agriculture

MDPI

Article

Tobacco Plant Detection in RGB Aerial Images

Xingping Sun, Jiayuan Peng, Yong Shen * and Hongwei Kang

School of Software, Yunnan University, Kunming 650000, China; sunxp@ynu.edu.cn (X.S.); pdunwannaplay@163.com (J.P.); hwkang@ynu.edu.cn (H.K.)
* Correspondence: sheny@ynu.edu.cn; Tel.: +86-133-0880-6539

Received: 2 February 2020; Accepted: 26 February 2020; Published: 28 February 2020

Abstract: Tobacco is an essential economic crop in China. The detection of tobacco plants in aerial images plays an important role in the management of tobacco plants and, in particular, in yield estimations. Traditional yield estimation is based on site inspections, which can be inefficient, time-consuming, and laborious. In this paper, we proposed an algorithm to detect tobacco plants in RGB aerial images automatically. The proposed algorithm is comprised of two stages: (1) A candidate selecting algorithm extracts possible tobacco plant regions from the input, (2) a trained CNN (Convolutional Neural Network) classifies a candidate as either a tobacco-plant region or a nontobacco-plant one. This proposed algorithm is trained and evaluated on different datasets. It demonstrates good performance on tobacco plant detection in aerial images and obtains a significant improvement on AP (Average Precision) compared to faster R-CNN (Regions with CNN features) and YOLOv3 (You Only Look Once v3).

Keywords: tobacco plant; region proposal; object detection; convolutional neural network

1. Introduction

Tobacco is native to South America [1] and it is widely cultivated in the southern and northern provinces of China. Not only can it be made into cigarettes, it also has a variety of significant medical properties [2]. Yield estimation plays an important role in the management of tobacco planting and in the agriculture precision, thus motivating many studies [3,4].

Traditional yield estimation is based on manual identification and counting, which can be time-consuming and laborious. In the last decade, there have been studies on how to estimate production using remote sensing data [5,6] which can be costly and computationally expensive. For these two reasons, finding a low cost and efficient method to automatically estimate the yield of tobacco plants is both urgent and necessary. Taking advantage of aerial images, detection of tobacco plants in them is a solution for automatic tobacco yield estimation. Aerial images provide useful resources to get an outlook of a vast area from a direct-down position. Due to the popularization of technology, aerial images captured by unmanned aerial vehicles are now both cost effective and of a guaranteed quality. Therefore, there are now many researches on estimating production by treating it as an object detection task in aerial images [4,7].

Object detection is one of the most exciting fields in computer vision. In 2009, Felzenszwalb et al. proposed the DPM (Deformable Parts Model) [8] where HOG (Histograms of Oriented Gradients) [9] are used as a feature extractor and SVM (Support Vector Machine) [10] is used as a classifier. Though DPM achieves good results, it searches the input image with sliding window [11–13] which makes it time-consuming and inefficient. Therefore, region proposal algorithms such as selective search [14] and edge boxes [15], were proposed to replace the search stage. Selective search combines exhaustive search with segmentation which sharply decreases the number of proposals and maintains proposals' quality at the same time. Edge boxes propose regions based on windows' edge information, which is faster and

more accurate. In 2014, on the background of rise of region proposal algorithms and CNN (Convolution Neural Networks), R-CNN (Regions with CNN features) was proposed [16,17]. It arranges a detection problem into three stages. (1) Get regions of interests by selective search, (2) extract features of these regions with CNN, (3) classify these features by multiple SVM classifiers. Later versions of R-CNN replaced selective search with RPN (Region Proposal Network) [18–20], which increases the detection speed and also maintains its accuracy. YOLO (You Only Look Once) was proposed then in 2016 [21]. Compared with R-CNN, YOLO aborts the region proposal stage which makes it feasible for real time detection but less accurate. Improvements of YOLO were proposed in [22,23] where YOLOv3 significantly improves the performance while maintaining its real-time properties.

Though these detection systems perform excellent on public datasets and competitions, they are not quite suitable for our problem. For one thing, tobacco plants in aerial images are small in size (37×37 pixels averagely). The systems we mentioned above tend to have poor localization performances for small objects. For the other, unlike public datasets who have millions of images for training, our dataset is not that sufficient. Therefore, in this paper, we proposed a method to detect tobacco plants in aerial images. We continued to use the basic idea of R-CNN, however, we have adjusted some of its parts to suit our problem. The differences are: (1) A tobacco plant region selecting algorithm is proposed to replace selective search in the region proposal stage, and (2) CNN is used as a feature extractor in R-CNN and the regions are classified by multiple binary SVM. In our method, CNN is directly used as a classifier. An overview of our model is shown in Figure 1.

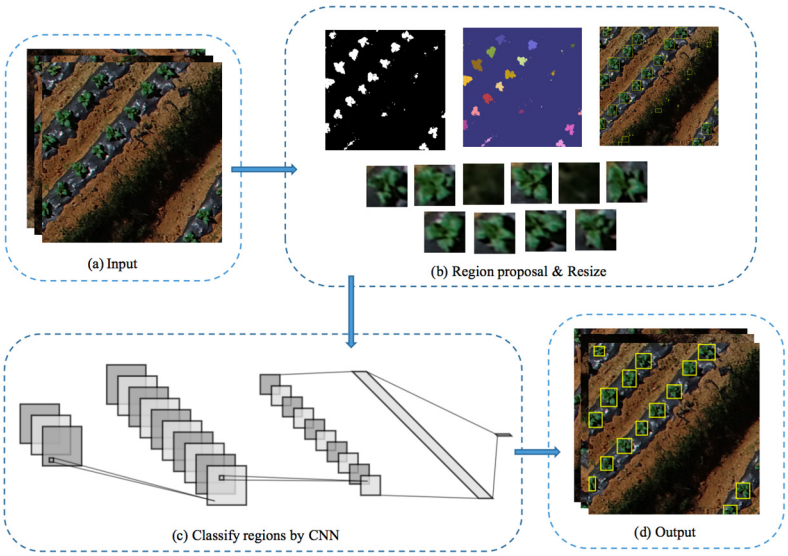

Figure 1. An overview of our proposed model. (**a**) The model takes in an input, (**b**) regions are proposed by our candidate selecting algorithm which is detailed in Section 3.1, (**c**) proposed regions are classified by a trained network which is illustrated in Section 3.2, (**d**) the model outputs the positive regions.

