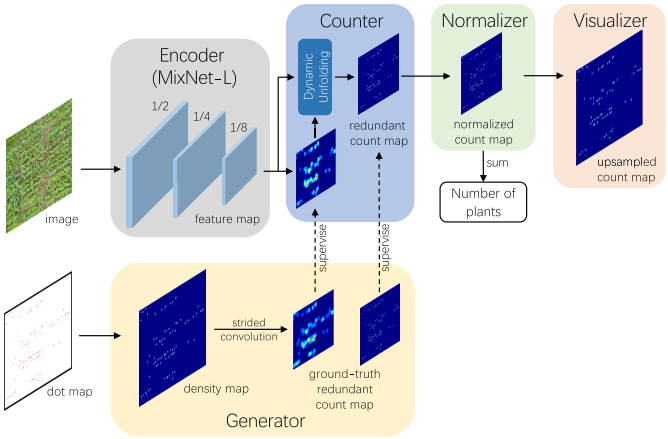

Fig. 5. Overview of TasselNetV3-Lite. Compared with TasselNetV2, TasselNetV3-Lite incorporates a novel dynamic unfolding module for count redistribution and enhances the encoder with mixed convolution and a pretrained model.

Lightweight models are preferred when real-time processing or high-throughput requirements are expected.

The overview of V3-Lite is shown in Fig. 5. Its pipeline is similar to V2. Since both V2 and V3 are fully convolutional models, they can accept all image sizes. Compared with V2, we make two modifications: 1) replacing plain convolution with an ImageNet-pretrained encoder MixNet-L [43] and 2) enhancing the naive counter with dynamic unfolding. We mainly explain the two modifications: the generator and our modified loss function.

1) Encoder: The use of MixNet-L is inspired by our preliminary evaluations showing that this backbone is superior to other high-capacity models in RPC.[3] We think that its superiority has two reasons: 1) MixNet-L is implemented with mixed convolution, which shares the same spirit with a successful multicolumn idea in crowd counting [20] and 2) the model is pretrained on ImageNet, which has been proven to be able to significantly improve counting [21]. We will present a step-to-step analysis in Section V-D to justify their benefits. In particular, MixNet-L has five encoding stages. Each encoding stage has a downsampling layer with a downsampling rate of 2. Following V2, we make use of the first three encoding stages in V3-Lite such that $s_e = 8$. Since MixNet-L is implemented with depthwise convolution, the number of parameters of V3-Lite is even fewer than V2 (0.3 M versus 0.5 M).

2) Counter: The benefit of dynamic unfolding and how it works have already been made clear in Section IV-B. With this new module, the counter not only outputs local counts from $p \times p$ regions but also generates local counts from $s_e \times s_e$ regions. The fine-resolution count map is what we use in the normalizer and the visualizer.

3) Generator: To supervise both C and R, the generator now generates two redundant ground-truth count maps: a coarse-resolution map $\hat{C}$ used to supervise C and a fine-resolution map $\hat{R}$ used to supervise R. $\hat{C}$ is generated per (2). The generation of $\hat{R}$ is somewhat tricky. The most intuitive way to explain this is to use the code implementation

[3]Our evaluations on backbone choices are at https://git.io/sfc2net

Algorithm 1 PyTorch Implementation of Generating a Fine-Resolution Ground-Truth Count Map

```
def R_generator(D, p, o, Io):
  # Input:
  #   D: density map
  #   p: patch size
  #   o: output stride
  #   Io: full-one matrix of size oxo
  # Output:
  #   R_hat: fine-resolution ground-truth
           map
   h, w = D.size()[2:]
   rh, rw, k = int(h/o), int(w/o),
       int(p/o)
   R_hat = F.conv2d(D, Io, stride=o)
   R_hat = F.unfold(R_hat, kernel_size=k)
   R_hat = F.fold(R_hat, (rh, rw),
     kernel_size=k)
   return R_hat
```

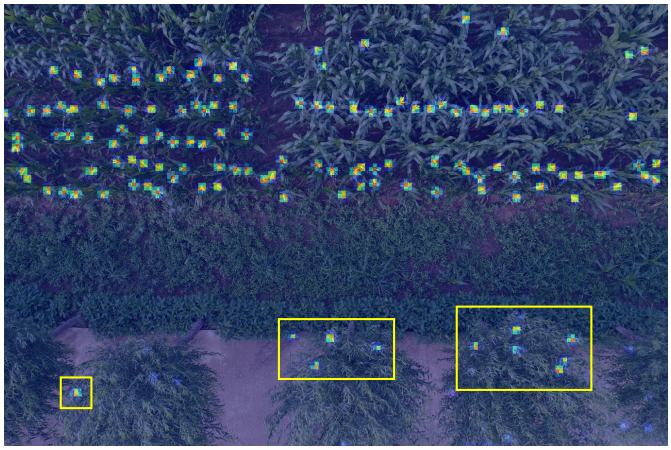

Fig. 6. Motivation behind TasselNetV3-Seg. In TasselNetV3-Lite, some false positives (yellow boxes) appear. These are visually similar background regions. Our intuition tells us that they may be suppressed if sufficient contextual information is taken into account. We, therefore, augment TasselNetV3 with a segmentation branch where context is encoded.

shown in Algorithm 1 where unfolding and folding are applied to a nonredundant fine-resolution count map. Examples of generated coarse-resolution and fine-resolution count maps are identical to Fig. 1.

4) Loss Function: Given $\hat{C}$ and $\hat{R}$, two loss functions can be defined. Following [9] and [10], ℓ_1 loss is chosen. We, therefore, define the coarse-resolution counting loss L_C and the fine-resolution counting loss L_R to be $L_C = (1/n)\sum_{i=1}^{n} |C_i - \hat{C}_i|$ and $L_R = (1/n)\sum_{i=1}^{n} |R_i - \hat{R}_i|$, respectively, where i denotes the ith training sample, and n is the number of training samples. In our experiments, we find that a direct sum of the two losses works well. Hence, the overall loss $L_{v3\text{lite}}$ takes the form

$$L_{v3\text{lite}} = L_C + L_R. \tag{3}$$

D. TasselNetV3-Seg

TasselNetV3-Lite is lightweight, but it has a limited receptive field such that it may not discriminate similar

