Computers, Materials & Continua
DOI:10.32604/cmc.2021.017089
Article

Tech Science Press

Small Object Detection via Precise Region-Based Fully Convolutional Networks

Dengyong Zhang[1,2], Jiawei Hu[1,2], Feng Li[1,2,*], Xiangling Ding[3], Arun Kumar Sangaiah[4] and Victor S. Sheng[5]

[1]Hunan Provincial Key Laboratory of Intelligent Processing of Big Data on Transportation, Changsha University of Science and Technology, Changsha, 410114, China
[2]School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha, 410114, China
[3]School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, 411004, China
[4]School of Computing Science and Engineering, Vellore Institute of Technology (VIT), Vellore, 632014, India
[5]Department of Computer Science, Texas Tech University, Lubbock, 79409, TX, USA
[*]Corresponding Author: Feng Li. Email: lif@csust.edu.cn
Received: 20 January 2021; Accepted: 29 March 2021

Abstract: In the past several years, remarkable achievements have been made in the field of object detection. Although performance is generally improving, the accuracy of small object detection remains low compared with that of large object detection. In addition, localization misalignment issues are common for small objects, as seen in GoogLeNets and residual networks (ResNets). To address this problem, we propose an improved region-based fully convolutional network (R-FCN). The presented technique improves detection accuracy and eliminates localization misalignment by replacing position-sensitive region of interest (PS-RoI) pooling with position-sensitive precise region of interest (PS-Pr-RoI) pooling, which avoids coordinate quantization and directly calculates two-order integrals for position-sensitive score maps, thus preventing a loss of spatial precision. A validation experiment was conducted in which the Microsoft common objects in context (MS COCO) training dataset was oversampled. Results showed an accuracy improvement of 3.7% for object detection tasks and an increase of 6.0% for small objects.

Keywords: Small object detection; precise R-FCN; PS-Pr-RoI pooling; two-stage detector

1 Introduction

Deep learning (DL) has become an active topic of research in the field of artificial intelligence, as it offers significant advantages in the domains of object detection, automated speech recognition, and natural language processing [1–3]. Convolutional neural networks (CNNs) are an important DL technique commonly used for object detection and related tasks [4]. AlexNet [5], released in 2012, has generated significant interest from academia. Girshick et al. [6] applied this model to object detection because of its robust ability to learn advanced image feature representations. Similarly, region-CNNs (R-CNNs) have been applied to the PASCAL VOC 2007 dataset, yielding significant increases in detection accuracy. In a recent study, the deformable part-based model (DPM v5) [7] improved the mean average precision (mAP) from 33.7% to 58.5% [8]. SPPnet [9] introduces a spatial pyramid pooling (SPP) layer that produces a fixed-size output from a feature map input of any size, without scaling the original image [8]. SPPnet is more than 20 times faster than R-CNN yet suffers no loss of detection accuracy [8]. Faster R-CNNs integrate several components, including feature extraction and region proposals, into a unified end-to-end framework [8]. Although object detection accuracy has increased with each of these developments, the detection and alignment accuracy for smaller objects remains much lower than that of larger objects. For example, in a study by Ren et al. [10] the mAP was 6.6% for small objects and 45.0% for large objects (using a Faster R-CNN).

Object detection is one of the most fundamental tasks in computer vision [11], playing a vital role in subsequent problems such as facial recognition [12], gait recognition, crowd counting, and instance segmentation. Among these, small object detection is highly challenging but essential for a variety of practical applications, such as detecting distant objects in aerial images [13,14]. In addition, data from self-driving cars often include objects like traffic lights and pedestrians that appear relatively small in each image. As such, more attention should be devoted to the detection of such objects [15].

Small objects of interest can be defined in two ways based on their relative size (an area smaller than 12% of the original image) or absolute size (a rectangular area less than 32×32). The absolute size was used to define small objects in the MS COCO dataset [16], commonly used for challenging problems in deep learning. Most early algorithms were general object detectors [8], like the traditional one-stage detector, You Only Look Once (YOLO) [17] models, the single-shot multi-box detector (SSD) [18], and two-stage detectors like GoogLeNet [19] and Faster R-CNN [10], none of which are highly accurate for small objects. Recent efforts to improve small object detection accuracy have included data augmentation, feature fusion, and context information integration. These are summarized in detail below.

Data augmentation: Training sample quantities can be increased by oversampling images containing small objects or by increasing the number of small objects in each training image [15]. The first approach is the least expensive and directly addresses any imbalance of small and large objects. In addition, the number of anchor-matching small objects in a feature map correspondingly increases with the number of small objects in each training sample [15]. This approach contributes to loss calculations in region proposal network (RPN) training [15] and thereby improves detection accuracy.

Feature fusion: Various feature map layers correspond to different receptive field sizes that express specific degrees of information extraction, with shallow layers being more suitable for detecting smaller objects [20]. However, when these features are used to detect larger objects, they are often only partly visible, indicating insufficient information. Conversely, feature maps in deeper layers are suitable for detecting larger objects. However, if these maps are used to detect smaller objects, excessive background information and redundant features are often introduced, which is suboptimal. Feature pyramid networks (FPNs) [20] can be used to fuse feature maps for different layers, to improve the accuracy of object detection. However, this approach increases computational overhead and does not address the problem of data imbalances between small and large objects in a training database [15].

Context information integration: Smaller objects, particularly faces, typically do not appear alone in images. As such, algorithms like Pyramidbox [21] integrate context information from nearby structures (e.g., heads and shoulders). This increases the size of small objects and makes them easier to detect. However, not all small objects exhibit contextual information.

