A DCNN-based Arbitrarily-Oriented Object Detector for Quality Control and Inspection Application

Kai Yao · Alberto Ortiz · Francisco Bonnin-Pascual

Received: date / Accepted: date

Abstract Following the success of machine vision systems for on-line automated quality control and inspection processes, an object recognition solution is presented in this work for two different specific applications, i.e., the detection of quality control items in surgery toolboxes prepared for sterilizing in a hospital, as well as the detection of defects in vessel hulls to prevent potential structural failures. The solution has two stages. First, a feature pyramid architecture based on Single Shot MultiBox Detector (SSD) is used to improve the detection performance, and a statistical analysis based on ground truth is employed to select parameters of a range of default boxes. Second, a lightweight neural network is exploited to achieve oriented detection results using a regression method. The first stage of the proposed method is capable of detecting the small targets considered in the two scenarios. In the second stage, despite the simplicity, it is efficient to detect elongated targets while maintaining high running efficiency.

Keywords Inspection · Object Detection · Oriented Object Detection · Deep Learning

1 Introduction

Machine vision systems are merging as progressively popular solutions specific to automated on-line quality and process control applications. Optical techniques, enabling non-contact and thus non-destructive inspection, are effectively suited for the crucial correct manipulation of the object under inspection. The existing machine vision system, integrated with the Deep Learning techniques, is capable of achieving consistent and accurate detection results. Deep Convolutional Neural Networks (DCNN) have achieved prominent performance in various computer vision tasks. In the benchmark of Object Detection, methods adopting the DCNN techniques (e.g., Faster R-CNN [25] and SSD [20]) outperformed existing works. Inspired by these works, the present work primarily uses DCNN-based object detection technology in two different tasks, i.e., an inspection task intended for vessel corrosion detection, and another relates to a quality control task that is to detect several control items, which are the sterilization unit of a hospital places in boxes and bags containing surgery tools.

Given the cost of shipbuilding and the performance of the ship, most vessels draw upon high-strength steel materials to design their structure. In the marine environment, steel materials are prone to severe corrosion, and the damage attributed to corrosion may cause substantial losses and even land crew members at risk in some scenarios. Accordingly, corrosion detection should be conducted to prevent engineering disaster from occurring. Fig. 1 presents four corroded patterns recorded in the Inspection dataset. The Quality Control task refers to a precise inspection problem requiring the detection of several control items in the surgery boxes and bags containing surgical tools required to be supplied by the sterilization unit of the hospital for surgeons and nurses before they start the surgery. The mentioned items evidence that the tools have properly undergone the required cleaning process. From left to right and
from top to bottom of Fig. 2, the six types of items to be detected include the label/bar coder adapted to track a box/bag of tools, the yellowish seal, the three types of paper tape changing to the black-, blue-, and pink-striped appearance under the sterilized box/bag, and an internal filter placed inside some boxes and creates the observable white-dotted texture (instead of black-dotted).

For the two tasks, though the ultimate target is to recognize and detect targets in the image, the problems to be solved for each task are different. Unlike objects in practice, the corrosion exhibits diverse shapes, colors, and textures. Moreover, some corrosion is tiny (e.g., pitting), and future harm is difficult to judge. Thus, the various appearances and tiny areas are the main challenges for the Inspection task. As opposed to the mentioned, the Quality Control task has a different situation. In the usual detection method, a straight rectangle is used to locate every object. Although this seems natural for relatively squared objects, such as the label and the seal, it is not straightforward for the paper tape because of its elongated shape. Thus, this task requires a detector capable of detecting the orientation of targets and using a rotated rectangle (RBox) to locate the object.

The present paper proposes a two-stage SSD-based solution detecting targets for the two tasks. In the first stage, a simple feature pyramid architecture is used based on SSD to enhance the detection performance in small targets. Second, a lightweight convolutional neural network is developed to regress the RBox parameters using the output of the first stage. Before training, a K-means clustering approach is adopted to select parameters of a series of default boxes. The proposed method is simple but effective, avoiding massive calculations attributed to considerable RBox proposals.

2 Related Works

This section briefly the relevant researches for DCNN-based straight and oriented object detection methods.

2.1 Object Detection

Object detection refers to a fundamental problem in the computer vision area, aiming to design an algorithm to find objects in pictures without human intervention. Among the recently proposed methods for object detection, two major strategies can be differentiated: (1) methods based on two-stage Region-CNN (R-CNN) algorithm and (2) methods based on one single-stage Network.

R-CNN [6] is one of the first attempts to use a two-stage architecture neural network for object detection. R-CNN processed the problem of object detection as a classification problem based on Region of Interests (ROIs). At the first stage, the selective search [31] algorithm was used to obtain ROIs; second, a Deep Convolutional Neural Network (DCNN) was utilized to extract features as the input of a Support Vector Machine (SVM) classifier. Besides, the training process was expensive in space and time. With VGG-16 [30], it took 2.5 GPU-days for the VOC2007 training set. Fast R-CNN [5] adopted an end-to-end model for object detection and significantly expedited the training process. To be specific, a multi-task loss function was developed to train bounding box regression and classification tasks simultaneously, and an ROI pooling layer was developed and employed to extract features after the last convolutional layer. Subsequently, Ren et al. [28] built a Region Proposal Network (RPN) by exploiting the architecture of Fast R-CNN to generate proposals instead of selective search, sharing the network of extracted features. Moreover, an anchor mechanism was developed to present prior knowledge for the bounding box regression task, making the training processing more stable than before. Faster R-CNN improved detection performance and narrowed the inference time. Under the Faster R-CNN framework, Feature Pyramid Network (FPN) [18] employed a feature pyramid architecture to merge the feature maps from different convolution layers by up-sampling. The multi-scale detection of FPN exhibited better performance, especially for small objects in the COCO benchmark.

On the other hand, some works [13, 20, 26, 27, 34] transferred the object detection problem to a one-stage bounding box regression problem, discarding the proposal generation procedure during the training processes. In YOLO [20], the input picture was split into $S \times S$ grids, and each grid cell was responsible for detecting targets regardless of the center point of the target falls in this cell. YOLO had high speed, whereas it was inaccurate. To tackle down this problem, YOLO9000 [27] was proposed with several improvements: (a) Batch Normalization [9] was applied in the network, bringing 2% improvement on mean Average Precision (mAP); (b) a default box mechanism (e.g., the Anchor Box in Faster R-CNN) was introduced, increasing recall from 81% (presented by YOLO) to 89%; (c) K-means clustering method was adopted to select default box instead of human-selected; (d) a multi-task training strategy was utilized to improve the robustness of the network.
Single Shot MultiBox Detector (SSD) [20] employed a series of default boxes with different ratios of width and height to regress the bounding box parameters. Furthermore, SSD adopted the multi-scale feature maps for object detection. SSD had the advantage that the large scale feature maps were adopted to detect small targets. In contrast, the small feature maps were used to detect big targets, so SSD exhibited the superiority performance for multi-scale object detection. Given the imbalance between positive and negative samples during training, SSD followed an online hard mining (OHEM) strategy to keep the ratio at 3:1. SSD enhanced the performance of YOLO9000 while maintaining its speed and obtained the state-of-the-art PASCAL VOC 2012 benchmark.