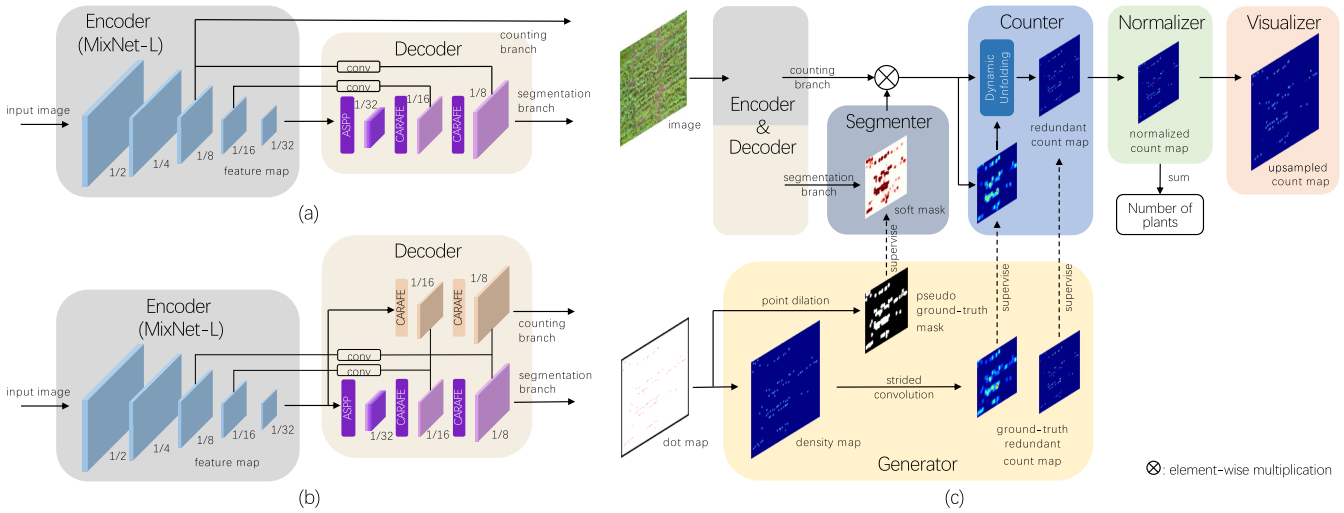

Fig. 7. Overview of TasselNetV3-Seg. V3-Seg augments V3-Lite with a decoder and a segmenter. We implement two variants of V3-Seg: (a) single-decoder V3-Seg† and (b) double-decoder V3-Seg‡. According to (c) full pipeline (the encoder and decoder are modularized), V3-Seg† and V3-Seg‡ only differ in the decoder design.

background regions shown in Fig. 6. Our experience in dense prediction tasks [25] tells us that context plays an important role in background suppression. We, thus, propose TasselNetV3-Seg where a segmentation idea is investigated. We consider segmentation because context encoding, such as atrous spatial pyramid pooling (ASPP) [44], is extensively studied in segmentation problems. We follow an encoder–decoder architecture similar to feature pyramid network [45]. The local count framework is, therefore, augmented with a decoder and a segmenter shown in Fig. 7. In this new framework, we further investigate two variants: TasselNetV3-Seg† and TasselNetV3-Seg‡. Their difference lies in the decoder design. V3-Seg† is with a limited-receptive-field counter, while V3-Seg‡ has a large-receptive-field counter. Since the counter, the normalizer, and the visualizer of TasselNetV3-Seg models remain the same compared to TasselNetV3-Lite, we focus on those modified components, which will be discussed next.

1) Encoder and Decoder: The overview of TasselNetV3-Seg is shown in Fig. 7. V3-Seg† and V3-Seg‡ only differ in the encoding–decoding stage. To acquire a sufficiently large receptive field, we now use all five encoding stages of MixNet-L in the encoder. The decoder has a counting branch and a segmentation branch. The counting branch uses the feature map of $1/8$ input resolution. This feature map comes from the encoder in V3-Seg† but from the decoder in V3-Seg‡. Since the receptive field of the decoder feature map is significantly larger than that of the encoder feature map, the information utilized by the counter is different. The segmentation branch also employs the feature map of $1/8$ input resolution. To acquire this, the $1/32$ encoder feature map first goes through a context encoding module ASPP [44] and then is upsampled by a data-dependent upsampling operator CARAFE [46] twice. The encoder feature map of the same resolution (the channel dimension is adjusted by 1×1 convolution) is summed to the corresponding decoder feature map after upsampling to supplement details. ASPP and CARAFE are both parametric operators, so this implementation can be viewed as parametric decoding. In Section V-D, we also investigate a nonparametric decoding idea where context encoding and upsampling are implemented by pyramid pooling module (PPM) [47] and bilinear interpolation, respectively. Our investigations show that parametric decoding outperforms nonparametric decoding.

2) Segmenter: Given the feature map generated from the decoder, the segmenter outputs a segmentation map. The segmentation map provides *a priori* knowledge of foreground and background. A key design choice in TasselNetV3-Seg is about how to use the segmentation map to modulate counting. Fig. 7(c) implements a middle-fusion strategy where a soft segmentation map is used to modulate the counting feature map before feeding it into the counter. This is the most effective implementation that we find after comparing three fusion strategies (details can be found in Section V-D). Fig. 8 shows some soft segmentation maps from different data sets. They provide a cue on possible plant locations. By fusing segmentation maps, background features can be explicitly suppressed so that the counter can focus on foreground instances.

3) Generator: Apart from generating two ground-truth redundant count maps to supervise the counter, the generator of TasselNetV3-Seg also generates a pseudo-ground-truth mask to supervise the segmenter. Since only dotted annotations are given, there is no real-sense ground-truth mask. We can only generate a pseudomask by point dilation. In this article, we apply a distance transform to generate this mask. Specifically, the Euclidean distance from each pixel to the closest pixel is computed by $d(P_{x,y}, F_{x,y}) = \|P_{x,y} - F_{x,y}\|_2^2$, where $P_{x,y}$ is a pixel indexed by coordinates x and y, and $F_{x,y}$ is the annotated point with the closest distance to $P_{x,y}$. To obtain the pseudomask, a threshold τ can be set such that

$$\hat{S}_{x,y} = \begin{cases} 1, & \text{if } d(P_{x,y}, F_{x,y}) < \tau \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

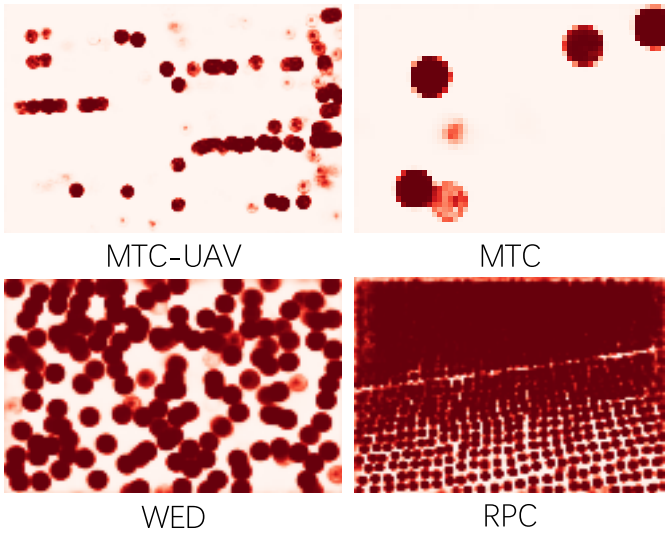

Fig. 8. Soft segmentation masks of different plant counting data sets. The darker the color is, the higher probability a region being a plant. Segmentation maps provide a cue where plants locate.

where $\hat{S}$ is the pseudo-ground-truth mask. $\hat{S}_{x,y} = 1$ indicates the foreground, and $\hat{S}_{x,y} = 0$ is the background. Point dilation can be viewed as drawing a circle around each point where its radius is controlled by τ. Since plant sizes can change, the foreground can be either the whole plant or a relevant part of it indicating a single plant is present. We remark that, an empirical estimate of the mask is sufficient as long as segmentation does not affect counting. In our experiments, we set $\tau = (p/2)$ so that the prediction of the counter is not affected by the segmenter.