This paper primarily focuses on two key aspects of object detection. The first is a PS-Pr-RoI pooling layer that represents relative spatial positioning and aligns coordinates, effectively improving detection accuracy and efficiency for small objects. The second step involves oversampling the training image with small objects and then training the detector on the augmented data. This approach achieved high detection accuracy for the MS COCO dataset. The proposed PS-Pr-RoI pooling layer can also be easily embedded in other detection frameworks.

The remainder of this paper is organized as follows. Section 2 presents related work. Section 3 outlines the proposed methodology, including the overall detection framework and specific implementation details. Section 4 discusses the validation experiment and analyzes corresponding results. Section 5 provides conclusions and considers avenues for future work.

2 Related Work

There have been two key periods in object detection history: the use of traditional models and the application of deep learning [8]. Most conventional algorithms rely on complex handcrafted features [8]. In addition, limitations in computing resources have required the development of acceleration protocols to speed up the classification process. Examples of conventional object detection algorithms include feature descriptors based on the histogram of oriented gradients (HOG) [22] and DPM, an extension of HOG [7]. Dala et al. used HOG to make significant improvements to scale-invariant feature transformations [23,24] and shape contexts [25]. HOG has also become an important foundation for several subsequent object detectors [26–28] and various computer vision applications. DPM, the peak of traditional object detection [6], can be used to decompose a complex detection problem into several simpler problems.

As the technology developed further, deep learning-based object detection became more accurate than traditional methods [29,30]. Common DL detection algorithms include R-CNN [6], SPPnet [9], Faster R-CNN [10], YOLO [17], and SSD [18]. R-CNNs [6] were proposed in 2014 and are used to extract a series of object proposals (candidate boxes) from an input image using a selective search algorithm [31], prior to adjusting each proposal to match a fixed size. A CNN is then applied to extract image features and linear support vector machine (SVM) classifiers are used to identify objects in each candidate box. While R-CNNs can improve detection accuracy, calculating convolutional features in each region is time consuming. SPPnet [9] was later proposed for generating fixed-size representations, only once extracting convolutional features from the original image. This approach substantially reduced calculations and is ∼20 times faster than an R-CNN, without sacrificing detection accuracy. However, SPPnet still suffers from certain limitations as it only fine tunes the fully connected layer. In addition, the detector is not an end-to-end deep learning object detector. Faster R-CNNs [10] utilize RPN, the first end-to-end deep learning object detector, but their speed is insufficient for many real-time applications. YOLO [17] transforms object detection into a regression problem, which speeds up the classification process but suffers from low detection accuracy. Liu et al. [18] proposed the SSD detector in 2015, which exhibits faster detection speed and higher accuracy than YOLO.

Existing DL detectors can be classified as either one-stage or two-stage processes. One-stage models, such as YOLO and SSD [17], directly transform object detection into a regression problem. Two-stage models, such as R-CNN [6], SPPnet [9], and Faster R-CNN [10], generate a set of region proposals for object detection. One-stage detectors are generally faster but less accurate, while two-stage detectors are typically more accurate but slower. The first stage of two-stage detection outputs a set of rectangular proposals with an object score [10], thereby improving sample set balance. Class specific detection is then implemented in the second stage. This approach is simpler than one-stage detection because, like DPM, it divides a complex problem into smaller, simpler problems.

As discussed above, despite the recent progress made in object detection, practical issues remain due to low image resolution, reduced information availability, and increased noise associated with small objects. After several years of development, some solutions have emerged to improve small object detection performance, including the use of smaller and denser anchors, improved anchor-matching strategies, and generative adversarial networks (GANs) [32].

3 Proposed Method

The primary aim of this paper is to improve the accuracy of small object detection and eliminate localization misalignment. As such, a two-stage object detector (R-FCN) was selected for the study [33]. An R-FCN was used as the basis architecture to facilitate the use of a new PS-RoI pooling module. This pooling operation includes clear relative spatial information and thus effectively improves the sensitivity of object positioning in deeper neural networks [33], which is beneficial for small object detection. In addition, it also pools all RoIs simultaneously across all feature maps, thereby increasing processing speeds.

As shown in Fig. 1, the employed convolutional neural network (CNN) is based on a residual net-101 (ResNet-101) [34], in which other architectures can also be used. The final convolutional layer produces $k^2 \times (C+1)$ position-sensitive score maps and the RPN then produces RoIs [35] for application to these score maps. In this study, PS-RoI pooling [33] was replaced with PS-Pr-RoI pooling, which avoids any coordinate quantization. In the final step, voting is conducted for the PS-Pr-RoI pooling output, produced from $k^2 \times (C+1)$ position-sensitive score maps. This process is discussed in detail below.