2. Materials

We have got five datasets provided by Yunnan TianYi Inc. Each dataset has 88 colored aerial images at a resolution of 72 ppi (pixels per inch) with 512 pixels in width and 512 pixels in height. These images were taken at a shooting height of 46.3 m in Tuogu Village, Xuanwei City, Yunnan Province which lies between 103° 06′ to 104° 13′ E and 26° 63′ to 27° 54′ N. We used two of the datasets to train and the remainder to evaluate. There are 1013 tobacco plants in the dataset, 386 of them are in the training set while 627 of them are in the testing set. Some aerial images in the datasets are shown in Figure 2. The following paragraphs of this section will detail the preprocessing work we have done on the datasets.

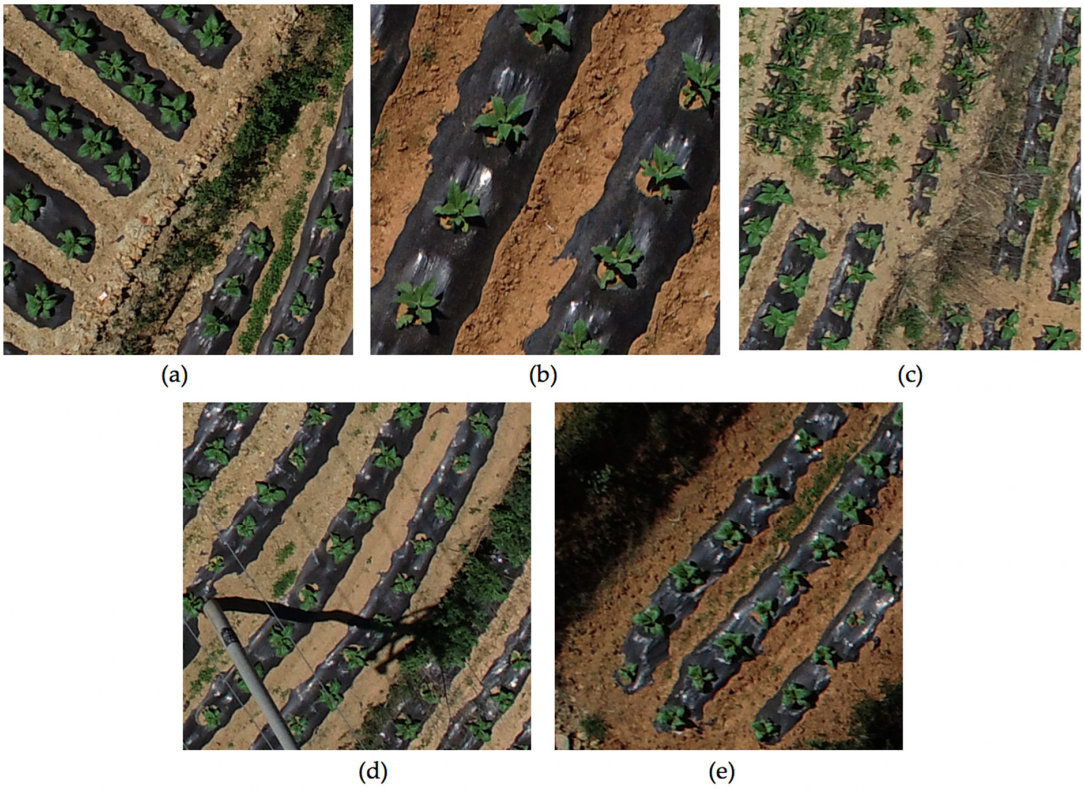

Figure 2. An overview on the datasets. (**a**–**e**) Come from dataset 1–5, respectively.

2.1. Data Annotation

To ensure the data's reliability, we have our datasets annotated by students from the School of Ecology of Yunnan University who have better knowledge of tobacco plants. Ground truth bounding boxes generate from annotations. A ground truth bounding box records an annotation by its top-left and bottom-right coordinates, which can be denoted as $(x_1,\ y_1,\ x_2,\ y_2)$. Ground truth bounding boxes provide useful information both in the evaluation stage and in the training stage.

2.2. Data Augmentation

We need to provide sufficient training samples to help networks obtain better learning weights, as well as prevent them from overfitting. Therefore, we augmented the dataset by rotation and flipping [24]. Augmentation is not only the transformation of images, but also the change of the corresponding ground truth bounding boxes. Denoted an original bounding box as $(x_1,\ y_1,\ x_2,\ y_2)$, transformations of it in different augmentations are as follows:

$$\left(x_1^{90},\ y_1^{90},\ x_2^{90},\ y_2^{90}\right) = (height - y_1, x_1,\ height - y_2, x_2) \tag{1}$$

$$\left(x_1^{180},\ y_1^{180},\ x_2^{180},\ y_2^{180}\right) = (width - x_1, height - y_1, width - x_2,\ height - y_2) \tag{2}$$

$$\left(x_1^{270},\ y_1^{270},\ x_2^{270},\ y_2^{270}\right) = (\ y_1, width - x_1,\ y_2, width - x_2) \tag{3}$$

$$\left(x_1^{h},\ y_1^{h},\ x_2^{h},\ y_2^{h}\right) = (\ width - x_1, y_1,\ width - x_2, y_2) \tag{4}$$

$$\left(x_1^{v},\ y_1^{v},\ x_2^{v},\ y_2^{v}\right) = (\ x_1, height - y_1,\ x_2, height - y_2) \tag{5}$$

where the *width* and *height* refer to the size of image, which are both 512 in our case. Equations (1)–(5) show the bounding box's transformation formulas of rotation by 90, 180, 270°, and flipping horizontally and vertically, respectively.