### 2.2 Arbitrary-Oriented Object Detection

Some researchers have considered that Straight Bounding Box (BBox) cannot correctly locate in some areas, where considerable objects are dense and staggering. Text detection in natural scenes refers to a typical scene relying on the Oriented Bounding Box (RBox) to locate the text. [11] proposed a method termed as R2CNN based on Faster R-CNN architecture. This method simultaneously regressed the parameters of BBox and RBox by a multi-task loss function. To be specific, a SmoothL1 loss function was employed to regress a tuple of BBox parameters, including coordinates of the center point and its width and height, as well as a tuple of RBox parameters, with the coordinates of the top-left and top-right corners and the height involved. This method achieved good performance on the text detection benchmark ICDAR 2013 and ICDAR 2015. To achieve a more accurate detection performance, [24] implemented a method of RBox’s proposals with the sliding window method. First, a Rotation Region Proposal Network (RRPN) was developed to output proposals of BBox and RBox by several hand-picked scales and angles. Second, similar to the ROI Pooling layer in Fast R-CNN, a Rotation Region of Interest (RROI) Pooling layer was proposed to map the coordinates of RBox proposals to the main network. Lastly, a tuple of RBox’s parameters acted as regressed terms, indicating the coordinates of the center point, height, width, and the rotated angle of RBox. Their approach obtained the benchmark of MSRA-TD500, ICDAR 2013, and ICDAR 2015.

Other studies (e.g., [19] and [16]) utilized one-stage architecture for RBox detection. A strategy was adopted for RBox proposals using a sliding window method with hand-picked parameters, such as scales and angles, on the feature maps. [19] directly regressed the orientation angle, and [16] used the angle to compute the two vertices and height of one RBox, and then employed these as regression targets. The authors tested their methods on different datasets and obtained good performance.

Apparent from methods based on object detection algorithms, other researchers focused on text detection using semantic segmentation backbone. EAST [35] employed a simple and direct pipeline to generate RBox by segmentation score maps. Specific to their work, a multi-channel FCN [22] is used as the backbone and fused feature maps from different convolutional layers. At the end of the network, their approach generated a semantic score map and achieved geometry detection results (e.g., RBox and quadrangle (QUAD) coordinate). Moreover, a loss based on RBox Intersection Over Union (RIOU) was developed, as well as a scale-normalized SmoothL1 loss for QUAD regression. Compared with other detection approaches. Their method achieved better detection performance on MSRA-TD500 and ICDAR 2015.

Since SSD considers detection accuracy and speed, this study integrates a feature pyramid architecture in
SSD to improve the small targets’ detection ability. To improve stability during training, a K-means clustering method is adopted to set the hyper-parameters of default boxes. Finally, a lightweight neural network is designed to regress the RBox parameters based on the BBox detection results in the first stage. The proposed approach is simple but effective and flexible to solve out two tasks.

3 Background

This section briefly discusses the straight bounding box regression method based on SSD in Sec. 3.1 and the oriented bounding boxes regression method in Sec. 3.2.

3.1 Single Shot MultiBox Detector (SSD)

SSD refers to a typical one-stage object detection approach. A standard VGG-16 network is taken as the backbone network, and its last fully-connected layer is replaced with a convolutional layer. Compared with most detection algorithm based on R-CNN, such as [9] and [28], SSD does not require any extra procedure to generate proposals. Alternatively, a mechanism of default boxes is used, presenting numerous default boxes for bounding box regression. On the other hand, inconsistent with R-CNN based methods, obtaining feature maps from a single convolutional layer, feature maps are extracted from layer conv4_3, conv7, conv8_2, conv9_2, and conv10_2. Subsequently, feature maps are connected and sent to a multi-task loss function to regress coordinates of bounding boxes, and the confidence value is computed for the respective category.

The idea of the use of default boxes is to provide prior knowledge for bounding box regression, which can improve the stability of the training. Similar to the anchor box in Faster R-CNN, the parameters of default boxes are set by different scales and aspect ratios. Then a sliding window strategy is applied for each pixel on feature maps to generate default boxes. Next, two matched pairs are found, i.e., the matched pair of ground truth and predicted bounding boxes, as well as the matched pair of ground truth and default box. The matched default boxes are considered prior boxes, and are adopted to regress targets in the loss function.

At the end of the network, a multi-task loss function is adopted, consisting of a SmoothL1 Loss for bounding box regression ($L_{reg}$), as well as a Softmax Cross-Entropy Loss over multi-category confidence value $c$ as defined below,

$$L_{cls}(x, c) = - \sum_{i \in Pos} x_{ij} \log(c_{ij}^p) - \sum_{i \in Neg} \log(\hat{c}_{ij}^p)$$

where $x_{ij}$ is an indicator function for the matched pair that comprises the predicted bounding box $p_i$ and the ground truth $g_j$.

For the classification task, the loss function is a Softmax Cross-Entropy Loss over multi-category confidence value $c$ as defined below,

$$L(x, c, p, g) = \frac{1}{N}(L_{cls}(x, c) + \alpha L_{loc}(x, p, g))$$

where $N$ denotes the number of matched prior boxes; $\alpha$ is adopted to balance the impact of classification loss and bounding box regression loss on the final loss.

The bounding box regression consists of two optimized box-delta parameters ($\hat{t}_i, t_j$), where $\hat{t}_i$ is determined from the predicted bounding box $p_i$ and the prior box $d$ and $t_j$ is calculated by ground truth $g_j$ and the prior box $d$. Moreover, the box-deltas are calculated with the center point coordinate $(c_x, c_y)$ and the scale $(w, h)$ of the bounding box, as expressed below:

$$\hat{t}_{i}^{cx} = \frac{p_{i}^{cx} - d_{x}}{d_{w}}$$
$$\hat{t}_{i}^{cy} = \frac{p_{i}^{cy} - d_{y}}{d_{h}}$$
$$\hat{t}_{i}^{w} = \log\left(\frac{p_{i}^{w}}{d_{w}}\right)$$
$$\hat{t}_{i}^{h} = \log\left(\frac{p_{i}^{h}}{d_{h}}\right)$$

Thus, the bounding box regression loss function is

$$L_{reg}(x, p, g) = \sum_{i \in Pos} \sum_{m \in \{c_x, c_y, w, h\}} x_{ij} \text{SmoothL1}(\hat{t}_{im} - t_{jm})$$

where $x_{ij}$ is an indicator function for the matched pair that comprises the predicted bounding box $p_i$ and the ground truth $g_j$.