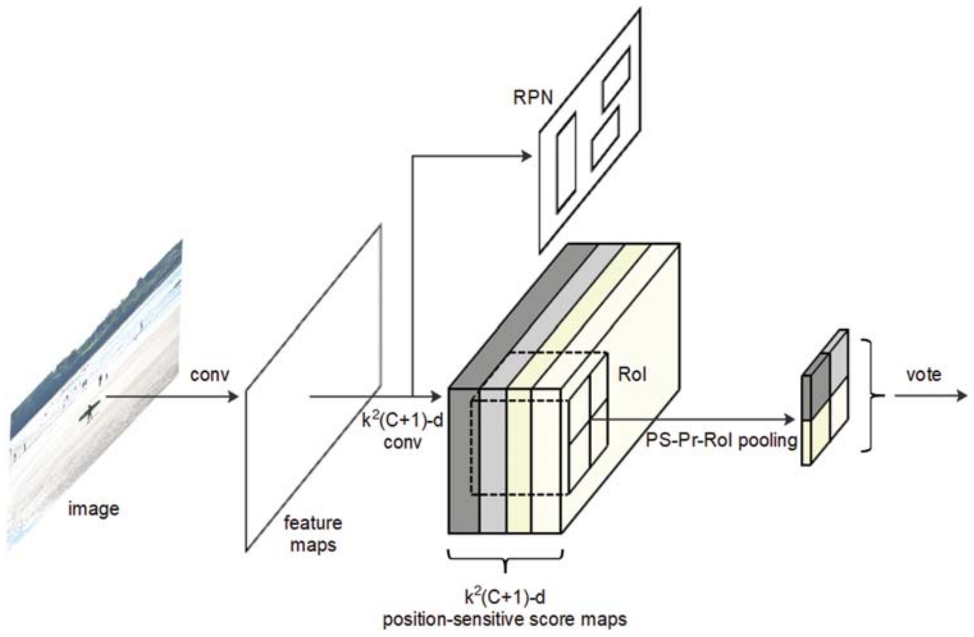

Figure 1: The overall precise R-FCN architecture

3.1 Precise R-FCN

The overall architecture for the presented detector can be divided into four parts: a basic convolutional neural network (ResNet-101) [34], an RPN [10], a position-sensitive score map layer (the last convolutional layer), and a PS-Pr-RoI pooling/voting layer [33].

3.1.1 Residual Networks

Residual networks are powerful tools that make training deeper networks possible. Prior to the development of residual networks, two potential problems arose as the number of network layers increased: decreasing accuracy and vanishing or exploding gradients [34]. Residual networks were developed to address these problems as their internal building blocks use shortcut connections that alleviate issues caused by increasing neural network depth. Each building block can be divided into two parts: shortcut connections and residual mappings. The building block formula is given by $x_{l+1} = f(y_1)$, where $y_l = h(x_l) + F(x_l, W_l)$ and $f(\cdot)$ is an activation function that generally uses a rectified linear unit (ReLU). The term $h(x_l)$ describes shortcut connections, as shown in the lower part of Fig. 2, and $F(x_l, W_l)$ is a residual mapping that is generally composed of two or three convolution operations (displayed in the upper part of Fig. 2). In convolutional neural networks, the dimensions of x_l may differ from that of x_{l+1}, requiring the use of a 1×1 convolution to increase or reduce dimensionality. In this case, $h(x_l) = W_l'x$, where $W_l'x$ is a 1×1 convolution operation. When $x_{l+1} = x_l + F(x_l, W_l)$, the residual network exhibits several common attributes, including an L layer that can be expressed as the sum of any shallower layers and a corresponding residual component, which prevents vanishing gradients in the residual network. Finally, the gradient of the L layer can be transferred to that of any layer shallower than itself.

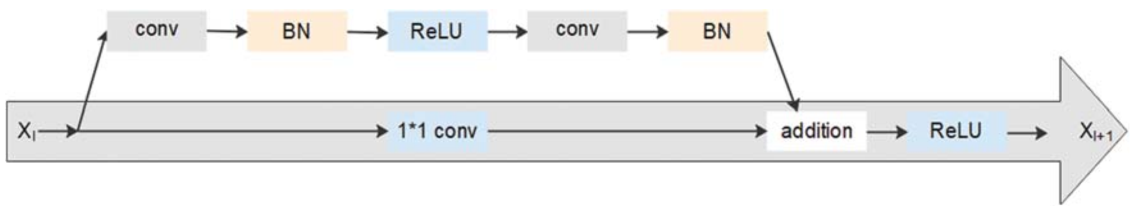

Figure 2: The building block. Here, the term addition refers to the unit addition operation and BN refers to the batch normalization

The residual network used in this study is a modified version of ResNet-101, developed by the authors of the R-FCN [33]. It included 100 convolutional layers, followed by a global average pooling layer, and a 1000-class fully connected layer [33]. After preserving 100 convolutional layers, we removed the global average pooling and fully connected layers, only using the convolutional layer to calculate feature maps [33]. We subsequently added a randomly initialized 1024-dimensional 1×1 convolutional layer to reduce the dimensionality and further utilize convolutional layers in the $k^2 \times (C+1)$ channel, to generate position-sensitive score maps [33].

3.1.2 Region Proposal Networks

RPNs can be used to generate RoIs for object detection. In a fast R-CNN [35], the entire image is sent to the CNN for the purpose of extracting feature maps. RoIs are then generated using an offline selection search algorithm. However, this process is time consuming and does not provide a clear methodology for generating RoIs. In contrast, RPNs are both efficient and concise as they incorporate RoI generation in end-to-end learning. Feature maps from the residual network are used as RPN input, while a set of rectangular proposals with an object score are produced as output [10]. As illustrated in Fig. 3, the RPN incorporates several key steps referred to as anchor generation, anchor object generation, RPN loss, and proposal generation.

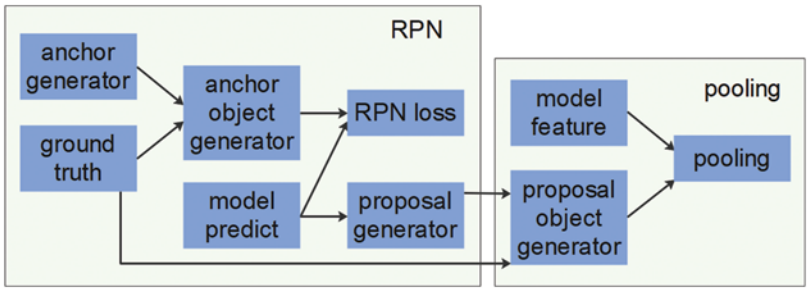

Figure 3: The region proposal network used to generate RoIs in the feature maps

Anchor generation: Anchors are preset using shared scale and ratio parameters. When an image of size 480×400 is input, using $stride = 16$, the size of the feature map will be 30×25. The anchor scale is then (4, 8, 16) and the parameter ratio is (0.5, 1, 2). As a result, nine anchors are produced.