4) Loss Function: We find that, to make this idea work, it is crucial to decouple the learning of counting and segmentation. We achieve this by designing a guided counting loss. This loss is inspired by an observation that, there are always missing annotations in a data set, probably due to fatigue of humans after long-time annotation. When learning the counter, these true foreground instances will be treated as background regions, leading to ambiguities in learning. Segmentation provides an opportunity to alleviate these ambiguities. The key insight is to *transform the uncertainty from the counter to the segmenter*. That is, the counter is only learned from annotated points, and ambitious backgrounds are cast to the segmenter. The segmenter is only required to correctly discriminate background regions where counts are zero. This is not hard because the probability for the segmenter to falsely label a background to be a foreground is much less than to falsely label a foreground to be a background. In particular, the guided counting loss L_{guide} is defined by $L_{\text{guide}} = L_{\text{GC}} + L_{\text{GR}}$, where

$$L_{\text{GC}} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[\hat{C}_i > 0]|C_i - \hat{C}_i| \tag{5}$$

and

$$L_{\text{GR}} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[\hat{R}_i > 0]|R_i - \hat{R}_i|. \tag{6}$$

$\mathbb{1}[\cdot]$ is an indicator function. The segmenter is supervised by a standard binary cross-entropy loss L_{bce}. The final loss $L_{v3\text{seg}}$ is the sum of the guided counting loss and the segmentation loss

$$L_{v3\text{seg}} = L_{\text{guide}} + L_{\text{bce}}. \tag{7}$$

E. Implementation Details

Here, we elaborate on implementation details. Our implementations are based on `PyTorch`. Local counts are regressed from 64×64 ($p = 64$) local regions with an output stride of 8 ($o = s_e = 8$). The *encoder* follows the official implementation of MixNet-L [43]. Its (1/8), (1/16), and (1/32) outputs have channel dimensions of 56, 160, and 264, respectively. Following the naming rule in Section IV-A, the *counter* of V3 series is defined by $\mathbf{A}_8^1$-$\mathbf{L}_1^1(m)$-$\mathbf{L}_1^1(1)$ (dynamic unfolding is not shown), where $\mathbf{A}_k^s$ is an average pooling layer with a s-stride $k \times k$ kernel. $m = 56$ in V3-Lite and V3-Seg†, and $m = 64$ in V3-Seg‡. Compared with the counter of TasselNetV2, the first convolutional layer in the V3 counter is replaced with average pooling. We do not observe any performance loss with this modification. The segmentation branch of the *decoder* is defined by $\mathbf{ASPP}_{264}^{64}$-$\mathbf{CARAFE}_{32}$-$\mathbf{CARAFE}_{32}$-$\mathbf{L}_3^1(64)$, where $\mathbf{ASPP}_{\text{inp}}^{ou}$ denotes an ASPP module with inp-dimensional input and oup-dimensional output, and $\mathbf{CARAFE}_q$ denotes a CARAFE operator processing a q-dimensional (squeezed) feature map. Note that CARAFE does not change the channel dimension (64) of input and output. The last $\mathbf{L}_3^1(64)$ is used to smooth the upsampled features. The counting branch in V3-Seg‡ is simplified to $\mathbf{CARAFE}_{32}$-$\mathbf{CARAFE}_{32}$-$\mathbf{L}_3^1(64)$. Two side convolutions take the form $\mathbf{L}_{\mathbf{n}1}^1(64)$. The *segmenter* is denoted by $\mathbf{L}_{\mathbf{n}1}^1(1)$-**Sigmoid**. The `sigmoid` function is used to normalize the segmentation map. The *normalizer* is implemented identical to TasselNet and TasselNetV2. Readers can refer to [9] and [10] for further details. The *visualizer* upsamples the normalized count map for final visualization. It applies a direct averaging strategy, i.e., for each normalized local count c, c is averaged into each 8×8 region such that each element in this region equals to $(c/64)$.

V. Results and Discussion

We first introduce evaluation metrics and training details. We then report performance and compare the TasselNetV3 series against state-of-the-art approaches. We also present ablation studies to justify key design choices. Finally, we compare the efficiency and extend our insights toward good practices.

A. Evaluation Metrics

We use the following evaluation metrics to quantify the counting performance: MAE, relative MAE (rMAE), root mean square error (RMSE), relative RMSE (rRMSE), and the coefficient of determination (R^2). Formally, MAE $= (1/n)\sum_{i=1}^{n}|P_i - G_i|$, RMSE $= ((1/n)\sum_{i=1}^{n}(P_i - G_i)^2)^{1/2}$, rMAE $= (1/n)\sum_{i=1}^{n}|(P_i - G_i/G_i)|$, rRMSE $=$

TABLE II
SETTINGS OF THE DOWNSAMPLING RATE r AND THE GAUSSIAN KERNEL σ OF DIFFERENT DATA SETS

Dataset	r	σ
MTC–UAV	1/4	$1/4 \times 20$
MTC	1/8	$1/8 \times 80$
WED	1/6	$1/6 \times 40$
RPC	1/4	$1/4 \times 16$

$((1/n)\sum_{i=1}^{n}((P_i - G_i/G_i))^2)^{1/2}$, and $R^2 = 1 - (\sum_{i=1}^{n}(P_i - G_i)^2)/(\sum_{i=1}^{n}(P_i - \overline{G})^2)$, where n is the number of images, P_i and G_i are predicted image count and ground-truth image count of the ith image, respectively, and $\overline{G}$ is the mean ground-truth count. G_i equals to the number of annotated points in the ith image.

B. Training Details

For a fair comparison, the V3 series and other competitors are trained and tested with the same configuration. In training, we use only 320×320 random cropping and random flipping for data augmentation. Encoder parameters are pretrained on ImageNet if applicable. BN layers of the encoder are not fixed. The stochastic gradient descent optimizer is used where the momentum and the weight decay are set to 0.95 and 5×10^{-4}, respectively. We optimize parameters for 500 epochs with a batch size of 16. The learning rate is initialized to 0.01 and reduced by $10\times$ at the 200th and 400th epochs, respectively. During inference, to process an image of arbitrary size, image borders are padded with zeros to be divisible by the output stride o. To reduce experimental costs, high-resolution images are downsampled. We examine the downsampling rate r from the range of values $r = \{(1/16), (1/8), (1/6), (1/4), (1/2), 1\}$, and r is chosen to be the minimum downsampling rate when no significant performance loss is observed. Following [9], the Gaussian kernel σ used to generate the density map for each data set is set empirically based on visual inspection on the training set such that the Gaussian responses approximately cover instances of interest. Detailed settings of the downsampling rates and kernel parameters for each data set are listed in Table II. These hyperparameters are fixed once chosen.

C. Comparison With State of the Art

We compare the V3 series with five state-of-the-art approaches, including MCNN [20], CSRNet [21], BCNet [41], SFC^{2}Net [12], and our direct baseline TasselNetV2 [10]. Other published results are also cited if applicable. We also report the number of model parameters as an indicator of computational complexity and specify whether an ImageNet-pretrained model is used. Quantitative results on the four plant counting data sets are listed in Tables III–VI. Qualitative results are shown in Figs. 9–12. We have the following observations.

1) V3-Lite consistently outperforms V2, even with fewer model parameters.
2) V3-Seg further exhibits at least 18.4% relative improvements with respect to MAE over V3-Lite. Notice that such improvements are achieved over a strong baseline.
3) V3-Seg sets the new state of the art with clear advantages over other competitors on the four plant counting data sets.
4) V3-Seg† works better than V3-Seg‡ on three out of four plant counting data sets, which suggests that it is sufficient to learn a focused counter with a limited receptive field. This observation, in a general sense, implies that plant counting is a local visual problem.
5) On the WED data set, the V3 series indicates a clear advantage over the detection baseline Faster R-CNN, with significantly fewer model parameters and lower counting errors.
6) SFC^{2}Net slightly outperforms V3-Seg variants on the RPC data set. We think the improvement comes from the use of a classification-based counter [41]. However, this counter does not provide consistent improvements on the other three data sets. In addition, SFC^{2}Net also suffers from the problem of poor visualization akin to V2.
7) Compared with V2 that only visualizes coarse regions, the output visualization of the V3 series is more explainable to allow one to infer which instance is counted.
8) Per Fig. 10, one can see the effect of background suppression of V3-Seg†, which suggests the segmentation branch works.
9) The visualization of CSRNet is also satisfactory when images do not present scale changes (the MTC-UAV and the WED data set), but its counting accuracy falls behind V3-Seg. The reason is that the pixel-level density map is not robust to size variations of plants, which introduces training ambiguities.
10) ImageNet-pretrained models should be used in plant counting whenever computational budgets are allowed.
11) It is the first time that $R^2 > 0.9$ is reported on the MTC data set. Given remarkable R^2 values on other data sets, the V3 series appears to be a strong plant counting tool.
12) In Table IV, V3-Seg‡ has even lower R^2 than V3-Lite, which does not appear in other data sets. One plausible explanation is that encoding large-receptive-field background context as in V3-Seg‡ may have a negative effect when background information differs (only the MTC data set follows the cross-domain setting such that background regions between the training set and testing set are significantly different).