For the classification task, the loss function is a Softmax Cross-Entropy Loss over multi-category confidence value $c$ as defined below,
Fig. 3 Samples of BBox (a) and RBox (b) in our dataset, and the different definition of RBox regression targets is shown in (c,d).

3.2 Oriented Objects Detection

On the whole, the BBox detection algorithms based on SSD exhibit high performance. However, a straight rectangle cannot locate objects of elongated shape. For instance, Fig. 3 [a] obviously indicates that the red rectangle cannot accurately locate the paper tape. Meanwhile, it contains a large area of background in the bounding box, and the area of paper tape only takes a small portion of the rectangle, which probably brings instability at the training phase. However, the RBox in Fig. 3 [b] can perfectly locate the tape.

Many existing works focusing on oriented detection comply with straight bounding box detection architecture. To be specific, their studies achieved RBox detection results by adding a regression loss function for the RBox parameters. Similar to a general learning model, a loss function is adopted to determine the error between prediction and ground truth. Like most object detection methods, as usual, they need to classify the category of the bounding box, as well as to regress the localization of the target. For the classification task, the Cross-Entropy ($L_{ce}$) loss function was extensively used with Softmax activated function, as expressed in Eq. 4. For the regression tasks, especially for RBox parameters regression, different solutions have been proposed. Next, some common loss functions and the parameters of RBox are to be discussed.

3.2.1 Regression Loss function

SmoothL1 Loss. SmoothL1 loss function is one of the most common loss functions for the bounding box regression task, such as in \cite{1,19,23,24,29,32,33}, as defined below:

$$L_{reg} = \sum_{i \in S} \text{Smooth}_{L1}(p_i, p^*)$$  \hspace{1cm} (6)

where,

$$\text{Smooth}_{L1}(x) = \begin{cases} 
0.5x^2, & |x| < 1 \\
|x| - 0.5, & \text{otherwise} 
\end{cases}$$  \hspace{1cm} (7)

where, $p$ and $p^*$ denote the predicted value and ground truth respectively; $x$ represents the error between $p$ and $p^*$. The derivative of SmoothL1 is expressed as:

$$\frac{d}{dx}(\text{Smooth}_{L1}(x)) = \begin{cases} 
-1, & x \leq -1 \\
x, & x \leq 1 \\
1, & x > 1
\end{cases}$$  \hspace{1cm} (8)

In \cite{21}, Liu et al. develop a continuous regression loss function based on SmoothL1, named SmoothLn, as expressed in Eq. 9. Their loss function achieves the tradeoff between robustness and stability.

$$\text{SmoothLn} = (|x| + 1)\ln(|x| + 1) - |x|$$  \hspace{1cm} (9)

Mean Square Error. Mean Square Error (MSE) refers to another typical regression loss function for RBox regression in \cite{7,25}, as defined below:

$$L_{mse} = ||y - y^*||^2$$  \hspace{1cm} (10)

where $y$ and $y^*$ indicate actual value and predicted value respectively.

3.2.2 RBox Regression Parameters

For the oriented object detection methods, most existing studies are usually parameterized into five variables (e.g., the coordinates of two vertexes and the height ($h$) or orientation ($\theta$) of RBox). As illustrated in Fig. 3 [c], the regression targets of RBox contain two vertexes on the diagonal and the orientation of RBox. On the other hand, according to Fig. 3 [d], another parameterized method employs the coordinates of two vertexes and the height of RBox.

For the expression of regression terms of RBox, two main types are categorized. Following the idea in \cite{32}, the bounding box regression was categorized into two branches, which are direct regression and indirect regression. The indirect regression method is derived from...
R-CNN, computing a set of offsets using ground truth and prior boxes, as expressed in Eq. 2 such as \[2, 11, 23, 24, 29, 32\]. As its name described, the direct regression method directly calculates the error between the prediction and ground truth \[7, 8, 35\].

For our tasks, first, since the orientation angle is periodic, which can confuse the network, so the parameterized method presented in Fig. 3 \[d\] is selected. Second, inconsistent with the text detection task, the orientation of text is assigned to a unified orientation to facilitate reading. However, the orientation of the targets in our tasks is commonly random and diverse. Thus, prior orientation for the RBox regression provides no necessary assistance. Moreover, using prior boxes for RBox increases the calculation to transfer the indirect regression terms to the real orientation, thereby reducing the system efficiency. Thus, a direct method is selected for RBox regression.

### 4 Detector Overview

The detector proposed in this work comprises two stages. The first stage employs a method based on SSD to regress the BBox containing the objects of interest. The set of default boxes employed by SSD is determined after a clustering analysis on the training set. Moreover, a feature pyramid architecture is adopted to fuse different feature maps from the network. This work primarily searches relevant bounding boxes and improves the object localization performance. At the second stage, a specifically designed network is adopted to regress the parameters of RBox maximally contained in one of the BBoxes. Furthermore, the parameterization of the dataset and the strategy of training and testing are presented.

In the following sections, the parameterization employed for bounding boxes is first described in Sec. 4.1. Second, the SSD-based method is presented in Sec. 4.2. Lastly, a full description of RBox regression is demonstrated in Sec. 4.3.

4.1 Bounding Boxes Parameterization

Figure 4 illustrates how bounding boxes are parameterized for this application. First, the yellow lines describe a 4-side polygon minimally enclosing the object, from which the minimal RBox is generated, indicating the use of a blue rectangle. Then, a minimal BBox is obtained from the rotated bounding box, demonstrating the use of a red rectangle. The anchor point coordinates (blue point in Fig. 4[a,b]) parameterize the latter \((c_x, c_y)\) and the box size \((w_b, h_b)\) as in the standard SSD. The aforementioned has been adopted to generate the ground truth necessary during training.

In addition, as part of the ground truth, and for testing purposes, two different datasets are defined for the boxes associated with each image’s targets illustrated through Fig. 4[c,d]. On the other hand, approach A in Fig. 4[c] defines one box for every target of the training image. Though this seems natural for relatively squared objects (e.g., the label and the seal presented in Fig. 4[a,b]), it is not straightforward for the paper tape since its elongated shape. Thus, approach B in Fig. 4[d] is defined, splitting the target into several parts to expedite better training and latter detection of such a type of target.