Anchor object generation: The anchor object layer differentiates positive and negative anchor samples. In this study, the intersection-over-union (IoU) was calculated between the anchor and the ground truth using three criteria. 1) For each ground truth, the anchor exhibiting the largest IoU overlap with a ground-truth box was selected as a positive sample. 2) Anchors exhibiting an IoU overlap higher than a certain threshold (T1) with any ground truth box were selected as positive samples. 3) If the IoU overlap with the ground truth box was less than a certain threshold (T2), the anchor was considered to be a negative sample. Anchors with an IoU value between T2 and T1 were labeled "do not care" samples that were not involved in RPN loss calculation or model optimization. These steps ensured that each ground truth was assigned a corresponding anchor and that a certain number of anchors could be selected to balance positive and negative samples.

RPN loss: Since RPNs include both classification and regression tasks, the corresponding loss term is comprised of both classification and regression loss. Cross-entropy loss was used for classification tasks while L1-smooth loss was used for regression tasks. These loss terms can be represented as:

$$L\left(s, t_{x,y,w,h}\right) = L_{cls}\left(s_{c^*}\right) + \lambda\left[c^* > 0\right] L_{reg}\left(t, t^*\right) \tag{1}$$

$$L_{cls}\left(s_{c^*}\right) = -\log\left(s_{c^*}\right) \tag{2}$$

$$L_{reg}\left(t, t^*\right) = \sum_{i \in \{x,y,w,h\}} smooth_{L1}\left(t_i - t_i^*\right) \tag{3}$$

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & if\ \ |x| < 1 \\ |x| - 0.5 & otherwise \end{cases} \tag{4}$$

where c^* is an RoI ground truth label and t^* represents the ground truth box.

Proposal generation: This step involves sorting by object score and inputting the first N1 terms to a non-maximum suppression (NMS) algorithm. The first N2 are then taken as the final output result after NMS is completed.

Fully convolutional networks are suitable for image classification applications due to their strong feature extraction capabilities. However, these networks focus solely on features and do not consider relative spatial positioning, making them unsuitable for object detection. In contrast, position-sensitive score maps can represent relative object locations. In this study, PS-Pr-RoI pooling was used to generate output of the same size from RoIs of different sizes, thereby producing a more accurate relative spatial position than that of RoI pooling [36]. This process does not quantize coordinates and is thus highly effective for eliminating localization misalignment and detecting small objects.

This approach can be described as follows. First, the original image is sent to a ResNet-101 [33] for feature map generation. An RPN and a convolutional layer are then used to acquire the $k^2 \times (C+1)$ and $4k^2$ position-sensitive score maps from the feature maps. PS-Pr-RoI pooling and voting operations are then performed on the maps to obtain classification and position information. The proposed small object detection algorithm based on a precise R-FCN is shown in Algorithm 1. Specific pooling and voting operations are described in detail as follows.

3.2 PS-Pr-RoI Pooling

In conventional RoI pooling, two quantization operations are performed on the RoI boundaries and bins, resulting in low position accuracy. RoI features are then updated without a gradient and thus cannot be adjusted during training. An RoI alignment step [37] can be used to eliminate quantization errors by canceling two quantization operations and using $x/16$ rather than $[x/16]$. This means that no quantization is performed on any coordinates involved in the RoI, its bins, or the sampling points [37]. The N sampling points within each bin were updated using bilinear interpolation with gradient conduction [37]. This produced a new parameter N, which is preset and cannot be adaptively adjusted according to the size of the bins. The gradient of these points is related to the features in the four integer positions surrounding them. Not all features in the bins possess a gradient.

Algorithm 1: Precise R-FCN-based small object detection algorithm

```
Input:
  An image
Output:
  Classification and position information
01: Convolve using a ResNet-101 to acquire feature map 1
02: Branch 1: Acquire RoIs by applying an RPN to feature map 1
03: Branch 2: Convolve feature map 1 to acquire $k^2 \times (C+1)$ dimensional position-sensitive score
    maps
04:   PS-Pr-RoI pooling to obtain feature map 2
05:   Voting on feature map 2
06:   Classification
07: Branch 3: Convolve feature map 1 to acquire $4k^2$ dimensional position-sensitive score maps
08:   PS-Pr-RoI pooling to obtain feature map 3
09:   Voting on feature map 3
10:   Position information
```

Precise RoI pooling does not involve any quantization operations and solves problems associated with the number of sampling points by eliminating it as a system parameter [36]. In this study, all feature maps were made continuous through interpolation and the pooling operation was calculated using a two-order integral, to ensure all features exhibited gradient transfer. PS-Pr-RoI pooling, which includes clear relative spatial positioning, effectively improved the position sensitivity of deeper convolutional layers. In addition, it does not perform quantization operations on any coordinates, which eliminates localization misalignment. This approach improved detection accuracy for small objects.

Position-sensitive score maps: The position-sensitive score maps produced in the previous step contain clear relative spatial information, generated by the last convolutional layer. These maps include $k \times k$ colors ($k = 2$ here for ease of explanation, $k = 7$ in practical applications) that represent the probability of an object existing at specific locations. For instance, the gray map represents the probability of objects being present in the top-left position, while the light yellow map represents the probability of an object in the bottom-right position. Maps of each color include $C + 1$ categories, including C object categories and a background category, for a total of $k^2 \times (C + 1)$ points.