By rotating and flipping, an original image is augmented into 12 different ones. In other words, by augmentation, we have 2112 images in the training set with 4632 target objects, which is now sufficient to guarantee a good training result for networks.

3. Methods

Our model consists of two steps. Step one is selecting regions of interest by a candidate selecting algorithm. The second step is classifying the resized candidates with a trained CNN. The details are shown in this section.

3.1. Candidates Selecting Algorithm

The first stage of our model is extracting regions of interest using a candidate selecting algorithm, which sharply shrinks the number of regions to which our network must pay attention. This algorithm is composed of three steps: (1) Binarizing the image based on color, (2) grouping the pixels in the binary image, and (3) extracting and resizing the regions. Figure 3 shows the process.

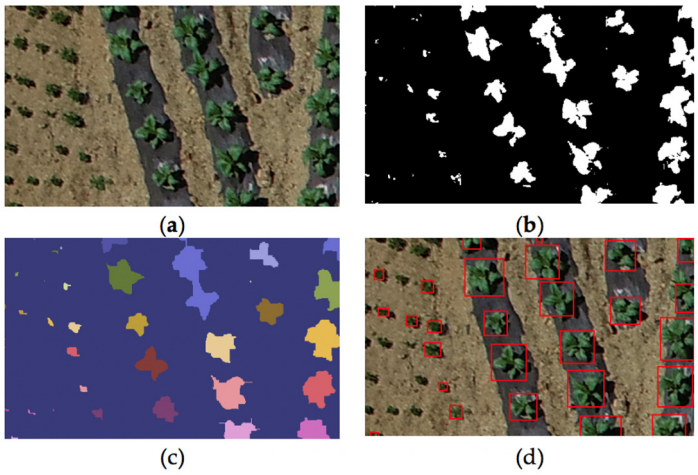

Figure 3. Process of candidate selecting algorithm. (**a**) The algorithm takes in an input, (**b**) binarizes the input by color, (**c**) groups the binary image, and (**d**) extracts the regions in the original image.

3.1.1. Binarization

Aerial images were read in the color space of RGB, which has three color channels of red, green, and blue. Different values in every channel form different colors to a naked eye. This binarizing step reserves the green-looking pixels while discarding the others. Concretely, the green pixels are changed to one while others are changed to zero, thus forming the binary image. We define a pixel as a green one by thresholding the extent to which a pixel's value in green channel is larger than its other two channels. The corresponding equations are listed as follows:

$$E_{gr} = I(r,\ c,\ \mathrm{G}) - I(r,c,\mathrm{R}) \tag{6}$$

$$E_{gb} = I(r,\ c,\ \mathrm{G}) - I(r,c,\mathrm{B}) \tag{7}$$

$$B(r,c) = \begin{cases} 1, & if\ E_{gr}(r,c) > thresh\ and\ E_{gb}(r,c) > thresh \\ 0, & else \end{cases} \tag{8}$$

where *I(r,c)* refers to the pixel that locates at row *r* and col *c* in the image matrix, and R, G, B refer to its value in red, green, and blue channel, respectively. E_{gr}*(r,c)* is the extent to which a pixel's green channel value is larger than its red channel value; E_{gb}*(r,c)* is the extent to which a pixel's green channel value is larger than its blue channel value; and *B(r, c)* refers to the pixel in the binary matrix, which is assigned a value of one if its E_{gr} and E_{gb} are larger than a threshold; otherwise it will be assigned a value of zero.

3.1.2. Grouping

The binary image can be regarded as a one-hot matrix, and in this step, we need to group the hot values. Connected hot points make a group. We have considered two methods for the grouping process. One is based on a searching algorithm and the other is based on clustering.

The common searching algorithms' ideas can be used here to ascertain a group, and we used BFS (Breadth First Search) [25] in our case. A round of search begins at a hot point in the matrix, and every search point has four search directions of left, top, right, and bottom if the corresponding point is hot, as well. Visited points will be marked in case of duplicated visiting.

A density-based clustering algorithm is also suitable for the grouping problem and we used DBSCAN (Density-Based Spatial Clustering of Application with Noise) [26] in our case. Concretely, we clustered hot points based on their coordinates under the parameters of $radius = 1$ and $minimum_{samples} = 5$. Not only can this DBSCAN-based method solve the grouping problem, it will also remove some anomaly hot points that do not satisfy the parameters which reduce the burden on the latter classification process to a certain extent.

3.1.3. Extraction and Resizing

After the grouping is completed, in the third step, we are to extract regions from the original aerial image according to groups. Regions are rectangles that can be ascertained by the coordinates of its left-top and right-bottom points. A searched group is composed of points, among which we denote the minimum value of its coordinate systems as x_{min}, y_{min} and the maximum ones as x_{max}, y_{max}, thus the region of a searched group can be represented as $(x_{min}, y_{min}, x_{max}, y_{max})$.

Before these extracted regions are inputted into the latter neural network, they need to be resized. That is because we aim to handle them with one specific classification model that accepts a fixed-size input. The size that all the regions should be resized to can be regarded as a parameter that can be adjusted. In our case, we resized each region to a squared shape of size 28×28. We also discard regions that are too small to be a tobacco plant. An extracting sample is shown in Figure 3.

3.2. Neural Network

Getting fixed-size regions from the former stage, in this stage, we are to tell whether each region is a tobacco plant region or not using a CNN-based model. This section outlines three parts. In the first part, we describe the functions of layers in CNNs. Then, we detail the architecture of our network and why we built it that way. Additionally, the details of the training process are shown in this section.

3.2.1. Layers in CNNs

Although CNNs are becoming deeper ever since deep neural networks [16] got big success on ImageNet [27,28], their architectures are quite similar. They are mostly composed of convolutional layers, pooling layers, and fully connected layers and our model continued to use this basic architecture.