According to Fig. 4[c], for the large area of ground truth in approach A, it is easy to get predicted BBox with a high IOU value as opposed to the small area of the ground truth in approach B. However, a predicted BBox with a high IOU for approach A contains only a small part of the target and most area inside it is the background (e.g., the yellow rectangle in Fig. 4[c]). Thus, approach B in Fig. 4[d] refers to an effective and feasible method for this work.

4.2 Straight Bounding Boxes (BBox) Regression

This section demonstrates the methodology of the feature pyramid architecture in SSD. Subsequently, the clustering method to select the hyper-parameters of default boxes is illustrated.

4.2.1 Feature Pyramid Single Shot Multi-box Detector (FPSSD)

Single Shot MultiBox Detector considers the use of feature maps from different layers to regress bounding boxes. More precisely, SSD applies large scale feature maps to detect small targets. Conversely, it uses small scale feature maps to detect big targets. However, the large scale feature maps contain numerous detailed features (e.g., edges, shape, and textures), whereas they lack the semantic information. Using features pyramid architecture to fuse the features maps from top layers to bottom layers, a type of enhanced features containing the semantic information and detailed features are obtained with the proposed method, which is exploited to detect different scales targets.

The idea of the Feature Pyramid originates from the Image Pyramid method. The Image Pyramid is a
method analyzing the image with multiple resolutions, generated by multi-scale sampling on the original image via a Gaussian kernel. The Image Pyramid imitates the multi-scale features of the image. As assisted by the hierarchical architecture network, a feature pyramid can be constructed in one feed-forward procedure, so the computational cost of multi-scale sampling can be eliminated. Therefore, the Feature Pyramid can efficiently address the multi-scale problem with a relative cost.

Several existing approaches aim at using the Feature Pyramid in DCNN, such as [4, 15, 18]. According to Fig. 5, there are four methods for fusing feature maps. Method A in Fig. 5 illustrates the most common strategy, which merges feature maps layer by layer by element-wise addition and applies detector on each scale feature map. Though the Feature Pyramid has been proved to be able to efficiently improve the detection performance for small targets in [18], it brings massive computation to reduce the detection efficiency, which is what this work wants to avoid. Another method is a lightweight fusion method named FSSD [15] shown in Fig. 5 [B]. In this method, the top-down path and down-top path are obviously independent of each other. First, it fuses the feature maps from the top to the bottom layers in the top-down path by enlarging the resolution of feature maps of top layers. Then, in the top-down path, the resolution of feature maps is decreased to four groups by interpolation. Lastly, the different resolution feature maps are combined by the concatenate layer and sent to the loss function. Though this method is capable of saving the computational cost as compared with method A, the feature maps applied in the detector are singular and lack various semantic information. According to Fig. 5 [C], the proposed method employs the identical strategy to FPN to fuse the feature maps. Moreover, to reduce the computation, a concatenate layer is implemented to combine the different feature maps. Subsequently, the combined feature maps are fed into the detector for detection. Lastly, Fig. 5 [D] illustrates the strategy of the original SSD applied. It shows that SSD does not have the feature fusion module, and it lacks the capability to capture both the low-level details features and high-level semantic information.

For a detailed demonstration of the FPSSD, Fig. 6 gives a diagram graph of the architecture. In the proposed method, the feature maps are extracted from conv4_3, fc7, conv6_2, conv7_2, conv8_2, and conv9_2 in the original SSD network. Next, a deconvolution layer is utilized to increase the dimension for the respective feature map. Since feature maps from top layers have more output channels than feature maps from bottom layers, a 1 × 1 convolutional layer termed lateral connection in [18] is used to unify the output channels of all feature maps. Lastly, feature maps integrated with top and bottom layers are sent to the detector to predict the category and localization of targets.

### 4.2.2 Default Boxes Selection

Overall, SSD predifines 9 default boxes per feature map location by imposing different size combinations \((w_k, h_k)\). Since the shape of the ground truth can vary significantly, and a group of selected default boxes termed as prior boxes are adopted to calculate the regression targets, so a proper selection of default boxes is critical to achieving a prominent detection performance. As already suggested in [3, 27], such a proper selection contributes to the stability of the underlying optimization process, converges faster, and effectively optimizes the IOU between predicted and correct boxes. Thus, our detection approach employs default boxes selected automatically by complying with the available data.

To be more specific, the well-known K-means algorithm runs over the bounding boxes belong to the ground truth, and box width and height act as the clustering features. Instead of the Euclidean distance, typically used by K-means implementation, IOU is defined as a distance metric since the former tends to miss large bounding boxes. Accordingly, the distance between a sample box \(b_i\) and the cluster centroid \(c_j\) is defined as:
Different strategies for fusing feature maps: A. feature maps are fused from top to bottom layer by layer, adopted by FPN; B. a lightweight architecture merges feature maps from top to bottom; C. the method of our FPSSD; D. the original SSD strategy uses feature maps from different layer separately.

\[
\begin{align*}
    d(b_i, c_j) &= 1 - \text{IOU}(b_i, c_j) \\
    &= 1 - \frac{o(b_i, c_j)}{a(b_i) + a(c_j) - o(b_i, c_j)}
\end{align*}
\]  

(11)

where \(o(\cdot, \cdot)\) denotes area overlap and \(a(\cdot)\) denotes area.

Table 1 lists the mIOU value for hand-picked default boxes vs. automatically selected using clustering.

| Dataset       | Approach    | # def. boxes | mIOU (%) |
|---------------|-------------|--------------|----------|
| Quality Control| Hand-Picked 4 | 36.75        |          |
|                | Hand-Picked 5 | 42.51        |          |
|                | Hand-Picked 6 | 49.53        |          |
|                | Clustering 4  | 55.44        |          |
|                | Clustering 5  | 58.05        |          |
|                | Clustering 6  | 61.70        |          |
| Inspection     | Hand-Picked 4 | 35.93        |          |
|                | Hand-Picked 5 | 37.96        |          |
|                | Hand-Picked 6 | 42.75        |          |
|                | Clustering 4  | 61.82        |          |
|                | Clustering 5  | 63.37        |          |
|                | Clustering 6  | 65.31        |          |

better performance will be (the trend can be observed to continue for \(\# \text{ def. boxes} \geq 7\)), although the number of clusters should not be high to keep the running time reasonable.

4.3 Oriented Bounding Boxes (RBox) Regression

To implement the RBox detection, the straight bounding box acts as the input to a specifically designed lightweight network in charge of regressing the RBox parameters. In this section, the proposed method is presented, expressing the RBox parameterization in Sec. 4.3.1 and the network architecture in Sec. 4.3.2.