PS-Pr-RoI pooling: Each RoI was divided into $k \times k$ regular bins produced via PS-Pr-RoI pooling, regardless of its size. This pooling layer performed selective pooling, with each bin response collected from one of the $k \times k$ position-sensitive score maps [33]. The features in these maps were initially discretized and were made continuous at any coordinate (x, y) using bilinear interpolation, in order to perform PS-Pr-RoI pooling [36]. The score maps were also continuous as a result of the continuous features, which were calculated at any given coordinate (x, y) using:

$$f(x, y) = \sum_{i,j} IC(x, y, i, j) \times w_{i,j} \tag{5}$$

Here, IC is an interpolation coefficient given by $IC(x, y, i, j) = \max(0, 1 - |x - i|) \times \max(0, 1 - |y - i|)$ and the feature $w_{i,j}$ is at a discrete position (i, j) in the map. Pooling was performed using:

$$PS\Pr Pool_c(bin, F) = \frac{\int_{y_1}^{y_2} \int_{x_1}^{x_2} f(x, y)\, dxdy}{(x_2 - x_1)(y_2 - y_1)} \tag{6}$$

It is evident this function works to average all continuous features, yielding the pooled response in the $(a, b)^{th}$ bin for the cth category $(0 \le a, b \le k - 1)$. Here, (x_1, y_1) is the top-left region of the bin, (x_2, y_2) is the bottom-right region of the bin, and the input F is one of the $k^2 \times (C + 1)$ score maps. The *PSPrPool* function is continuously differentiable and its partial derivative x_1 can be calculated using:

$$\frac{\partial PS\Pr Pool_c(bin, F)}{\partial x_1} = \frac{PS\Pr Pool_c(bin, F)}{x_2 - x_1} - \frac{\int_{y_1}^{y_2} f(x_1, y)\, dy}{(x_2 - x_1)(y_2 - y_1)} \tag{7}$$

The input of the PS-Pr-RoI pooling is $k^2 \times (C + 1)$ position-sensitive score maps. Following PS-Pr-RoI pooling, the position-sensitive score maps for different colors are formed into new $k \times k$ bins. For example, in Fig. 4, the gray map only occupies a bin at the top-left position, while the light yellow map is assigned to only the bin at the bottom-right position. Once all bins have been recombined, they take the form of a thin solid block shown on the right side of Fig. 4. This block forms the output of the PS-Pr-RoI pooling operation and has a dimension of $C+1$.

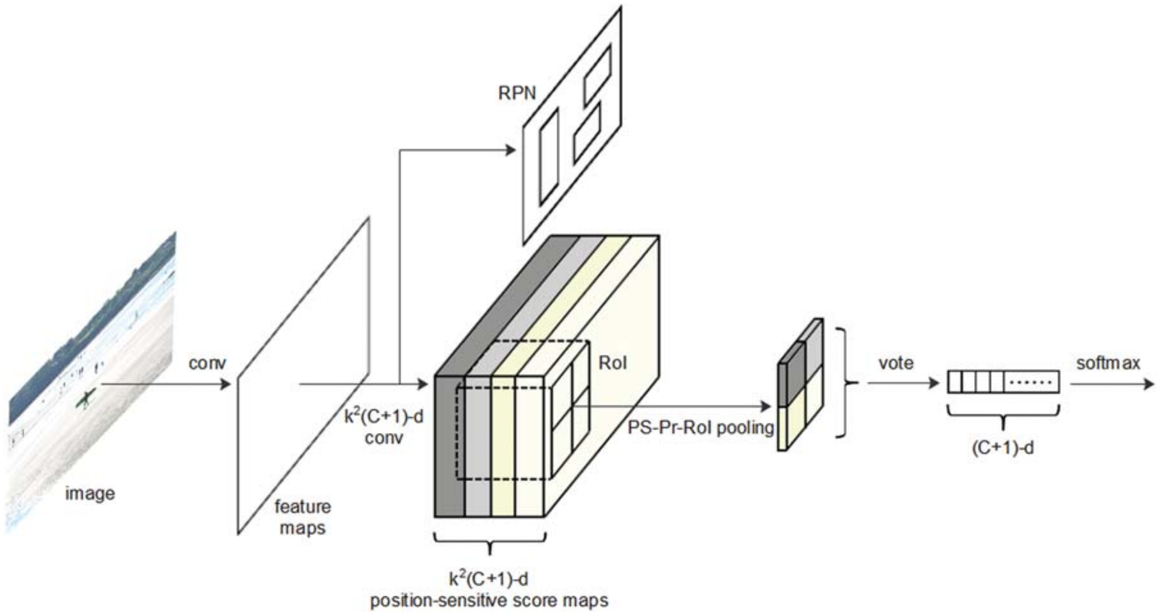

Figure 4: Implementation of PS-Pr-RoI pooling and voting operations

Voting: The output of the pooling step is used for voting. The $k \times k$ position-sensitive scores for each class are averaged directly [33] to produce a $C+1$-dimensional vector, given by:

$$\mathrm{r_c}(\theta) = \sum_{\mathrm{bin}} \mathrm{r_c}(bin, \theta) \tag{8}$$

The position-sensitive object score for the cth category bin can be expressed as [33]:

$$r_c(bin, \theta) = PS\,\mathrm{Pr}\,Pool_c(bin, F) = \frac{\int_{y_1}^{y_2}\int_{x_1}^{x_2} f(x, y)\,dxdy}{(x_2 - x_1)(y_2 - y_1)} \tag{9}$$

The $C+1$ softmax responses were calculated across categories and can be represented as follows:

$$\mathrm{s_c}(\theta) = \mathrm{e}^{r_c(\theta)} / \sum_{c'=0}^{C} \mathrm{e}^{r_{c'}(\theta)} \tag{10}$$

This term was used to calculate the cross-entropy loss during training and to rank RoIs during inference [33]. The PS-Pr-RoI pooling and voting steps are detailed in Algorithm 2.