Filters form a convolutional layer. As shown in Figure 4, the filters walk through the input and conduct convolution operations with the overlapping area each step they take. A convolutional layer is often used to extract features and it works similarly to DPM [8]. Filters can be viewed as patterns in DPM, they go through the input to determine whether it has a certain feature. Additionally, areas that match a certain feature will get larger outputs. However, the weights of filter are learned in the back-propagation process of the network while DPM's patterns are calculated by HOG [9]. Many state-of-the-art classification models [16,27,28] have shown convolutional layers' powerful ability in feature extraction.

Pooling layers are used to down sample the feature map. There are two major ways for pooling: Max pooling and average pooling [29]. The pooling operation is also based on a sliding window which is called a pool here. The pool goes through the inputted feature map, and the pooling operation is conducted in the overlapping area of the pool. For max pooling, the pooling operation outputs the

maximum value of a matrix while the average pooling operation outputs the average value of a matrix. The effectiveness of a pooling layer indicates that the relative positions of features are much more important than the absolute positions of them. Not only does the pooling layer down sample the feature map, it also reduces the parameters of a neural network model. Typically, a pooling layer is inserted into a set of convolutional layers periodically [16,27,28].

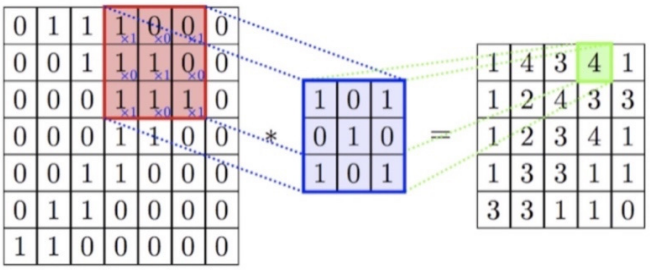

Figure 4. Calculation process of a filter in the convolutional layer. The first matrix is the input, the middle one is the filter, and the right one is the result. The highlighted regions show the calculation process of a step taken by the filter.

There are 2–3 fully connected layers at the end of a classification model. The fully connected layers are used to classify the features and the last layer is often activated by a SoftMax function [30]. The SoftMax function is defined in Equation (9), it normalizes an input vector into values ranging from 0 to 1 whose summation is 1. For this characteristic, a SoftMax function is often used to activate the last layer of a classification model.

$$f(z_j) = \frac{e^{z_j}}{\sum_{i=1}^{k} e^{z_k}} \tag{9}$$

where z is the input vector of k dimension.

3.2.2. Network Architecture

Faced with our current problem, in this stage of the detection system, we required CNNs that could accomplish a binary classification task of determining whether a proposed region is a tobacco plant region or not. We are not using the state-of-the-art classification CNNs architectures [16,27,28] given that the input we take here is much smaller. A shallow CNN that is built with the basic layers we mentioned in the last paragraph can solve our problem well within reasonable time. We take Lecun's digit classification model [31] as a reference, considering its input size is close to ours. Two modifications have done to it: (1) Replace the sigmoid activation function with ReLU (Rectified Linear Unit) which has been proven to be more suitable for the neural network's back-propagation process [32]. (2) The number of neurons in the last fully connected layer is changed to two since we are facing a binary classification problem. The network architecture is shown in Figure 5. The input of network is fixed-size color images which are extracted regions from an aerial image. The subsequent layers are defined as follows.

1. The convolutional layer has 20 filters of size $3 \times 3 \times 3$ pixels. The inputs are padded with zero, and the stride of filter is one. We use ReLU as its activation function. This layer is supposed to extract features of the input.
2. The previous layer's output is a matrix of size $28 \times 28 \times 20$. It is processed here by a pooling layer whose pool size is 2×2, and its stride is two. After the pooling process we have mentioned in the last paragraph, the feature map is down sampled to half of its original size.
3. Former max pooling layer outputs matrix of size $14 \times 14 \times 20$. Before it is inputted into the fully connected layer, it needs to be flattened. Thus, right after the max pooling layer is a layer that flattens the matrix into a vector with a dimensionality of 3920.
4. The first fully connected layer contains 256 neurons. It is activated by ReLU, as well.

5. The second fully connected layer contains two neurons and it is activated by SoftMax function. It outputs a two-dimensional vector, where the first dimension indicates the input's probability of being a tobacco plant, and the second dimension indicates the probability of a nontobacco plant.

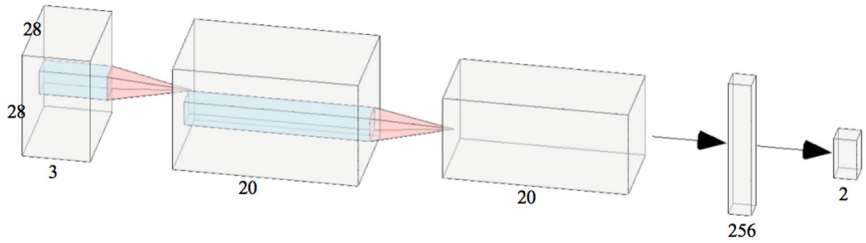

Figure 5. Network architecture. The input image is a colored region that we extract from the original aerial image. The first layer is a convolutional layer with 20 filters. The convolutional layer is connected to a max pooling layer whose pool-size is 2×2. After the pooling layer are two fully connected layers.

3.2.3. Training of Network

The network is trained with regions proposed by the candidate selecting algorithm. In the training process, we need to annotate the regions. Prior to the introduction of our annotation method, we need to make clear the concept of IoU (Intersection over Union). IoU is a metric used to measure how good a predicted bounding box is, and its calculation formula is shown in Equation (10). Figure 6 shows a visualization of it.

$$IoU = \frac{B_{pred} \cap B_{gt}}{B_{pred} \cup B_{gt}} \tag{10}$$

where B_{pred} is the area of a predicted bounding box, and B_{gt} is the area of a ground-truth bounding box.