4.3.1 RBox Parameterization

RBox is expressed by the intersects \((d_1, d_2)\) of the upper side of the rotated rectangle with the sides of the straight rectangle, as shown in Fig. 4 [a]. Optionally, a parameter \(h\) is added to select one of the two possible rectangles that may arise from the tuple \((d_1, d_2)\). To determine \((d_1, d_2)\) individually, a clockwise order is defined onto the four corners of the RBox as well in Fig. 4 [b]. Thus, the network regresses the value \((d_1, d_2, h)\) we defined.

4.3.2 RBox Regression Network

A lightweight convolutional network is developed based on LeNet [14] to regress RBox parameters, whereas it presents several differences below: (1) the input size is \(63 \times 63\) after incorporating an additional convolution layer at the beginning of the network, in order
to avoid reducing the image to LeNet’s 28 × 28 pixels, which means losing too much information; (2) after the convolutional layer, normalization is used to accelerate the converge procedure during training, in which Batch Normalization \[9\] is selected since it can decrease the effect of covariate shift from the hidden layers added after convolution; (3) since the bounding box parameters \((d_1, d_2, h)\) range from 0 to 1, a sigmoid layer lies between the last fully connected layer and the loss layer; (4) lastly, the Euclidean distance acts as the target of regression, and the final layer refers to a Euclidean loss layer:

\[
L(d, g) = \frac{1}{2N} \sum_{i \in N} (\|d_1 - g_1\|^2 + \|d_2 - g_2\|^2 + \|d_h - g_h\|^2) \tag{12}
\]

where \(d\) denotes the predicted offsets and height; \(g\) represents the ground truth; \(N\) is the size of the minibatch. The architecture of the RBox regression network is illustrated in Fig. 7.

5 Experiments and Discussion

This section reports the experimental results and assesses the performance of the detectors. First, the experimental setup is presented in Sec. 5.1. Second, the results obtained regarding the BBox regression are presented in Sec. 5.2. Finally, the results of the RBox regression are given in Sec. 5.3.

5.1 Experimental Setup

**Experimental Environment.** The experiments in this work are implemented using Caffe \[10\] platform. The Quality Control dataset consists of 484 images for overall 7 categories, and the Inspection dataset contains 214 images for corrosion detection. Given the problem of quantization due to down-sampling and up-sampling operation, all the images are resized to 512. Next, an SGD optimizer is selected, the weight decay and the momentum are set as 0.001 and 0.9, respectively. Inspiring by the idea of YOLO9000, default boxes are selected by the results of clusters (6 centroids considering the running time), as employed for all the experiments in this work.

The network is trained on the PC platform (CPU: Intel i9-9900K, GPU: Nvidia RTX 2080Ti GPU, RAM: 64G). Similar to the training configuration of VOC2007, we adopt a multiple steps strategy, where the learning rate is set to \(10^{-5}\) at the first 8000 iterations, then 6000 iterations with a learning rate of \(10^{-6}\), and 6000 iterations with a learning rate of \(10^{-7}\). The batch size is set to 10, which is the best configuration for our GPU.

**Assessment Metrics.** To evaluate the performance of our approach, the mean Intersection Over Union (mIOU) (Jaccard index) is determined with a predicted bounding box \((P)\) and ground truth \((G)\), as expressed in Eq. 13. Moreover, the standard Recall \((R)\) and Precision \((P)\) are obtained to assess the detection performance using
BBox and RBox. For our two tasks, more targets should be detected, so several potential risks can be identified in the Inspection task, and more evidence can be presented in the Quality Control task. In other words, our solution pursues higher recall value as well. Identical to the original SSD, the VGG-16 network is taken as the backbone. Similar to the policy for common object detection, the confidence threshold for detection is set to 0.7.

\[
mIOU = \frac{P \cap G}{P \cup G} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{13}
\]

To assess the performance of oriented detection, we compute the mean RBox IOU (mRIOU) as metrics by Eq.\([13]\). Inconsistent with the mIOU of a straight bounding box, the shape of the intersection of two RBoxes is irregular, so the area of intersection is determined as follows. For any convex polygon \(P\), the vertices of the polygon are arranged counterclockwise as \(\{v_1, v_2, \cdots, v_n\}\), and the vertex coordinates are \(\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}\). Accordingly, the area of \(P\) can be considered the sum of triangles, consisting of three vertices of the polygon. On that basis, the area \(S_p\) of polygon is written as:

\[
S_p = \frac{1}{2} \sum_{i=1}^{n} (x_i y_{i+1} - x_{i+1} y_i) \tag{14}
\]

Besides, to compare the regression results directly, the mean absolute error (MAE) that is the difference between the prediction and ground truth on the test set is calculated by Eq.\([15]\).

\[
\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |x_p - x_g| \tag{15}
\]

where \(x_p\) denotes prediction and \(x_g\) represents the corresponding ground truth. Therefore, lower MAE means better regression results.

Comparative Experiments.

For the BBox detection, SSD512 (using 6 clustered centroids to select default boxes and \(512 \times 512\) input size) is taken as the baseline for BBox detection. For the RBox regression network, first, this work considers finetuning the pre-trained AlexNet\([12]\) to regress the RBox parameters as the baseline. To be more precise, except for the last fully-connected layer, the weights of other layers are frozen, and we only train the weights of the last fully-connected layer. On the other hand, there are previous oriented detection solutions for text detection (e.g., TextBoxes++\([17]\)), which is an SSD-based method for text detection. It has two stages. In the first stage, TextBoxes++ can output straight and oriented detection results, and in the second stage, the authors design a neural network to recognize the context of the text. Thus, we select the first stage of TextBoxes++ to compare the RBox detection performance.

5.2 BBox Detection Results

In this part, we implement the proposed method for our two tasks. First, an ablation study is conducted focusing on the lateral connections and the different methods of feature map fusion in the Feature Pyramid architecture. Second, the experimental results are presented, and its performance is compared with the baseline.

5.2.1 Ablation Study

Table\[2\] lists the performance of FPSSD on our two tasks. As can be seen, the comparative experiments (\(a, d\)) employ the lateral connection, while experiments (\(b, c\)) do not, so they are adopted to discuss the effect of lateral connection. Besides, experiments (\(a, b\)) adopt element-wise sum to fuse feature maps, and experiments (\(c, d\)) apply the Concatenate layer to combine feature maps. So, the two group experiments are used to select the better way to fuse feature maps. At last, the standard mAP, Recall, and Precision are calculated to assess the detection performance.