Algorithm 2: Implementation of PS-Pr-RoI pooling and voting algorithm

```
Input:
  $k^2 \times (C+1)$-dimensional position-sensitive score maps
Output:
  Classification confidence
01: Continuous position-sensitive score maps
02: RoIs applied to score maps
03: PSPrPool
04: $r_c\,(\text{bin}, \theta)$
05: $\mathrm{r_c}\,(\theta)$
06: $\mathrm{s_c}\,(\theta) = \mathrm{e}^{r_c(\theta)} / \sum_{c'=0}^{C} \mathrm{e}^{r_{c'}(\theta)}$
07: Classification confidence
```

Bounding box regression was defined in a similar way using a $4k^2$-dimensional convolutional layer. For each RoI, a $4k^2$-dimensional vector, used to parameterize a bounding box, was generated and then aggregated into a four-dimensional vector by average voting.

4 Experimental Results and Analysis

Detection performance was evaluated experimentally for small objects using the mean average precision (mAP) metric, as shown in Fig. 5.

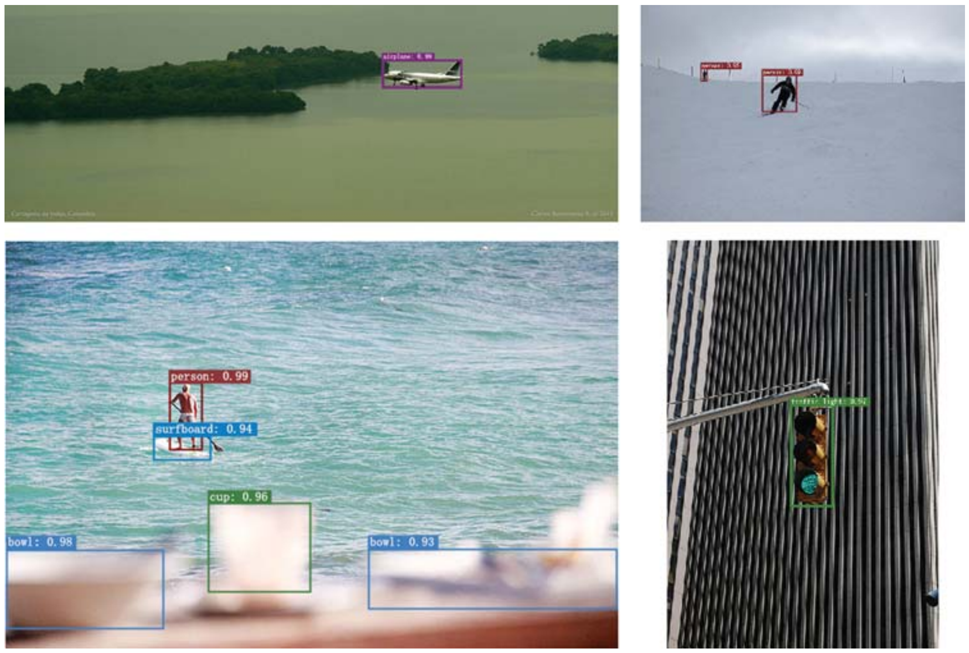

Figure 5: Precise R-FCN results for the COCO test-dev datasets. These results are based on a ResNet-101, achieving an AP of 32.9. The bounding box, classification, and confidence values are also included

4.1 Experimental Environment

The validation experiment included a GPU utilizing an Nvidia GeForce GTX TITAN X. The convolutional neural network environment and configuration were as follows. All detectors in the experiment were run on Ubuntu 16.04 using the TensorFlow deep learning framework and CUDA (version 10.1) programmed with Python 3.7.

4.2 Implementation Details

The accuracy of the detector was evaluated using MS COCO datasets containing a total of 80 objects. By default, we used a single-scale training and testing step, following a process similar to that of Kisantal et al. [15] for processing the training data. Multiple image copies were created that contained small objects (using a preset sampling ratio of 3), rather than performing a random oversample [15]. ResNet-101 was used as the basic convolutional neural network, pre-trained with ImageNet [38]. The ResNet's effective stride was reduced from 32 to 16 pixels to increase the resolution of the position-sensitive score maps and improve the accuracy of small object detection. All layers before and at the conv4 stage remained unchanged and the RPN was computed on top of the conv4 stage [33]. The RPN was used to identify RoIs, defined as positive samples with an IoU overlap of at least 0.5 with a ground truth box. Other samples were deemed negative. N3 proposals were calculated for each image during the forward pass and the corresponding loss was also evaluated. In addition, the 128 sample RoIs with the highest loss were selected for backpropagation [33]. A total of 300 RoIs were used for testing [10] and were post-processed by NMS using a threshold of 0.3 IoU. The loss functions defined for each RoI are shown in Eqs. (1)–(4). The balance weight was set to $\lambda = 1$ and $c^* > 0$ requires RoIs to be in the foreground. PS-Pr-RoI pooling was then applied with a momentum of 0.9 and a weight decay of 0.0005. The stochastic gradient descent (SGD) [39] method was implemented to identify optimal solutions by training the loss function.