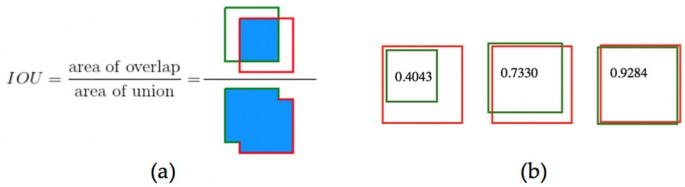

Figure 6. Visualizations of IoU (Intersection over Union): (**a**) The visualization of IoU calculation formula, it is the fraction of the overlapping area and the union area of two bounding boxes, (**b**) the ground truth bounding boxes are shown in red, predicted bounding boxes are shown in green, and their IoUs are shown in the boxes.

The performance of network depends heavily on the training samples, so it is very important to give network training samples with high quality. To ensure the samples' quality, we do not manually classify the regions, we explicit the IoU metric to decide whether a region is positive or not [19]. Concretely, a proposed region's IoU can be calculated since we have got the manually annotated ground truths in the preprocessing stage. A proposed region who has an IoU larger than 0.7 will be classified as a positive sample. Additionally, a region with an IoU smaller than 0.3 will be classified as a negative one. Regions with IoU between 0.3 and 0.7 are not involved in the training process. Figure 7 shows the training samples filtered by an IoU where the first row is the negative sample and the second row is the positive one. We can see that this sample selecting method efficiently finds training samples of high quality.

Figure 7. Training samples selected by IoU (Intersection over Union). The top row shows the negative samples and the bottom row shows the positive ones.

Having a balanced number of positive and negative samples is also important. An imbalanced training sample may cause an imbalanced result. In our case, the number of negative samples is much larger than that of positive ones. To achieve a good performance, we must guarantee an adequate number of samples for training. For these two reasons, we augmented the positive samples by rotation and flipping in a way shown in the preprocessing stage (Section 2).

4. Experiments and Results

To evaluate our proposed method, we compared it with some state-of-the-art algorithms and models in this section. For the tobacco region proposal algorithm, we compared it with selective search and edge boxes. For the whole detection model, we compared it with faster R-CNN [19] and YOLOv3 [22]. Details of the experiments and results will be shown following of this section.

4.1. Environment

Our experiment environment is a MacBook Pro 2017 (Apple Inc., Cupertino, California, CA, USA) with a CPU of 2.3 GHz Intel Core i5 and 8 G memory. Selective search and edge boxes are implemented with APIs (Application Programing Interfaces) provided by OpenCV. Faster R-CNN and YOLOv3 are implemented by the code opened on Github [33,34]. In general, the image processing methods are accomplished with OpenCV, the neural network models are built with Keras and TensorFlow, and the matrix operation is performed by Numpy. Their specific versions are shown in Table 1. These open source tools simplified the implementation of algorithms to a large extent.

Table 1. The software environment of our experiment.

Keras	TensorFlow	Numpy	OpenCV
2.1.6	1.10.0	1.14.5	4.1.2

4.2. Metrics

A region proposal algorithm is always targeting at a good performance of detection, so the performance of a region proposal algorithm can be evaluated by the detection results [11]. Therefore, the region proposal algorithms and the detection models can be evaluated with the same metrics. Before diving into those metrics, we first need to provide certain definitions: (a) Confidence Score: The probability that a region is a tobacco plant, (b) IoU is the intersection area of a predicted box and a ground truth one, (c) TP, true positive, refers to a correctly classified positive sample, (d) FP, false positive, refers to a negative sample that is classified as positive, and (e) FN, false negative, refers to a positive sample that is classified as negative. Confidence score and IoU are to decide whether a detection is true positive or not. In our experiments, we continued to use the thresholds proposed in faster R-CNN [19] and YOLOv3 [22]—a detection result is considered to be TP if its confidence score is larger than 0.5 and IoU is larger than 0.4, otherwise the detection will be considered as a FP, and a missing tobacco plant is a FN.

One of our metrics is the PR (precision-recall) curve. (a) Precision is the fraction of samples classified as positive that are actually positive. (b) Recall is the fraction of actually positive samples that are classified as positive. A good model should have both high precision and high recall. However, precision and recall have an inverse increasing tendency [35]. Additionally, the PR curve is a trade-off between these two metrics. Concretely, the PR curve takes every sample's confidence score as a split point to reclassify samples where samples with a confidence score larger than the split point is classified as positive otherwise negative. Therefore, taking each sample as a split point, we will get a set of precision and recall values. Sorting the set in ascending order and taking recall values as horizontal coordinates while precision as vertical coordinates will get the PR curve. The PR curve of a good model will be convexed to the upper right, which means it remains a high precision at a high recall, and thus a good model will have a large area under the PR curve. The other one is AP (Average Precision). AP is based on the PR curve whose value is approximately the area under the PR curve. More concretely, it is the precision averaged across all unique recall levels. A recall level's precision is defined as the highest precision found for any recall that is larger than it. They can be calculated as follows:

$$Precision = \frac{TP}{TP + FP} \tag{11}$$

$$Recall = \frac{TP}{TP + FN} \tag{12}$$

$$p(r) = \max_{r > r'} p(r') \tag{13}$$

$$AP = \sum_{i=1}^{n} (r_{i+1} - r_i) p(r_{i+1}) \tag{14}$$

4.3. Evaluation on Region Proposal Algorithms

As a region proposal algorithm is always targeting at promoting the performance of detection, we firstly trained a binary classification CNN whose training data is the manually classified regions outputted by our method, selective search, and edge boxes. Then, a region proposal method is evaluated by the detection result. These three region proposal algorithms we compared here are all unsupervised, therefore all of the datasets can be used to evaluate them.

Firstly, we compared our region proposal algorithm that using different grouping algorithms, one is based on BFS and the other is based on DBSCAN. Their detection results on different datasets and the time they take to select candidates in one image in shown in Table 2. We can see that the BFS-based method and DBSCAN-based one were well matched in the detection performance. Generally, the DBSCAN method has a slightly better AP because areas that do not match its parameters have been removed in the proposal stage due to DBSCAN's anomaly detection characteristic. However, the BFS-based method was averagely 40 times faster than the DBSCAN based one. We determined that the time difference should be caused by the time complexity of them, where the BFS-based method has a time complexity of $O(4n)$ and DBSCAN has a complexity of $O\left(n^2\right)$. Moreover, BFS can directly search on the input matrix whereas DBSCAN needs to transform the input into coordinate sets before it starts to cluster. Above all, we considered the candidate selecting algorithm based on BFS as more suitable for our current problem.