D. Ablation Study

Here, we conduct several ablation studies to justify some key design choices in the TasselNetV3 series.

1) Mixed Convolution, Dynamic Unfolding, and Imagenet Pretraining: We first validate the use of mixed convolution, dynamic unfolding, and ImageNet pretraining. We present a step-by-step analysis of the MTC-UAV data set. Results are shown in Table VII. One can see that each component has an independent contribution to performance improvement. Among them, ImageNet pretraining brings the most significant improvement, reducing the MAE from 17.7 to 13.1. For dynamic unfolding, while its effect is positive, it is mainly

TABLE III
PERFORMANCE OF DIFFERENT METHODS ON THE MTC-UAV DATA SET

Method	Venue, Year	Backbone	#Param.	MAE	RMSE	rMAE	rRMSE	R^2
MCNN	CVPR 2016	—	0.1M	34.5	43.1	24.4%	35.0%	0.8954
CSRNet	CVPR 2018	VGG16	16.3M	11.8	17.1	7.7%	12.7%	0.9809
BCNet	TCSVT 2019	VGG16	14.8M	11.7	17.5	7.2%	11.5%	0.9792
SFC^{2}Net	PLPH 2020	MixNet	8.3M	12.2	17.2	6.5%	8.5%	0.9802
TasselNetV2	PLME 2019	—	0.5M	13.9	22.9	7.5%	12.1%	0.9681
TasselNetV3-Lite	This paper	MixNet	0.3M	13.1	21.1	7.4%	11.8%	0.9739
TasselNetV3-Seg†	This paper	MixNet	7.5M	8.8	13.5	5.2%	7.8%	0.9875
TasselNetV3-Seg‡	This paper	MixNet	7.7M	10.0	14.8	5.8%	9.3%	0.9855

TABLE IV
PERFORMANCE OF DIFFERENT METHODS ON THE MTC DATA SET

Method	Venue, Year	Backbone	#Param.	MAE	RMSE	rMAE	rRMSE	R^2
MCNN	CVPR 2016	—	0.1M	17.9	21.9	273.4%	692.5%	0.3288
CSRNet	CVPR 2018	VGG16	16.3M	6.9	11.5	77.8%	190.3%	0.8221
BCNet	TCSVT 2019	VGG16	14.8M	5.2	9.2	31.8%	62.5%	0.8803
SFC^{2}Net	PLPH 2020	MixNet	8.3M	5.0	9.4	17.7%	24.6%	0.8866
TasselNet	PLME 2017	—	0.5M	6.6	9.9	44.8%	89.9%	0.8659
TasselNetV2	PLME 2019	—	0.5M	5.4	9.2	31.9%	69.5%	0.8923
TasselNetV3-Lite	This paper	MixNet	0.3M	4.9	9.1	17.5%	25.2%	0.9054
TasselNetV3-Seg†	This paper	MixNet	7.5M	4.0	6.9	19.6%	35.2%	0.9396
TasselNetV3-Seg‡	This paper	MixNet	7.7M	4.5	8.9	20.0%	35.2%	0.8977

TABLE V
PERFORMANCE OF DIFFERENT METHODS ON THE WED DATA SET

Method	Venue, Year	Backbone	#Param.	MAE	RMSE	rMAE	rRMSE	R^2
MCNN	CVPR 2016	—	0.1M	11.5	15.6	8.4%	11.0%	0.3792
CSRNet	CVPR 2018	VGG16	16.3M	4.2	5.2	3.2%	4.1%	0.9358
BCNet	TCSVT 2019	VGG16	14.8M	4.1	4.9	3.1%	3.8%	0.9371
Faster R-CNN	AGRFORMET 2019	InceptionResNetV2	54.3M	4.6	5.9	–	5.3%	0.9100
SFC^{2}Net	PLPH 2020	MixNet	8.3M	4.2	5.1	3.2%	3.9%	0.9310
TasselNet	PLME 2017	No	0.5M	6.8	8.3	–	7.1%	0.7900
TasselNetV2	PLME 2019	No	0.5M	5.3	6.8	4.1%	5.3%	0.9000
TasselNetV3-Lite	This paper	MixNet	0.3M	4.6	5.7	3.5%	4.3%	0.9372
TasselNetV3-Seg†	This paper	MixNet	7.5M	3.6	4.6	2.7%	3.6%	0.9420
TasselNetV3-Seg‡	This paper	MixNet	7.7M	4.0	5.0	3.1%	4.0%	0.9415

TABLE VI
PERFORMANCE OF DIFFERENT METHODS ON THE RPC DATA SET

Method	Venue, Year	Backbone	#Param.	MAE	RMSE	rMAE	rRMSE	R^2
MCNN	CVPR 2016	—	0.1M	92.1	121.5	15.3%	19.8%	0.8903
CSRNet	CVPR 2018	VGG16	16.3M	49.2	74.6	7.5%	10.5%	0.9135
BCNet	TCSVT 2019	VGG16	14.8M	31.3	49.8	4.8%	6.4%	0.9641
SFC^{2}Net	PLPH 2020	MixNet	8.3M	25.5	38.1	3.8%	5.4%	0.9765
TasselNetV2	PLME 2019	—	0.5M	51.5	92.2	7.4%	11.8%	0.8780
TasselNetV3-Lite	This paper	MixNet	0.3M	43.2	66.4	6.1%	8.5%	0.9395
TasselNetV3-Seg†	This paper	MixNet	7.5M	35.0	51.0	5.2%	7.0%	0.9605
TasselNetV3-Seg‡	This paper	MixNet	7.7M	30.8	46.3	4.5%	6.2%	0.9657

designed to address the explainability issue of the visualization. Mixed convolution also benefits counting.