The role of lateral connections. The lateral connection is a convolutional layer with a \(1 \times 1\) kernel, which is adapted to unify the output channels from top to bottom layers. In the VGG-16 network, the output channels of the top layers are more than the output channels of the bottom layers. Accordingly, after the upsampling, the output channels should be unified. On the other hand, in the feature pyramid architecture, the scale of feature maps decreases gradually through a series of pooling layers, and the deconvolution operation is adopted to expand the scale of feature maps. In the mentioned process, the locations of targets in feature maps will generate offset. Using the lateral connection, the locations of targets can be passed from the finer level of the top layer via the lateral connections to the bottom layer.

Specific to the Quality Control task, the FPSSD obtains better performance than the baseline. We can see the value of mAP and recall of case (\(a, b, c, d\)) is higher
Table 2: Ablation study: effect of lateral connection and different feature maps fusion approaches in the Feature Pyramid architecture. FPNL: Feature Pyramid Non-Lateral, FPL: Feature Pyramid Lateral, SUM: element-wise sum, CAT: concatenate layer.

| DataSet  | Approach                      | mAP(%) | Recall(%) | Precision(%) |
|----------|-------------------------------|--------|-----------|--------------|
| Quality Control | SSD 512                       | 80.55  | 81.11     | 96.63        |
|          | FPSSD 512 + FPNL + SUM (a)    | 84.71  | 85.43     | 93.24        |
|          | FPSSD 512 + FPL + SUM (b)     | 87.91  | 88.15     | 95.50        |
|          | FPSSD 512 + FPL + CAT (c)     | 86.32  | 86.82     | 95.79        |
|          | FPSSD 512 + FPNL + CAT (d)    | 85.83  | 86.46     | 91.24        |
| Inspection | SSD 512                        | 82.18  | 83.11     | 94.34        |
|          | FPSSD 512 + FPNL + SUM (a)    | 81.31  | 82.41     | 95.13        |
|          | FPSSD 512 + FPL + SUM (b)     | 90.91  | 91.13     | 1.0          |
|          | FPSSD 512 + FPL + CAT (c)     | 81.33  | 82.64     | 94.33        |
|          | FPSSD 512 + FPNL + CAT (d)    | 81.72  | 82.62     | 95.63        |

than the original SSD, as shown in Tab. 2. Then, comparing the performance between case a and b on Quality Control dataset, case b is suggested to obtain higher mAP, recall, and precision than case a. Identically, the metrics of case c are higher than those of case d. Thus, the lateral connection is conducive to improving the detection performance for the Quality Control task.

As for the Inspection task, the performance of FPSSD is inconsistent with the performance of the Quality Control task. Experiments a, c, and d are suggested to obtain the lower mAP and recall compared with the baseline listed in Tab. 2. It is considered that there are two main factors leading to these results: the detected target of the Inspection task is corrosion, which is not an object in practice, displaying the irregular shape and various colors in the identical image. However, the target of the same category in the Quality Control dataset shares similar characteristics between each other. Another factor is that some small regions of defects are excessively small to be considered defects, while it displays a similar color and shape to the big region labeled as a defect. Thus, the two factors introduce instability in our experiments. However, case b, using the lateral connection, obtains the maximum metrics, significantly higher than the baseline.

Notably, in our two tasks, we are mainly interested in detecting all the targets despite this means increasing the number of false positives in the prediction. Therefore, we seek to attain high values of recall, allowing a reduction in precision. In Tab. 2, the maximum recall is obtained in case b (87.91% and 96.63%) on two datasets, employing the lateral connection in the feature pyramid architecture.

Concatenate or Element-wise Sum? The Concatenate and Element-wise Sum are two common methods for combining two input vectors. In most of the Deep Learning frameworks, the concatenate layer refers to a utility layer connecting its multiple inputs to one single output according to the channel set by the user, and Element-wise sum adds the feature values at the identical position on the two input feature maps.

5.2.2 Experimental Results

Figure 8 displays some samples in the detection results of FPSSD. Moreover, the results of the original SSD are also displayed for comparison. It is observed that the proposed method can obtain better performance than SSD512.

For the application of the Quality Control task, the most difficult task is to detect three kinds of paper tape and discriminate its category. Obviously, the first row is more accurate than the second row in Fig. 8. Even in a very intensive scene, FPSSD can detect almost all parts of paper tape. For the Inspection dataset (the third and fourth rows in Fig. 8), the proposed approach also achieves better results, and they are able to detect the small region of corrosion as well.

Table 3 lists the quantitative results. For the Quality Control task, it is clear that our FPSSD obtains higher performance than SSD. In Tab. 3, FPSSD obtains 0.7191, 0.9139, and 0.8219 recall for three types...
of paper tape, which are 0.1723, 0.03, and 0.0832 higher than the recall value of the original SSD. By comparing the average value of mIOU, the mIOU of all categories of FPSSD is indicated to be 0.8443, which is 0.0039 higher than SSD512. However, the average recall of FPSSD is higher 6.04% than SSD512. Thus, FPSSD can detect more targets than SSD512 and produce the same quality bounding boxes as SSD512. For other targets (Label, Seal, and Filter) in the Quality Control dataset, FPSSD obtains higher metrics than SSD512.

As revealed from the comparison of the general performance, all the average recall, precision, mAP, and mIOU are higher than those of SSD512.

For the Inspection task, the recall and precision
and it is not derivative when input is 0. In our case, L is sensitive to the outliers because of the squared error dynamically altered, so it can converge fast. Derivable in the range of all input, and its gradient is opposed to two loss functions, MSE (L) (as suggested from the example in Fig. 8 [fourth column] in Table 3). As explained in previous sections, the mentioned targets get, the results of FPSSD are inaccurate, and some detection on both tasks, however, for some elongate targets, shown in the second row in Fig. 9. In these figures, the red points indicate the offset (d1, d2), while the green line is adopted to represent the third regression term (d3). Furthermore, the black line is adopted to connect two red points to display the orientation of the target. Obviously, the black line in the first row (using two regression targets) better adhere to the orientation of the target than the second row (using three regression targets).

5.3.1 Discussion on RBox Regression Targets

According to Fig. 4[a], if (d1, d2, h) is known, the unique RBox can be obtained inside one BBox. On the other hand, given (d1, d2), the orientation of the RBox can be determined. In this way, the h of RBox can start with the point of d1 or d2, respectively, resulting in two oriented bounding boxes.

Figure 9 gives some examples of the two tasks’ test set with two and three regression targets. Moreover, the AlexNet is finetuned to regress the RBox parameters, shown in the second row in Fig. 9. In these figures, the red points indicate the offset (d1 and d2), while the green line is adopted to represent the third regression term (d3). Furthermore, the black line is adopted to connect two red points to display the orientation of the target. Obviously, the black line in the first row (using two regression targets) better adhere to the orientation of the target than the second row (using three regression targets).