Table 2. Breadth first search (BFS)-based and density-based spatial clustering of application with noise (DBSCAN)-based candidate selecting algorithms' AP (average precision) and the time they take to select candidates in one image.

Datasets	Algorithms	AP	Time(s)
Dataset1	BFS	0.4916	**0.0049**
	DBSCAN	**0.4964**	0.2273
Dataset2	BFS	**0.6393**	**0.0043**
	DBSCAN	0.6260	0.2264
Dataset3	BFS	**0.5804**	**0.0043**
	DBSCAN	0.5613	0.1490
Dataset4	BFS	0.5104	**0.0039**
	DBSCAN	**0.5312**	0.1791
Dataset5	BFS	0.5806	**0.0036**
	DBSCAN	**0.5958**	0.1390
Average	BFS	0.5604	**0.0041**
	DBSCAN	**0.5621**	0.1841

Moreover, we have also compared the BFS-based candidate selecting algorithm with two state-of-the-art region proposal algorithms: Selective search [14] and edge boxes [15]. For the convenience of expression, the BFS-based candidate selecting algorithm is abbreviated as TCSA (Tobacco Candidate Selecting Algorithm) in the following content.

Results on datasets with different region proposal algorithms are shown in Figure 8. Similar to what we have mentioned in Section 4.2, a PR curve trades off between precision and recall and a larger area under the PR curve means a better performance. It can be seen from the figure that TCSA proposes regions of higher quality than selective search and edge boxes.

Table 3 shows the accurate numeric results of these region proposal algorithms in different datasets, including TP, FP, FN, recall, precision, and AP.

- TCSA reaches the best recall over the five datasets. The average recall rate of our region proposed method is 18% higher than selective search and 22% higher than edge boxes.
- TCSA also gets the best AP over the datasets. The average AP of TCSA is 18% higher than selective search and 21% higher than edge boxes.
- The precision of TCSA is averagely 5% higher than selective search and 7 % better than edge boxes. However, selective search reaches better precision on some of the datasets where selective search outputs fewer predictions to guarantee its precision. Selective search's precision is guaranteed by its segmentation process and in further studies we will attempt to take that as a reference to improve the precision of TCSA.

Above all, TCSA proposes better tobacco plant regions in aerial images than selective search and edge boxes.

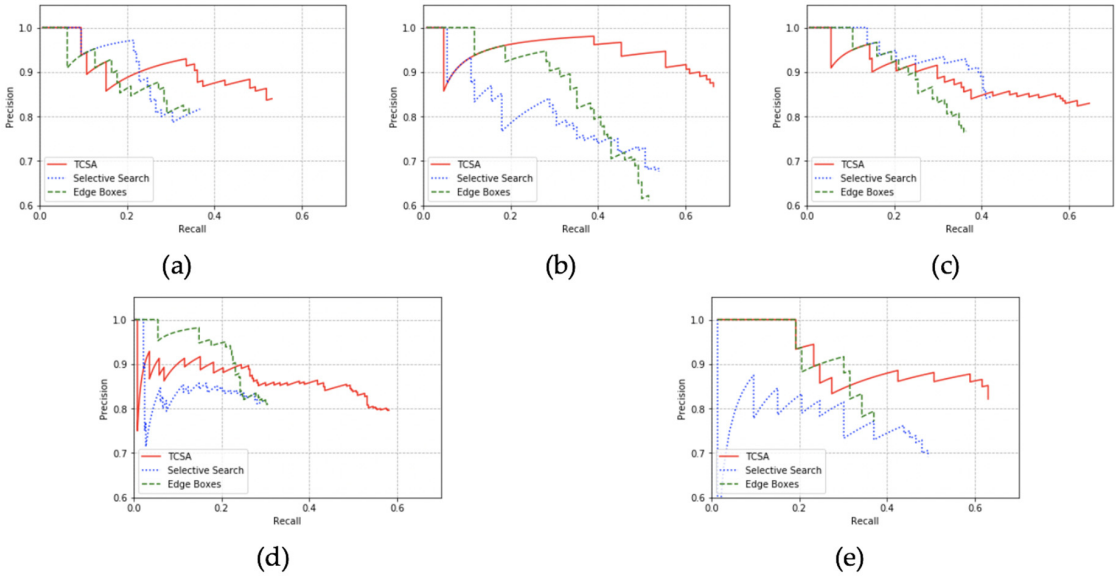

Figure 8. PR (precision-recall) curves of region proposal algorithms in different datasets. The results in datasets 1–5 are shown in (**a**–**e**). TCSA (tobacco candidate selecting algorithm) is presented as red solid lines, while selective search is presented as blue dotted lines, and edge boxes are presented as green dashed lines.

Table 3. The TP (true positive), FP (false positive), FN (false, negative), AP (average precision) of TCSA (tobacco candidate selecting algorithm), selective search and edge boxes.