2) Parametric Decoding Outperforms Nonparametric Decoding: Here, we demonstrate why parametric decoding with ASPP and CARAFE is used to encode context and for upsampling. We compare parametric decoding with a nonparametric decoding counterpart—PPM and bilinear interpolation. Since context is closely related to the input

TABLE VII

COMPARISONS OF ENCODER/COUNTER CHOICES ON THE MTC-UAV DATA SET. `scratching` AND `pretraining` DENOTE TRAINING FROM SCRATCH AND IMAGENET PRETRAINING, RESPECTIVELY

Encoder	Counter	Training	MAE	RMSE	rMAE	rRMSE	R^2
plain conv	standard	scratching	19.4	29.1	9.9%	14.1%	0.9571
plain conv	dynamic unfolding	scratching	18.7	28.2	9.3%	13.2%	0.9595
mixed conv	dynamic unfolding	scratching	17.7	27.0	9.1%	13.0%	0.9623
mixed conv	dynamic unfolding	pretraining	13.1	21.1	7.4%	11.8%	0.9739

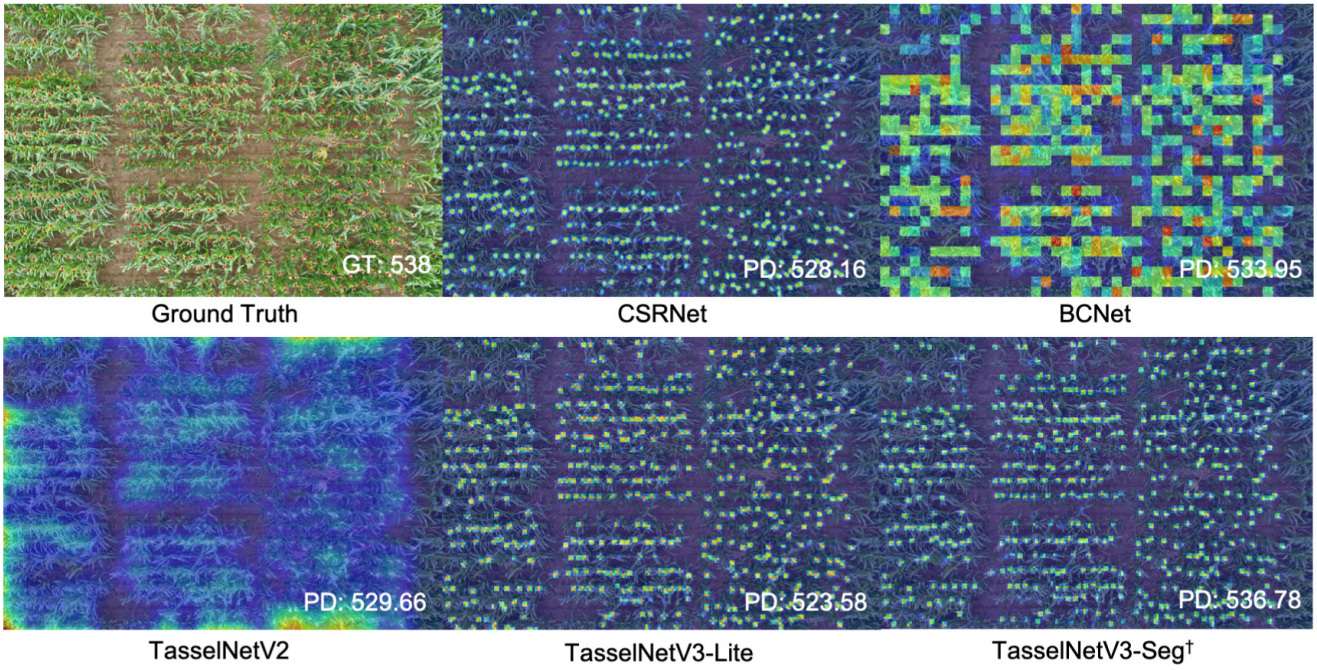

Fig. 9. Qualitative results on the MTC-UAV data set. GT denotes the ground-truth count and PD the predicted count.

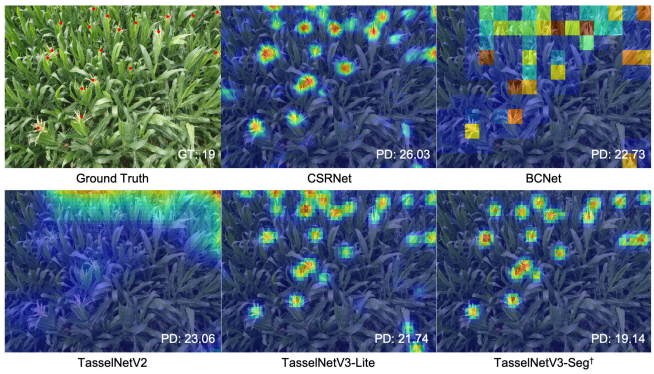

Fig. 10. Qualitative results on the MTC data set. GT denotes the ground-truth count and PD the predicted count.

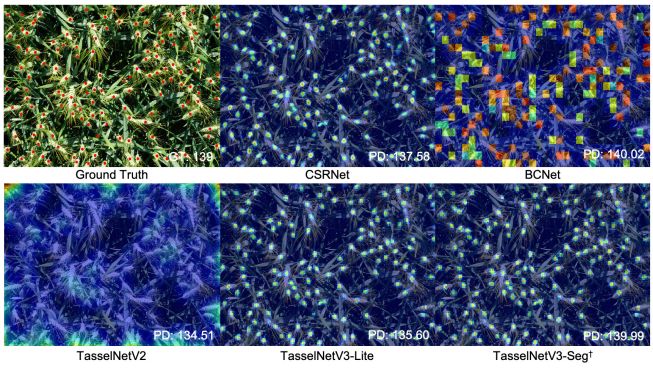

Fig. 11. Qualitative results on the WED data set. GT denotes the ground-truth count and PD the predicted count.

image size, we report results with varying training image sizes. Results are shown in Fig. 13. It can be observed that, increased training image sizes lead to decreased MAE, and parametric decoding generally achieves lower errors than nonparametric decoding. We think the reason is that parametric decoding provides further flexibility in encoding context.

3) How to Integrate a Priori Knowledge? From Early Fusion to Late Fusion: In Fig. 7, we implement a middle-fusion strategy to integrate *a priori* foreground–background knowledge. However, when we first approach this problem, it is not immediately clear what is the optimal choice to do so. In the open literature, there mainly exist three strategies to integrate external knowledge:

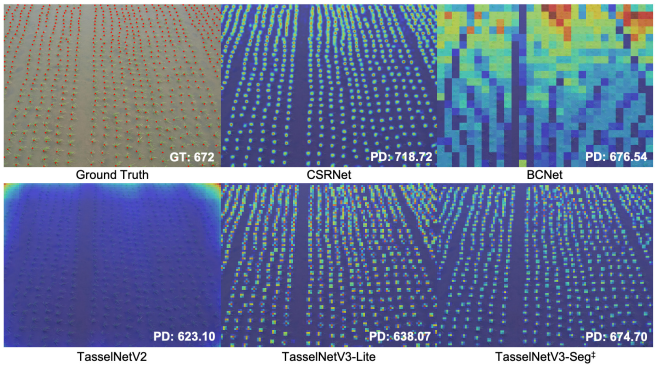

Fig. 12. Qualitative results on the RPC data set. GT denotes the ground-truth count and PD the predicted count.

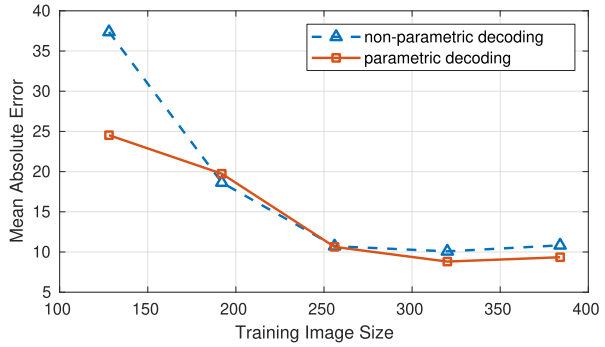

Fig. 13. Comparison between nonparametric decoding and parametric decoding. Parametric decoding with ASPP and CARAFE is generally better than nonparametric decoding with PPM and bilinear upsampling.