In Table 4, the MAE for each regression target is computed to assess the regression performance of using different regression targets. As can be observed, the MAE of d1 and d2 of the 2 regression targets method is lower than the MAE value in the 3 regression targets method. Moreover, the average MAE of the 2 regression targets method is lower than the 3 regression targets method. In this way, the method of the 2 regression targets has better performance.

On the other hand, we finetune the fully-connected layers of AlexNex to regress the RBox parameters. For our two tasks, it can be seen clearly that the average MAE of AlexNet is higher than our network in Tab. 4. Accordingly, our RBox regression network has better regression performance than AlexNex for our tasks.

At last, we connect the FPSSD and the RBox regression network to get oriented detection in the inference stage. The input of the RBox regression network is the prediction of FPSSD, where there are offsets between the prediction of FPSSD and the ground truth.

### Table 3: Performance results of FPSSD (R-average recall, P-average precision, mAP-mean average precision, mIOU-mean intersection of union)

| DataSet       | Class       | R   | P   | mAP  | mIOU  |
|---------------|-------------|-----|-----|------|-------|
| Inspection    | Corrosion   | 0.9113 | 1.0 | 0.9091 | 0.9375 |
| (FPSSD)       |             |     |     |      |       |
| Inspection    | Corrosion   | 0.8311 | 0.9434 | 0.8218 | 0.8486 |
| (SSD512)      |             |     |     |      |       |
| Quality       | Label       | 0.9177 | 0.9779 | 0.9097 | 0.8707 |
| Control       | Seal        | 0.8566 | 0.9697 | 0.8461 | 0.8382 |
| (FPSSD)       | Black tape  | 0.7191 | 0.8793 | 0.7055 | 0.7695 |
|                 | Blue tape   | 0.9139 | 0.9421 | 0.9093 | 0.8468 |
|                 | Pink tape   | 0.8219 | 0.9614 | 0.8206 | 0.8236 |
|                 | Intl. Filter| 1.0  | 1.0 | 1.0  | 0.9166 |
| Average        | Label       | 0.8821 | 0.9691 | 0.8783 | 0.8484 |
| Quality        | Seal        | 0.8301 | 0.9769 | 0.8289 | 0.8256 |
| Control        | Black tape  | 0.5468 | 0.9342 | 0.5387 | 0.7673 |
| (SSD512)       | Blue tape   | 0.8839 | 0.9328 | 0.8773 | 0.8346 |
|                 | Pink tape   | 0.7387 | 0.9668 | 0.7261 | 0.8259 |
|                 | Intl. Filter| 0.9841 | 1.0  | 0.9841 | 0.9111 |
| Average        | Label       | 0.8111 | 0.9663 | 0.8055 | 0.8404 |

FPSSD are 7.82% and 5.66% higher than SSD512, while it is capable of detecting smaller areas of corrosion under the identical definition of default boxes, shown in Fig. 3. On the other hand, comparing the mIOU, the mIOU of FPSSD takes up 93.75%, while the mIOU of SSD512 is 84.86%, demonstrating that FPSSD can produce higher quality bounding box than SSD512.

At last, FPSSD improves the performance of detecting small-scale corrosion on the Inspection dataset and is capable of identifying targets in the dense scene on the Quality Control dataset.

5.3 RBox Regression Results

Though FPSSD can obtain good performance on BBox detection on both tasks, however, for some elongate targets, the results of FPSSD are inaccurate, and some bounding boxes contain several parts of other targets (as suggested from Fig. 3 [fourth column]). As explained in previous sections, the mentioned two issues are expected to be solved with RBox detection.

As for the bounding box regression, SmoothL1 (L1) and MSE (L2) loss functions are most widely used. As opposed to two loss functions, L2 loss is continuous and derivable in the range of all input, and its gradient is dynamically altered, so it can converge fast. L2 loss is sensitive to the outliers because of the squared error value. However, L1 loss is insensitive to outliers, and it is not derivative when input is 0. In our case, since the sigmoid function is adopted at the end of the network, and the output of the network has no apparent outliers, so L2 loss is used.

The RBox regression dataset here is built on the BBox ground truth of our two datasets by cropping and resizing, and a random offset value is also set for the four vertices of the BBox ground truth. At the inference stage, the output of FPSSD is applied as the input of the RBox regression network, and a direct regression method is employed to obtain the RBox detection results.
Therefore, the distribution of the RBox training set is different from the prediction of FPSSD, which requires the RBox regression network to be robust to process various input images. Addressing this problem, we set a random offset for the coordinates of four vertexes of ground truth to construct the training set of RBox regression. In the following subsection, the performance of the final oriented detection results of the solution can be assessed.

### 5.3.2 Experimental Results

Figure 10 displays the final results of our solution. These results are divided into 3 groups. The first group provides results for the case of using two regression targets, which include two oriented bounding boxes (i.e., red and green), corresponding to starting with the point of $d_1$ or $d_2$ to compute the height of RBox. The second group presents the results of 3 regression targets. The last group displays some samples of TextBoxes++ [16] predictions.

As shown in Fig. 10, the RBox regression network (using 2 regression targets) can provide more accurate oriented detection than TextBoxes++. Compared with the results of 3 regression targets, though it provides only 1 oriented detection, it cannot correctly and effectively detect the orientation of the targets. As discussed in the previous subsection, the three targets method has weaker robustness than the two targets approach. As for the results of TextBoxes++, due to the small scale of the dataset of our two tasks, though the network has converged, it lacks the ability to detect the target orientation.

Table 5 compares the mRIOU for the results in Fig. 10. For a fair comparison, we select the biggest RBox for the case of using 2 regression targets. The value of mRIOU for the case of two regression targets is higher than the value for the case of three regression targets for all of the categories on the Quality Control dataset. For the elongated objects (the three types of paper tape), the value of mRIOU value achieved by using 2 regression targets reaches 0.6247, 0.5604, and 0.4993, which are significantly higher than others. As for the Inspection task, the mRIOU for the case of 2 regression targets is also highest than others.

Thus, our solution can provide more accurate results in the case of BBox and RBox detection.

### 6 Conclusion and Future Work

A two-stage arbitrarily-oriented object detection method making use of regression of oriented bounding box parameters has been expressed for the Inspection and
Fig. 10 The final prediction of our solution. The group $a$ denotes detection results for our two tasks using two regression targets; the group $b$ shows detection results for our two tasks using three regression targets; the final group $c$ indicates the detection results of TextBoxes++. 
Quality Control application. At the first stage, we add
a feature pyramid architecture to fuse the feature maps
in the SSD network and select parameters of default
boxes by a clustering approach. As revealed from
the experimental results, it improves the detection perfor-
mance of SSD without affecting the running time a lot
comparing with the original SSD. At the second stage,
a simple but effective and flexible neural network is de-
signed to regress RBox. The network can predict the
orientation of targets for our two tasks.