Datasets	Algorithms	TP	FP	FN	Recall	Precision	AP
Dataset1	TCSA	84	16	74	**0.5316**	0.8400	**0.4916**
	Selective Search	60	11	98	0.3797	**0.8451**	0.3543
	Edge Boxes	55	12	103	0.3481	0.8209	0.3205
Dataset2	TCSA	85	13	43	**0.6641**	**0.8673**	**0.6393**
	Selective Search	71	31	57	0.5547	0.6961	0.4606
	Edge Boxes	66	42	62	0.5156	0.6111	0.4586
Dataset3	TCSA	117	24	64	**0.6464**	0.8298	**0.5804**
	Selective Search	76	14	105	0.4199	**0.8444**	0.4003
	Edge Boxes	66	20	115	0.3646	0.7674	0.3368
Dataset4	TCSA	211	54	152	**0.5813**	0.7962	**0.5104**
	Selective Search	107	24	256	0.2948	**0.8168**	0.2535
	Edge Boxes	111	26	252	0.3058	0.8102	0.2874
Dataset5	TCSA	46	10	27	**0.6301**	**0.8214**	**0.5806**
	Selective Search	36	16	37	0.4932	0.6923	0.4010
	Edge Boxes	27	8	46	0.3699	0.7714	0.3492

4.4. Evaluation on Detection Systems

We compared our detection system with faster R-CNN [19] and YOLOv3 [22] whose codes are opened on Github [33,34]. We trained a faster R-CNN with a backbone network of ResNet50 [28] which has a better performance than VGG16 [27] on ImageNet. The YOLOv3 is trained with bottleneck layers [28] in the backbone network, which sharply decreases its parameters and allows it to have faster predictions. Since these detection systems are all based on supervised learning, we use two of the datasets for their training, and the rest of the datasets for evaluation. For the convenience of expression, our detection system is abbreviated as TPD (Tobacco Plant Detector) in the following content.

Figure 9 shows the PR curves of detection systems on three testing datasets. We can see that TPD has a larger area under the PR curve than YOLOv3 and faster R-CNN which indicates that TPD has both a higher precision and higher recall on the detection of tobacco plants in aerial images.

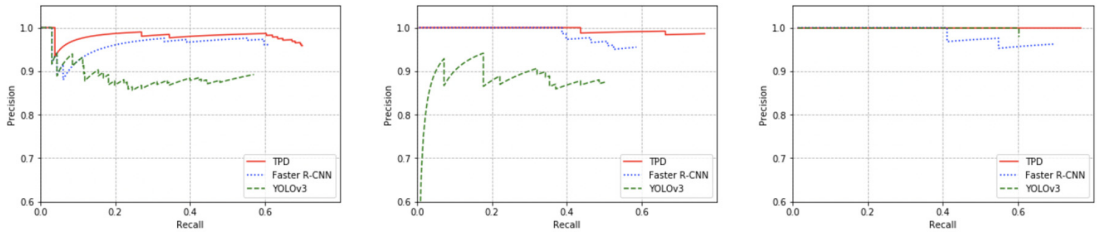

Figure 9. PR (precision-recall) curves of detection systems in three testing datasets. Evaluation results of testing datasets 3, 4, and 5 are shown in (**a**–**c**), respectively. TPD (tobacco plant detector) is presented as red solid lines, faster R-CNN (regions with CNN features) is presented as dotted blue lines, and YOLOv3 (you only look once v3) is presented as green dashed lines.

Table 4 shows the accurate numeric results of these detection systems in different testing datasets, including TP, FP, FN, recall, precision, and AP.

- TPD reaches the best recall over all of the testing datasets. On average, the recall is 12.5% higher than faster R-CNN and 21% higher than YOLOv3.
- The precision of TPD is averagely 8% higher than faster R-CNN and 7% higher than YOLOv3. However, TPD has a slightly lower precision on dataset 3 than faster R-CNN which should be caused by TCSA's lower precision on this dataset.
- TPD gets the best AP over the datasets, as well. Averagely, it is 16% higher than faster R-CNN and 23% higher than YOLOv3.

Overall, TPD has better performance in tobacco plants detection in aerial images when compared with faster R-CNN and YOLOv3.

Table 4. The TP (true positive), FP (false positive), FN (false negative), AP (average precision) of TPD (tobacco plant detector), faster R-CNN (regions with CNN features) and YOLOv3 (you only look once v3).

Datasets	Algorithms	TP	FP	FN	Recall	Precision	AP
Dataset3	TPD	254	11	109	**0.6997**	0.9585	**0.6903**
	Faster R-CNN	221	9	142	0.6088	**0.9609**	0.5945
	YOLOv3	207	25	156	0.5702	0.8922	0.5163
Dataset4	TPD	139	2	42	**0.7680**	**0.9858**	**0.7646**
	Faster R-CNN	106	5	75	0.5856	0.9550	0.5791
	YOLOv3	91	13	90	0.5028	0.8750	0.4574
Dataset5	TPD	56	0	17	**0.7671**	**1.0000**	**0.7671**
	Faster R-CNN	51	2	22	0.6986	0.9623	0.6896
	YOLOv3	44	1	29	0.6027	0.9778	0.6027

Finally, some example results in different datasets are visualized in Figure 10, where the predicted bounding boxes are outlined in yellow. We can see from the figure that TPD has an effective performance on tobacco plant detection in aerial images.

Figure 10. Visualization of some results in different datasets.

5. Conclusions

Our proposed method is mainly inspired by ideas proposed in R-CNN. As proposed in R-CNN, we select regions of interest rather than inputting the whole image into the network. We did not use the region proposal algorithms that R-CNN recommended (such as selective search [14]), because we are dealing with one specific object—the tobacco plant, which means our problem is easier. Therefore, we propose a more targeted region selecting algorithm in the first place. R-CNN uses the convolutional neural network as a feature extractor, we use it as a classifier. Experiments on different datasets have demonstrated that our proposed algorithm performs well on tobacco plant detection in aerial images generally. However, this proposed algorithm still needs to improve its precision especially in the region of proposal stage. In future studies, we will work to improve the candidate selecting algorithm in the hope that our proposed method can play a role in the counting stage of the tobacco plants' yield estimation and thereby contribute to the agriculture precision.

Author Contributions: Methodology, J.P.; data curation Y.S.; software, J.P.; validation, H.K., X.S., and J.P.; formal analysis, H.K.; investigation, Y.S.; resources, Y.S.; writing—original draft preparation, J.P.; writing—review and editing, J.P. and H.K.; visualization, J.P.; supervision, X.S.; project administration, X.S.; funding acquisition, Y.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded by the Open Foundation of Key Laboratory of Software Engineering of Yunnan Province Grant No. 2015SE204 and the National Natural Science Foundation of China Grant No. 61663046 and No. 61876166.