TABLE VIII
PERFORMANCE OF DIFFERENT FUSION STRATEGIES

Fusion	MAE	RMSE	rMAE	rRMSE	R^2
early	11.6	17.9	6.8%	10.4%	0.9787
middle	8.8	13.5	5.2%	7.8%	0.9875
late	10.5	14.8	6.2%	8.6%	0.9855

early fusion, middle fusion, and late fusion. Early fusion is often used to concatenate different forms of input [42], [48]. Middle fusion is used to enable multiplicative interactions between the prior input and the feature map, such as spatial attention [49]. Late fusion is often considered a postprocessing idea. Fig. 14 illustrates different fusion strategies that we implement. In early and late fusions, the segmentation map is binarized, while, in middle fusion, we use the soft segmentation map for smooth optimization. A comparison of different fusion strategies is shown in Table VIII. Middle fusion turns out to be the most effective one.

4) Benefit of Pseudomask Supervision and Guided Counting Loss: Finally, we justify the effectiveness of pseudomask supervision and guided counting loss. By disabling the supervision of pseudomask, the predicted segmentation map works in a way akin to the attention map, which gives the first baseline "w.o. L_{seg}." By disabling the guided counting loss, the counter is learned on both foreground and

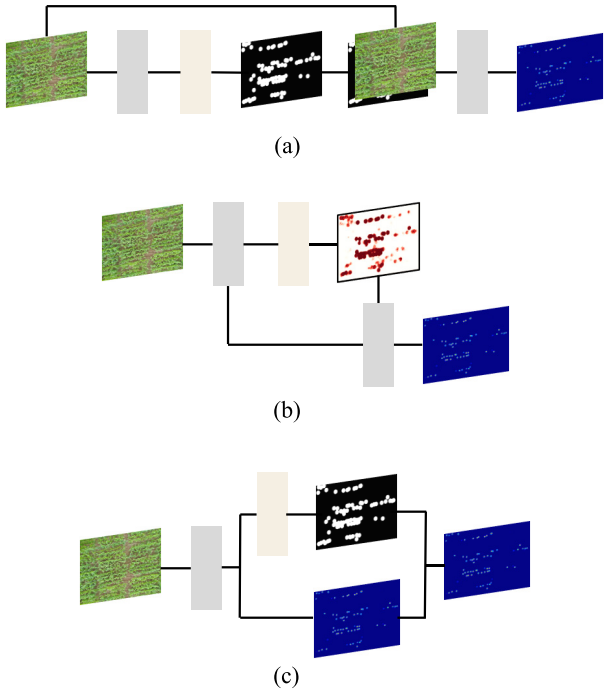

Fig. 14. Different fusion strategies. (a) Early fusion: the binarized segmentation map is concatenated with the input image to form a four-channel input to the counting branch. (b) Middle fusion: a soft segmentation map is used as an attention map to refine the encoder feature before feeding to the counter. (c) Late fusion: the binarized segmentation map and the count map are first generated, respectively, and then are fused by multiplication.

TABLE IX
EFFECT OF MASK SUPERVISION

Supervision	MAE	RMSE	rMAE	rRMSE	R^2
w.o. L_{seg}	15.3	22.0	9.1%	13.6%	0.9680
w.o. L_{guide}	13.2	18.0	6.6%	8.3%	0.9781
w. L_{seg} & L_{guide}	8.8	13.5	5.2%	7.8%	0.9875

background regions. This gives the second baseline "w.o. L_{guide}." Results are listed in Table IX. They suggest that both L_{seg} and L_{guide} matter. This is an interesting observation in TasselNetV3-Seg. A plausible explanation is that segmentation simplifies counting. In standard plant counting, the counter is required to tackle substantial background regions. A large number of background samples lead to significant sample imbalance such that overestimates in sparse regions frequently occur [41]. Segmentation alleviates this problem because background regions are no longer required to be fitted by the counter. TasselNetV3-Seg somewhat shares a similar spirit to the divide-and-conquer idea. Hence, if segmentation is assigned to tackle background regions, the counter should trust the segmenter that it can address the background issue. The counter then is supposed to focus on foreground regions; otherwise, sample imbalance remains. We believe that this is why decoupling the learning of the segmenter and the counter works.

E. Efficiency Comparison

We further compare the efficiency of different models. We use frames per second (FPS) as the metric. Fig. 15

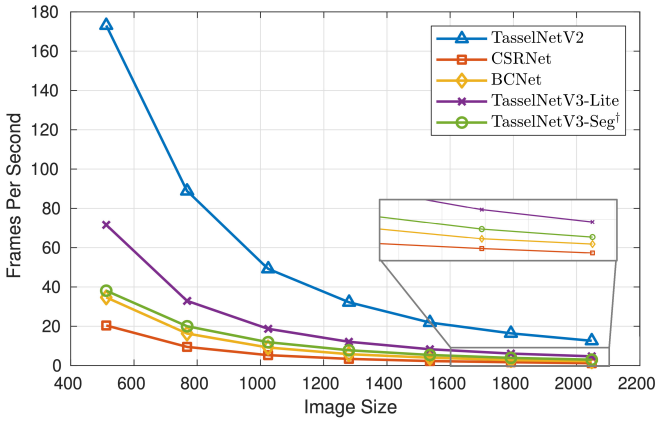

Fig. 15. Efficiency comparison of different models. FPS are averaged over 100 forward passes on `Image_Size×Image_Size×3` random input with Nividia GTX 1070 GPU, Intel i7-8700 CPU, and 16-GB RAM. The efficiencies of SFC^{2}Net and TasselNetV3-Seg‡ are identical to TasselNetV3-Seg†.

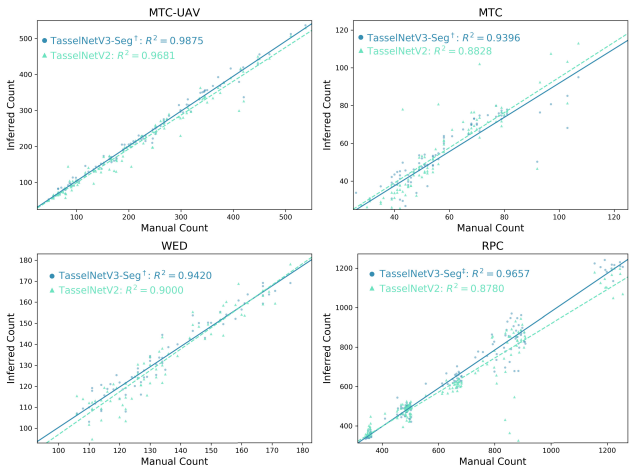

Fig. 16. Coefficients of determination of TasselNetV2 and the best performing TasselNetV3 on different data sets.

demonstrates the FPS of different methods with changing image sizes. TasselNetV2 is the most efficient one. While the model capacity of TasselNetV3-Lite is smaller than TasselNetV2, TasselNetV3-Lite is not sufficiently efficient because of the poor implementation of depthwise convolution in current deep learning libraries. Depthwise convolution does not achieve ideal acceleration over standard convolution. In high-capacity models, TasselNetV3-Seg† is not only superior over CSRNet and BCNet in accuracy but also faster than the two models in inference. Overall, with an affordable consumer-grade GPU Nividia GTX 1070, TasselNetV3-Lite is able to achieve real-time processing capability (30 FPS) on 768 × 768 RGB images, and TasselNetV3-Seg variants can process images of the same resolution with around 20 FPS. Further acceleration can be achieved with more advanced hardware.

F. Suggestions Toward Good Practices

The TasselNetV3 series provides a strong plant counting tool with practical applications. Here, we have also some suggestions for better practices.