4.3 Results and Analysis

The proposed detector is simple and requires no additional features, such as multi-scale training or testing. In addition, with the aid of PS-Pr-RoI pooling, our detector can increase object detection accuracy beyond that of some conventional algorithms. Tab. 1 shows average precision (AP) values for $IoU = 0.5 : 0.05 : 0.95$. The AP_S, AP_M, and AP_L terms represent detector AP measurements for small, medium, and large objects, respectively. As seen in the table, the proposed detector achieved substantial improvements in performance compared with Faster R-CNN and R-FCN, as measured by AP_S, AP_M, and AP_L. This is primarily due to the inclusion of a novel PS-Pr-RoI pooling module that eliminates localization misalignment, by not involving any quantization of coordinates, and the use of features exhibiting gradient transfer. This PS-Pr-RoI pooling can effectively improve position sensitivity for deeper convolutional layers as it includes a relative spatial position. Notably, deformable convolutional networks [40] outperformed our detector as measured by AP, through the use of deformable convolutions, deformable PS-RoI pooling, and a multi-scale training strategy. However, our approach produced a superior AP_S value, demonstrating that PS-Pr-RoI pooling can improve the detection of small objects.

Ablation experiments: In this study, PS-Pr-RoI pooling is proposed as a replacement for PS-RoI pooling in an R-FCN. Our approach was found to not only improve AP but also to significantly improve AP_S for the processed training datasets. This is a useful way of solving the problem of large and small object imbalances in training datasets such as MS COCO [15]. Tab. 2 displays the accuracy achieved by the same detector for various training data. Experimental results showed this strategy improved the AP_S index but decreased the AP_L index. One possible explanation is that the training datasets were preprocessed, adding a large number of images containing small objects. This increased the contribution of small objects to computational loss, which biased the overall network towards small object detection and thus resulted in a decrease in AP_L.

Table 1: Detection accuracy for the proposed detector compared with some conventional detectors

Detector	Basis architecture	Pooling	Training data	Test data	AP	AP_S	AP_M	AP_L
Faster R-CNN	ResNet-101	RoI pooling	Train	Val	27.2	6.6	28.6	45.0
R-FCN	ResNet-101	PS-RoI pooling	Trainval	Test-dev	29.2	10.3	32.4	43.3
Deformable convolutional networks	ResNet-101 + deformable convolution	Deformable PS-RoI pooling	Multi-sc trainval	Test-dev	34.5	14.0	37.7	50.3
Precise R-FCN	ResNet-101	PS-Pr-RoI pooling	Trainval	Test-dev	32.9	16.3	36.2	44.8

Table 2: The results of ablation experiments. Detectors were trained on the MS COCO trainval dataset and detection results are provided for the MS COCO test-dev set

Detector	Basis architecture	Pooling	Training data	AP	AP_S	AP_M	AP_L
R-FCN	ResNet-101	RoI pooling	MS COCO	29.2	10.3	32.4	43.3
Precise R-FCN	ResNet-101	PS-Pr-RoI pooling	MS COCO	32.8	15.7	36.2	45.2
Precise R-FCN	ResNet-101	PS-Pr-RoI pooling	MS COCO + oversampling ($3 \times$ small object image)	32.9	16.3	36.2	44.8

5 Conclusion

In this study, the accuracy of small object detection was improved and localization misalignment was eliminated using PS-Pr-RoI pooling. This pooling module was embedded in a two-stage detector R-FCN, which greatly increased detection accuracy for small objects. In a future study, we intend to combine PS-Pr-RoI pooling with other detectors for use in other types of image research [41]. Other detector frameworks could also be easily integrated with the studied module for increased object detection accuracy.

Funding Statement: This project was supported by the National Natural Science Foundation of China under grant U1836208, the Hunan Provincial Natural Science Foundations of China under Grant 2020JJ4626, the Scientific Research Fund of Hunan Provincial Education Department of China under Grant 19B004, the “Double First-class” International Cooperation and Development Scientific Research Project of Changsha University of Science and Technology under Grant

2018IC25, and the Young Teacher Growth Plan Project of Changsha University of Science and Technology under Grant 2019QJCZ076.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] Z. Yang, S. Zhang, Y. Hu, Z. Hu and Y. Huang, "VAE-Stega: Linguistic steganography based on variational auto-encoder," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 880–895, 2021.

[2] L. Y. Xiang, S. H. Yang, Y. H. Liu, Q. Li and C. Z. Zhu, "Novel linguistic steganography based on character-level text generation," *Mathematics*, vol. 8, no. 9, pp. 1558, 2020.

[3] J. Wang, Y. N. Tang, S. M. He, C. Q. Zhao, P. K. Sharma *et al.,* "Logevent2vec: Logevent-to-vector based anomaly detection for large-scale logs in internet of things," *Sensors*, vol. 20, no. 9, pp. 2451, 2020.

[4] D. J. Zeng, Y. Dai, F. Li, J. Wang and A. K. Sangaiah, "Aspect based sentiment analysis by a linguistically regularized CNN with gated mechanism," *Journal of Intelligent & Fuzzy Systems*, vol. 36, no. 5, pp. 3971–3980, 2019.

[5] A. Krizhevsky, I. Sutskever and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 37, no. 9, pp. 1097–1105, 2012.

[6] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conf. on Computer Vision and Pattern Recognition*, Columbus, OH, USA, pp. 580–587, 2014.

[7] P. Felzenszwalb, D. A. McAllester and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *2008 IEEE Conf. on Computer Vision and Pattern Recognition*, Anchorage, AK, USA, pp. 1–8, 2008.