Subsequent work plans to apply a more efficient Fea-
ture Pyramid architecture on the deeper backbone
to improve the detection performance. Besides, the RBox
regression network should be more robust. Thus, our
solution can be enhanced by presenting more accurate
prediction from the first stage, and designing a more
robust and efficient RBox regression network in the sec-
ond stage.

References
1. Dai, Y., Huang, Z., Gao, Y., Xu, Y., Chen, K., Guo,
J., Qiu, W.: Fused text segmentation networks for multi-
oriented scene text detection. In: 2018 24th International
Conference on Pattern Recognition (ICPR), pp. 3604–
3609. IEEE (2018)
2. Dickenson, M., Gueguen, L.: Rotated rectangles for sym-
ylized building footprint extraction. In: CVPR Work-
shops, pp. 225–228 (2018)
3. Erhan, D., Szegedy, C., Toshev, A., Anguelov, D.: Scal-
able object detection using deep neural networks. In: Pro-
cedings of the IEEE conference on computer vision
and pattern recognition, pp. 2147–2154 (2014)
4. Fu, C.Y., Liu, W., Ranga, A., Tyagi, A., Berg, A.C.: Dsd:
Deconvolutional single shot detector. arXiv preprint
arXiv:1701.06659 (2017)
5. Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE
international conference on computer vision, pp. 1440–
1448 (2015)
6. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich fea-
ture hierarchies for accurate object detection and seman-
tic segmentation. In: Proceedings of the IEEE conference
on computer vision and pattern recognition, pp. 580–587
(2014)
7. Gupta, A., Vedaldi, A., Zisserman, A.: Synthetic data
for text localisation in natural images. In: Proceedings of
the IEEE conference on computer vision and pattern
recognition, pp. 2315–2324 (2016)
8. He, W., Zhang, X.Y., Yin, F., Liu, C.L.: Deep direct re-
gression for multi-oriented scene text detection. arXiv
preprint arXiv:1703.08289 (2017)
9. Ioffe, S., Szegedy, C.: Batch normalization: Accelerat-
ing deep network training by reducing internal covariate
shift. arXiv:1502.03167 (2015). URL http://arxiv.org/
abs/1502.03167
10. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long,
J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Con-
volutional architecture for fast feature embedding. In: Pro-
cedings of the 22nd ACM international conference
on Multimedia, pp. 675–678. ACM (2014)
11. Jiang, Y., Zhu, X., Wang, X., Yang, S., Li, W., Wang,
H., Fu, P., Luo, Z.: R2cnn: rotational region cnn for
orientation robust scene text detection. arXiv preprint
arXiv:1706.09579 (2017)
12. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet
classification with deep convolutional neural networks.
Communications of the ACM 60(6), 84–90 (2017)
13. Law, H., Deng, J.: Cornernet: Detecting objects as paired
keypoints. In: Proceedings of the European Conference
on Computer Vision (ECCV), pp. 734–750 (2018)
14. LeCun, Y., et al.: Lenet-5, convolutional neural net-
works. URL: http://yann. lecun. com/exdb/lenet
15. Li, Z., Zhou, F.: Fsd: feature fusion single shot multibox
detector. arXiv preprint arXiv:1712.00960 (2017)
16. Liao, M., Shi, B., Bai, X.: Textboxes++: A single-shot
oriented scene text detector. IEEE transactions on image
processing 27(8), 3676–3690 (2018)
17. Liao, M., Shi, B., Bai, X.: Textboxes++: A single-shot
oriented scene text detector. IEEE Transactions on Image
Processing 27(8), 3676–3690 (2018)
18. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B.,
Belongie, S.: Feature pyramid networks for object detec-
tion. In: Proceedings of the IEEE conference on computer
vision and pattern recognition, pp. 2117–2125 (2017)
19. Liu, L., Pan, Z., Lei, B.: Learning a rotation invariant
detector with rotatable bounding box. arXiv preprint
arXiv:1711.09405 (2017)
20. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S.,
Fu, C.Y., Berg, A.C.: SSD: Single Shot MultiBox Detec-
tor. In: Proc. European Conference on Computer Vision,
pp. 21–37 (2016)
21. Liu, Y., Jin, L.: Deep matching prior network: Toward
tighter multi-oriented text detection. In: Proceedings of
the IEEE Conference on Computer Vision and Pattern
Recognition, pp. 1962–1969 (2017)
22. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional
networks for semantic segmentation. In: Proceedings of
the IEEE Conference on Computer Vision and Pattern
Recognition, pp. 3431–3440 (2015)
23. Lyu, P., Yao, C., Wu, W., Yan, S., Bai, X.: Multi-oriented
scene text detection via corner localization and region
segmentation. In: Proceedings of the IEEE conference
on computer vision and pattern recognition, pp. 7553–7563
(2018)
24. Ma, J., Shao, W., Ye, H., Wang, L., Wang, H., Zheng,
Y., Xue, X.: Arbitrary-oriented scene text detection via
rotation proposals. IEEE Transactions on Multimedia
20(11), 3111–3122 (2018)
25. Qin, S., Manduchi, R.: Cascaded segmentation-detection networks for word-level text spotting. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 1, pp. 1275–1282. IEEE (2017)

26. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779–788 (2016)

27. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp. 7263–7271 (2017)

28. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems, pp. 91–99 (2015)

29. Shi, B., Bai, X., Belongie, S.: Detecting oriented text in natural images by linking segments. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2550–2558 (2017)

30. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

31. Uijlings, J.R., Van De Sande, K.E., Gevers, T., Smeulders, A.W.: Selective search for object recognition. International journal of computer vision 104(2), 154–171 (2013)

32. Yang, Q., Cheng, M., Zhou, W., Chen, Y., Qiu, M., Lin, W.: Inceptext: a new inception-text module with deformable psroi pooling for multi-oriented scene text detection. arXiv preprint arXiv:1805.01167 (2018)

33. Yang, X., Fu, K., Sun, H., Yang, J., Guo, Z., Yan, M., Zhan, T., Xian, S.: R2cnn++: Multi-dimensional attention based rotation invariant detector with robust anchor strategy. arXiv preprint arXiv:1811.07126 (2018)

34. Zhou, X., Wang, D., Krähenbühl, P.: Objects as points. arXiv preprint arXiv:1904.07850 (2019)

35. Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W., Liang, J.: East: an efficient and accurate scene text detector. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 5551–5560 (2017)