1) In resource-constrained applications, we advocate the use of TasselNetV3-Lite because, in most cases, its counting performance is sufficiently accurate but is much efficient.
2) The best imaging view is from nadir because it does not introduce significant scale variations that complicate counting.
3) The imaging height should be kept consistent between training and testing because a plant captured at 10-m height is not simply a resized version of that plant captured at 20-m height. Appearance changes with changed scales. The current model is neither robust to scale variations nor to equivalent operations, such as image scaling.
4) TasselNetV3-Seg is encouraged to use at the early growth stage of plants. Compared with highly occluded scenes, segmentation is more reliable in separated plants.
5) One should not expect that plant sizes can be estimated reliably from dotted annotations. The TasselNetV3 series is recommended when only the population of plants is the index of interest.

G. Further Discussions

We remark that, despite good performance is reported, plant counting is not a solved problem. According to the plots of coefficients of determination shown in Fig. 16, while R^2 is high, one can see that there exist many overestimations and underestimations. Hence, the seemingly good image-level MAE is often an averaged effect of overestimated and underestimated counts. One should be aware of this issue when using a plant counting tool. In addition, the use of plant counting and proposed solutions should depend on the application at hand and circumstances, e.g., if the ultimate goal is crop prediction (preliminary counting of plants is not essential), the network can be trained to do the prediction task directly.

Aside from open challenges mentioned in Section III-B, another important challenge is called the *domain shift*. On the MTC-UAV data set, training data and testing data follow a similar distribution because images are captured under almost the same external environment, which conforms to the i.i.d. assumption in standard machine learning. This assumption, however, hardly meets in reality where testing data, also known as, target domain, exhibit a domain shift against training ones, also known as, source domain. Decreased counting performance is often observed when a trained model is tested on plants of different cultivars, on fields with different illumination/weather conditions, and on images captured by different cameras or at different heights. This is a typical cross-domain setting. In this setting, it is worth noting that the interpretation and observations may change. In the open literature, one feasible solution to mitigate domain shifts is to use a technique called transfer learning or domain adaptation [50]–[52].

To try domain adaptation ideas in plant counting, several difficulties still need to be taken into account: 1) collecting cross-domain plant data is expensive, and this,

sometimes, requires globalwise collaboration, e.g., the global wheat challenge [13] and 2) the adaptation approach requires further investigation. Compared with classification-based visual adaptation problems, regression-based ones are less studied. Existing approaches modeling classwise differences cannot be easily modified to fit a continuous regression problem, such as counting.

VI. Conclusion

In the local count framework, we introduce dynamic unfolding—a novel network module designed for the counter to generate human-interpretable output visualization. By augmenting the standard counter in TasselNetV2 with this module, we further propose the TasselNetV3 series: a lightweight variant TasselNetV3-Lite with a truncated MixNet-L encoder and two medium-capacity variants, TasselNetV3-Seg† and TasselNetV3-Seg‡, with an extra segmentation branch. We investigate alternative architecture designs, fusion strategies, and loss functions to make this idea work. We evaluate the TasselNetV3 series on four field-based plant counting data sets. Results indicate that the TasselNetV3 series significantly surpasses its baseline TasselNetV2 and also outperforms or at least is on par with other state-of-the-art object counters. Our evaluations also reveal some interesting findings that can provide guidance for agricultural practitioners to develop their customized plant counting models.

For future work, we plan to investigate cross-domain plant counting. In particular, aside from inbred lines considered in this work, we will include other hybrid lines. In addition, we will also consider location factors.

Acknowledgment

The authors would like to thank Shan-Ling Cao, Wei-Min Hao, and Mei-Yue Wang for their help in annotating the MTC-UAV data set.

References

[1] M. Weiss, F. Jacob, and G. Duveiller, "Remote sensing for agricultural applications: A meta-review," *Remote Sens. Environ.*, vol. 236, Jan. 2020, Art. no. 111402.

[2] M. V. Giuffrida, M. Minervini, and S. Tsaftaris, "Learning to count leaves in rosette plants," in *Proc. Comput. Vis. Problems Plant Phenotyping Workshop*, 2015, pp. 1.1–1.13.

[3] Y. Zhu, Z. Cao, H. Lu, Y. Li, and Y. Xiao, "In-field automatic observation of wheat heading stage using computer vision," *Biosystems Eng.*, vol. 143, pp. 28–41, Mar. 2016.

[4] C. Zhou, G. Yang, D. Liang, X. Yang, and B. Xu, "An integrated skeleton extraction and pruning method for spatial recognition of maize seedlings in MGV and UAV remote images," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 8, pp. 4618–4632, Aug. 2018.

[5] S. Madec *et al.*, "Ear density estimation from high resolution RGB imagery using deep learning technique," *Agricult. Forest Meteorol.*, vol. 264, pp. 225–234, Jan. 2019.

[6] S. Ghosal *et al.*, "A weakly supervised deep learning framework for sorghum head detection and counting," *Plant Phenomics*, vol. 2019, Jun. 2019, Art. no. 1525874.

[7] F. Fiorani and U. Schurr, "Future scenarios for plant phenotyping," *Annu. Rev. Plant Biol.*, vol. 64, no. 1, pp. 267–291, Apr. 2013.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[9] H. Lu, Z. Cao, Y. Xiao, B. Zhuang, and C. Shen, "TasselNet: Counting maize tassels in the wild via local counts regression network," *Plant Methods*, vol. 13, no. 1, pp. 79–95, Dec. 2017.

[10] H. Xiong, Z. Cao, H. Lu, S. Madec, L. Liu, and C. Shen, "TasselNetv2: In-field counting of wheat spikes with context-augmented local regression networks," *Plant Methods*, vol. 15, no. 1, p. 150, Dec. 2019.

[11] H. Xiong, H. Lu, C. Liu, L. Liu, Z. Cao, and C. Shen, "From open set to closed set: Counting objects by spatial divide-and-conquer," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 8362–8371.

[12] L. Liu, H. Lu, Y. Li, and Z. Cao, "High-throughput rice density estimation from transplantation to tillering stages using deep networks," *Plant Phenomics*, vol. 2020, pp. 1–14, Aug. 2020.

[13] E. David *et al.*, "Global wheat head detection (GWHD) dataset: A large and diverse dataset of high-resolution RGB-labelled images to develop and benchmark wheat head detection methods," *Plant Phenomics*, vol. 2020, pp. 1–12, Aug. 2020, Art. no. 3521852.

[14] J. A. Fernandez-Gallego, S. C. Kefauver, N. A. Gutiérrez, M. T. Nieto-Taladriz, and J. L. Araus, "Wheat ear counting in-field conditions: High throughput and low-cost approach using RGB images," *Plant Methods*, vol. 14, no. 1, p. 22, Dec. 2018.

[15] L. Malambo, S. Popescu, N.-W. Ku, W. Rooney, T. Zhou, and S. Moore, "A deep learning semantic segmentation-based approach for field-level sorghum panicle counting," *Remote Sens.*, vol. 11, no. 24, p. 2939, Dec. 2019.

[16] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollar, "Panoptic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9404–9413.

[17] V. N. Vapnik, *Statistical Learning Theory*, vol. 1. New York, NY, USA: Wiley, 1998.

[18] A. B. Chan, Z.-S. John Liang, and N. Vasconcelos, "Privacy preserving crowd monitoring: Counting people without people models or tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–7.

[19] V. Lempitsky and A. Zisserman, "Learning to count objects in images," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 1324–1332.

[20] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma, "Single-image crowd counting via multi-column convolutional neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 589–597.