Acknowledgments: The dataset is provided by Yunnan TianYi Inc. Thanks to the opened python libraries; they greatly facilitate the implementation.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Musk, A.W.; De Klerk, N.H. History of tobacco and health. *Respirology* **2003**, *8*, 286–290. [CrossRef] [PubMed]
2. Fiore, M.C.; Bailey, W.C.; Cohen, S.J.; Dorfman, S.F.; Goldstein, M.G.; Gritz, E.R.; Heyman, R.B.; Jaen, C.R.; Kottke, T.E.; Lando, H.A.; et al. *Treating Tobacco Use and Dependence: Clinical Practice Guideline Respiratory Care*; U.S. Department of Health and Human Services: Irving, TX, USA, 2008.
3. Stein, M.; Bargoti, S.; Underwood, J. Image based mango fruit detection, localisation and yield estimation using multiple view geometry. *Sensors* **2016**, *16*, 1915. [CrossRef] [PubMed]
4. Bargoti, S.; Underwood, J.P. Image segmentation for fruit detection and yield estimation in apple orchards. *J. Field Robot.* **2017**, *34*, 1039–1060. [CrossRef]
5. Ferencz, C.; Bognar, P.; Lichtenberger, J.; Hamar, D.; Tarcsai, G.; Timár, G.; Molnár, G.; Pásztor, S.; Steinbach, P.; Székely, B.; et al. Crop yield estimation by satellite remote sensing. *Int. J. Remote Sens.* **2004**, *25*, 4113–4149. [CrossRef]
6. Prasad, A.K.; Chai, L.; Singh, R.P.; Kafatos, M. Crop yield estimation model for Iowa using remote sensing and surface parameters. *Int. J. Appl. Earth Obs. Geoinf.* **2006**, *8*, 26–33. [CrossRef]
7. Fan, Z.; Lu, J.; Gong, M.; Xie, H.; Goodman, E.D. Automatic tobacco plant detection in UAV images via deep neural networks. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2018**, *11*, 876–887. [CrossRef]
8. Felzenszwalb, P.F.; Girshick, R.B.; McAllester, D.; Ramanan, D. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *32*, 1627–1645. [CrossRef]
9. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–25 June 2005.
10. Suykens, J.A.; Vandewalle, J. Least squares support vector machine classifiers. *Neural Process. Lett.* **1999**, *9*, 293–300. [CrossRef]
11. Fares, M.A.; Elena, S.F.; Ortiz, J.; Moya, A.; Barrio, E. A sliding window-based method to detect selective constraints in protein-coding genes and its application to RNA viruses. *J. Mol. Evol.* **2002**, *55*, 509–521. [CrossRef] [PubMed]
12. Yang, L.; Yang, G.; Yin, Y.; Xiao, R. Sliding window-based region of interest extraction for finger vein images. *Sensors* **2013**, *13*, 3799–3815. [CrossRef] [PubMed]
13. Alexe, B.; Deselaers, T.; Ferrari, V. Measuring the objectness of image windows. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 2189–2202. [CrossRef] [PubMed]
14. Uijlings, J.R.; Van De Sande, K.E.; Gevers, T.; Smeulders, A.W. Selective search for object recognition. *Int. J. Comput. Vis.* **2013**, *104*, 154–171. [CrossRef]
15. Zitnick, C.L.; Dollár, P. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 391–405.
16. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.
17. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
18. Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
19. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 91–99.

20. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2961–2969.
21. 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, Las Vegas, NV, USA, 26–30 June 2016; pp. 779–788.
22. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
23. Shafiee, M.J.; Chywl, B.; Li, F.; Wong, A. Fast YOLO: A fast you only look once system for real-time embedded object detection in video. *arxiv* **2017**, arXiv:1709.05943. [CrossRef]
24. Zhong, J.; Lei, T.; Yao, G. Robust vehicle detection in aerial images based on cascaded convolutional neural networks. *Sensors* **2017**, *17*, 2720. [CrossRef] [PubMed]
25. Beamer, S.; Asanovic, K.; Patterson, D. Direction-optimizing breadth-first search. *Sci. Program.* **2013**, *21*, 137–148. [CrossRef]
26. Arlia, D.; Coppola, M. Experiments in parallel clustering with DBSCAN. In *Lecture Notes in Computer Science, Proceedings of the European Conference on Parallel Processing, Manchester, UK, 28–31 August 2001*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 326–331.
27. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
28. 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, Las Vegas, NV, USA, 26–30 June 2016; pp. 770–778.
29. Scherer, D.; Müller, A.; Behnke, S. Evaluation of pooling operations in convolutional architectures for object recognition. In *Lecture Notes in Computer Science, Proceedings of the International Conference on Artificial Neural Networks, Thessaloniki, Greece, 15–18 September 2010*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 92–101.
30. Jang, E.; Gu, S.; Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv* **2016**, arXiv:1611.01144.
31. LeCun, Y.; Jackel, L.; Bottou, L.; Brunot, A.; Cortes, C.; Denker, J.; Drucker, H.; Guyon, I.; Muller, U.; Sackinger, E.; et al. Comparison of learning algorithms for handwritten digit recognition. In Proceedings of the International Conference on Artificial Neural Networks, Perth, Australia, 27–30 November 1995; Volume 60, pp. 53–60.
32. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, 21–24 June 2010; pp. 807–814.
33. Faster R-CNN Implemented with Python. Available online: https://github.com/rbgirshick/py-faster-rcnn (accessed on 5 February 2015).
34. Yolov3 a Keras implementation of YOLOv3 (Tensorflow Backend). Available online: https://github.com/qqwweee/keras-yolo3 (accessed on 3 April 2018).
35. Davis, J.; Goadrich, M. The relationship between Precision-Recall and ROC curves. In Proceedings of the 23rd International Conference on Machine Learning. ACM, Pittsburgh, PA, USA, 25–29 June 2006; pp. 233–240.