[8] Z. X. Zou, Z. W. Shi, Y. H. Guo and J. P. Ye, "Object detection in 20 years: A survey," arXiv preprint arXiv: 1905.05055v2, pp. 1–39, 2019.

[9] K. M. He, X. Y. Zhang, S. Q. Ren and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[10] S. Q. Ren, K. M. He, R. Girshick and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[11] A. Voulodimos, N. Doulamis, A. Doulamis and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational Intelligence and Neuroscience*, vol. 2018, pp. 1–3, 2018.

[12] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[13] A. Qayyum, I. Ahmad, M. Iftikhar and M. Mazher, "Object detection and fuzzy-based classification using UAV data," *Intelligent Automation & Soft Computing*, vol. 26, no. 4, pp. 693–702, 2020.

[14] C. L. Li, X. M. Sun and J. H. Cai, "Intelligent mobile drone system based on real-time object detection," *Journal on Artificial Intelligence*, vol. 1, no. 1, pp. 1–8, 2019.

[15] M. Kisantal, Z. Wojna, J. Murawski, J. Naruniec and K. Cho, "Augmentation for small object detection," arXiv preprint arXiv: 1902.07296v1, pp. 1–15, 2019.

[16] T. Y. Lin, M. Maire, S. Belongie, J. Hays and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conf. on Computer Vision*, Zurich, Switzerland, pp. 740–755, 2014.

[17] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, pp. 779–788, 2016.

[18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed *et al.,* "SSD: Single shot multibox detector," in *European Conf. on Computer Vision*, Amsterdam, Netherlands, pp. 21–37, 2016.

[19] C. Szegedy, W. Liu, Y. Q. Jia, P. Sermanet, S. Reed *et al.,* "Going deeper with convolutions," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Boston, MA, USA, pp. 1–9, 2015.

[20] T. Y. Lin, P. Dollár, R. Girshick, K. M. He, B. Hariharan *et al.,* "Feature pyramid networks for object detection," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, pp. 2117–2125, 2017.

[21] X. Tang, D. K. Du, Z. Q. He and J. T. Liu, "Pyramidbox: A context-assisted single shot face detector," in *Proc. of the European Conf. on Computer Vision*, Munich, Germany, pp. 797–813, 2018.

[22] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conf.*, San Diego, CA, USA, pp. 886–893, 2005.

[23] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. of the Seventh IEEE Int. Conf. on Computer Vision*, Kerkyra, Greece, pp. 1150–1157, 1999.

[24] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[25] S. Belongie, J. Malik and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 4, pp. 509–522, 2002.

[26] R. Girshick, P. Felzenszwalb and D. Mcallester, "Object detection with grammar models," *Advances in Neural Information Processing Systems*, vol. 24, pp. 442–450, 2011.

[27] P. F. Felzenszwalb, R. B. Girshick and D. McAllester, "Cascade object detection with deformable part models," in *2010 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, San Francisco, CA, USA, pp. 2241–2248, 2010.

[28] T. Malisiewicz, A. Gupta and A. A. Efros, "Ensemble of exemplar-svms for object detection and beyond," in *2011 Int. Conf. on Computer Vision*, Barcelona, Spain, pp. 89–96, 2011.

[29] C. Song, X. Cheng, Y. X. Gu, B. J. Chen and Z. J. Fu, "A review of object detectors in deep learning," *Journal on Artificial Intelligence*, vol. 2, no. 2, pp. 59–77, 2020.

[30] H. Wu, Q. Liu and X. Liu, "A review on deep learning approaches to image classification and object segmentation," *Computers, Materials & Continua*, vol. 60, no. 2, pp. 575–597, 2019.

[31] J. R. R. Uijlings, K. E. A. V. D. Sande, T. Gevers and A. W. M. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[32] A. Creswell, T. White, V. Dumoulin and K. Arulkumaran, "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[33] J. F. Dai, Y. Li, K. M. He and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Advances in Neural Information Processing Systems*, Barcelona, Spain, vol. 1, pp. 379–387, 2016.

[34] K. M. He, X. Y. Zhang, S. Q. Ren and S. Jian, "Deep residual learning for image recognition," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, pp. 770–778, 2016.

[35] R. Girshick, "Fast R-CNN," in *Proc. of the IEEE Int. Conf. on Computer Vision*, Boston, MA, USA, pp. 1440–1448, 2015.

[36] B. Jiang, R. Luo, J. Mao, T. Xiao and Y. Jiang, "Acquisition of localization confidence for accurate object detection," in *Proc. of the European Conf. on Computer Vision*, Munich, Germany, pp. 784–799, 2018.

[37] K. M. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 386–397, 2020.

[38] J. Deng, W. Dong, R. Socher, L. J. Li and F. F. Li, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conf. on Computer Vision and Pattern Recognition*, Miami, FL, USA, pp. 248–255, 2009.

[39] S. Ruder, "An overview of gradient descent optimization algorithms," arXiv preprint arXiv: 1609. 04747v2, pp. 1–14, 2016.

[40] J. F. Dai, H. Z. Qi, Y. W. Xiong, Y. Li, G. Zhang *et al.,* "Deformable convolutional networks," in *IEEE Int. Conf. on Computer Vision*, Venice, Italy, pp. 764–773, 2017.

[41] D. Y. Zhang, X. Chen, F. Li, A. K. Sangaiah and X. L. Ding, "Seam-carved image tampering detection based on the co-occurrence of adjacent LBPs," *Security and Communication Networks*, vol. 2, pp. 1–12, 2020.