[21] Y. Li, X. Zhang, and D. Chen, "CSRNet: Dilated convolutional neural networks for understanding the highly congested scenes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1091–1100.

[22] K. Chen, C. C. Loy, S. Gong, and T. Xiang, "Feature mining for localised crowd counting," in *Proc. Brit. Mach. Vis. Conf.*, 2012, pp. 21.1–21.11.

[23] J. P. Cohen, G. Boucher, C. A. Glastonbury, H. Z. Lo, and Y. Bengio, "Count-ception: Counting by fully convolutional redundant counting," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 18–26.

[24] X. Liu, J. Yang, and W. Ding, "Adaptive mixture regression network with local counting map for crowd counting," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 241–257.

[25] H. Lu, Y. Dai, C. Shen, and S. Xu, "Index networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, 2020.

[26] H. Lu, Z. Cao, Y. Xiao, Y. Li, and Y. Zhu, "Region-based colour modelling for joint crop and maize tassel segmentation," *Biosystems Eng.*, vol. 147, pp. 139–150, Jul. 2016.

[27] F. Cointault, D. Guerin, J.-P. Guillemin, and B. Chopinet, "In-field triticum aestivum ear counting using colour-texture image analysis," *New Zealand J. Crop Horticultural Sci.*, vol. 36, no. 2, pp. 117–130, 2008.

[28] F. Kurtulmus, W. S. Lee, and A. Vardar, "Green citrus detection using 'eigenfruit', color and circular Gabor texture features under natural outdoor conditions," *Comput. Electron. Agricult.*, vol. 78, no. 2, pp. 140–149, Sep. 2011.

[29] S. W. Chen *et al.*, "Counting apples and oranges with deep learning: A data-driven approach," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 781–788, Apr. 2017.

[30] S. Bargoti and J. Underwood, "Deep fruit detection in orchards," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 3626–3633.

[31] Y. Long, Y. Gong, Z. Xiao, and Q. Liu, "Accurate object localization in remote sensing images based on convolutional neural networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 5, pp. 2486–2498, May 2017.

[32] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[33] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.

[34] S. Sun *et al.*, "Three-dimensional photogrammetric mapping of cotton bolls *in situ* based on point cloud segmentation and clustering," *ISPRS J. Photogramm. Remote Sens.*, vol. 160, pp. 195–207, Feb. 2020.

[35] M. Rahnemoonfar and C. Sheppard, "Deep count: Fruit counting based on deep simulated learning," *Sensors*, vol. 17, no. 4, p. 905, Apr. 2017.
[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[37] S. Aich and I. Stavness, "Leaf counting with deep convolutional and deconvolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 2080–2089.
[38] J. Wu, G. Yang, X. Yang, B. Xu, L. Han, and Y. Zhu, "Automatic counting of *in situ* rice seedlings from UAV images based on a deep fully convolutional neural network," *Remote Sens.*, vol. 11, no. 6, p. 691, Mar. 2019.
[39] L. P. Osco *et al.*, "A convolutional neural network approach for counting and geolocating citrus-trees in UAV multispectral imagery," *ISPRS J. Photogramm. Remote Sens.*, vol. 160, pp. 97–106, Feb. 2020.
[40] H. Lu, Z. Cao, Y. Xiao, Z. Fang, and Y. Zhu, "Toward good practices for fine-grained maize cultivar identification with filter-specific convolutional activations," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 2, pp. 430–442, Apr. 2018.
[41] L. Liu, H. Lu, H. Xiong, K. Xian, Z. Cao, and C. Shen, "Counting objects by blockwise classification," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 10, pp. 3513–3527, Oct. 2020.
[42] H. Lu, Y. Dai, C. Shen, and S. Xu, "Indices matter: Learning to index for deep image matting," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3266–3275.
[43] M. Tan and Q. V. Le, "MixConv: Mixed depthwise convolutional kernels," in *Proc. Brit. Mach. Vis. Conf.*, 2019, pp. 1–14.
[44] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 801–818.
[45] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2117–2125.
[46] J. Wang, K. Chen, R. Xu, Z. Liu, C. C. Loy, and D. Lin, "CARAFE: Content-aware ReAssembly of FEatures," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3007–3016.
[47] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2881–2890.
[48] A. Milioto, P. Lottes, and C. Stachniss, "Real-time semantic segmentation of crop and weed for precision agriculture robots leveraging background knowledge in CNNs," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 2229–2235.
[49] H. Zhao *et al.*, "Psanet: Point-wise spatial attention network for scene parsing," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 267–283.
[50] H. Lu, Z. Cao, Y. Xiao, and Y. Zhu, "Two-dimensional subspace alignment for convolutional activations adaptation," *Pattern Recognit.*, vol. 71, pp. 320–336, Nov. 2017.
[51] H. Lu *et al.*, "When unsupervised domain adaptation meets tensor representations," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 599–608.
[52] H. Lu, C. Shen, Z. Cao, Y. Xiao, and A. van den Hengel, "An embarrassingly simple approach to visual domain adaptation," *IEEE Trans. Image Process.*, vol. 27, no. 7, pp. 3403–3417, Jul. 2018.

Hao Lu received the Ph.D. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2018.

He was a Post-Doctoral Fellow with the School of Computer Science, The University of Adelaide, Adelaide, SA, Australia. He is an Associate Professor with the School of Artificial Intelligence and Automation, Huazhong University of Science and Technology. His research interests include computer vision and machine learning. He is working on object counting and dense prediction problems.

Liang Liu received the B.S. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2016, where he is pursuing the Ph.D. degree with the School of Artificial Intelligence and Automation.

His research focuses on object counting and its applications.

Ya-Nan Li received the B.S. degree from Wuhan University, Wuhan, China, in 2012 and the Ph.D. degree from the Huazhong University of Science and Technology, Wuhan, in 2018.

She is a Lecturer with the School of Computer Science and Engineering, Wuhan Institute of Technology, Wuhan. Her research interests include computer vision and machine learning, with particular emphasis on image segmentation, domain adaptation, and various computer vision applications in agriculture.

Xiao-Ming Zhao received the B.S. and M.S. degrees from Jinlin Agricultural University, Changchun, China, in 2006 and 2009, respectively.

He is a Field Testing Manager with the Center for Crop Functional Genomics and Molecular Breeding, China Agricultural University, Beijing, China. He has over ten years' experience in field testing. His research interests include high-throughput field testing and phenotype unmanned aircraft vehicle (UAV).

Xi-Qing Wang received the B.S., M.S., and Ph.D. degrees from Northwest A&F University, Yangling, China, in 1990, 1993, and 1998, respectively.

He is a Professor with the Center for Crop Functional Genomics and Molecular Breeding, China Agricultural University, Beijing, China. He has over 30 years' experience in academic institutions and agricultural industries. His research interests are in the areas of agriculture and phenomics robots.

Zhi-Guo Cao (Member, IEEE) received the B.S. and M.S. degrees in communication and information systems from the University of Electronic Science and Technology of China, Chengdu, China, in 1985 and 1990, respectively, and the Ph.D. degree in pattern recognition and intelligent system from the Huazhong University of Science and Technology, Wuhan, China, in 2001.

He is a Professor with the School of Artificial Intelligence and Automation, Huazhong University of Science and Technology. He has authored dozens of papers at international journals and conferences, which have been applied to automatic observation systems for object recognition in video surveillance systems, for crop growth in agriculture, and for weather phenomena in meteorology based on computer vision. His research interests spread across image understanding and analysis, depth information extraction, and object detection.

Dr. Cao's projects have received provincial or ministerial level awards of Science and Technology Progress in China.