DC-SPP-YOLO: Dense Connection and Spatial Pyramid Pooling Based YOLO for Object Detection

Zhanchao Huang,　Jianlin Wang

College of Information Science and Technology, Beijing University of Chemical Technology,

Beijing 100029, China

Abstract： Although YOLOv2 approach is extremely fast on object detection; its backbone network has the low ability on feature extraction and fails to make full use of multi-scale local region features, which restricts the improvement of object detection accuracy. Therefore, this paper proposed a DC-SPP-YOLO (Dense Connection and Spatial Pyramid Pooling Based YOLO) approach for ameliorating the object detection accuracy of YOLOv2. Specifically, the dense connection of convolution layers is employed in the backbone network of YOLOv2 to strengthen the feature extraction and alleviate the vanishing-gradient problem. Moreover, an improved spatial pyramid pooling is introduced to pool and concatenate the multi-scale local region features, so that the network can learn the object features more comprehensively. The DC-SPP-YOLO model is established and trained based on a new loss function composed of mean square error and cross entropy, and the object detection is realized. Experiments demonstrate that the mAP (mean Average Precision) of DC-SPP-YOLO proposed on PASCAL VOC datasets and UA-DETRAC datasets is higher than that of YOLOv2; the object detection accuracy of DC-SPP-YOLO is superior to YOLOv2 by strengthening feature extraction and using the multi-scale local region features.

Keywords： Object detection, Convolutional neural network，YOLOv2，Dense connection, Spatial pyramid pooling

1.Introduction

The object detection approaches based on computer vision have been widely used in security monitoring, automatic driving, medical diagnosis, and other fields.

Early vision-based object detection approaches, which had low detection accuracy and narrow application range, rely on object features such as edges, key points or templates. In this regard, Haar-like features, HOG (Histogram of Oriented Gradient), LBP (Local Binary Patterns) and other feature extraction approaches with better object expression ability were proposed and used for object detection task together with machine learning approaches [1]. In 2007, Felzenszwalb et al. [2] introduced the DPM (Deformable Parts Models) approach creatively, which got a higher detection accuracy than other approaches at that time by a new object detection pipeline based on handcrafted futures and machine learning. After that, a variety of object detection approaches based on handcrafted features and machine learning were proposed one after another and performed well in successive PASCAL VOC object detection challenges [3] [4]. However, most of these methods scanned through the entire image to detect the object regions by a sliding window, which had

low detection efficiency. Also, the accuracy of object detection that was restricted by the expression ability of handcrafted futures was challenging to improve further.

With the improvement of computing performance and the abundance of data resources, the AlexNet approach based on CNN (Convolutional Neural Network) was proposed by Krizhevsky et al. [5] in 2012, which proved that using the features extracted by CNN could classify the object images more accuracy than using the handcrafted futures. The AlexNet provided a new idea for the object detection task that was regarded as the extension of image classification [6]. In 2014, the R-CNN proposed by Girshick et al. [7] employed CNN to extract features in object detection task for the first time and got the object detection accuracy which was superior to the state-of-the-art approaches at that time.

Since then, the object detection approaches based on deep learning have gradually replaced the object detection approaches based on handcraft features and machine learning, and have become a research hotspot in this field. Generally, there are two main categories of CNN-based object detection approaches: the object proposal-based approaches and the regression-based approaches [8].

The object proposal-based approaches are mostly improved and developed from the R-CNN. For the problem of slow detection speed of R-CNN, Fast R-CNN [9] and Faster R-CNN [10], which adopted Selective Search and RPN (Regional Proposal Network) instead of sliding window search respectively, were proposed to simplify the region proposal generation and improve the object detection speed. In 2016, Dai et al. [11] proposed an R-FCN (Region-based Fully Convolutional Networks) to solve the problem that the ROI-wise subnetwork of Faster R-CNN did not share calculations in different region proposals. In the past two years, based on the Faster R-CNN and R-FCN, RRPN (Rotation Region Proposal Networks) [12], R-FCN-3000 [13] and other object proposal-based approaches [14] [15] of which the detection accuracy was further improved were presented. However, the frameworks of proposal-based approaches that had two stages, the region proposal generation and the subsequent feature resampling, were much more complex in comparison with the regression-based approaches; which resulted in low speed and difficulty in real-time performance.

In 2016, Redmon et al. [16] presented for the first time a regression-based approach, YOLO (You Only Look Once), for object detection where a single convolutional network that simultaneously predicted bounding box coordinates and class probabilities was trained end-to-end directly. Even if YOLO opened the door to achieve real-time object detection, it was difficult to detect small-sized objects in the image, and the error of bounding box coordinates was large. In this regard, Liu et al. [17] proposed an SSD (Single Shot Multi-Box Detector) that introduced reference boxes and detected the object on multi-scale feature maps to improve the accuracy of object detection. In 2017, Redmon and Farhadi [18] proposed the YOLOv2 approach, and its accuracy and speed of object detection were significantly ameliorated compared with the YOLO approach; however, this method still used the Darknet19 with low ability on feature extraction as the backbone

network and did not fully utilize the multi-scale local region features of the object, which constrained the further improvement of detection accuracy. Subsequently, the deep residual network was employed as the backbone network to get the detection accuracy that was further superior to state-of-the-art approaches in DSSD (Deconvolutional Single Shot Detector) [19] and YOLOv3 [20]; on the other hand, the detection speed of these approaches was severely degraded due to the excessive number of network layers. In 2018, Zhou et al. [21] introduced DenseNet-169 [22], a dense convolutional network with better performance than the deep residual network, as the backbone network of SSD and proposed an STDN (Scale-Transferrable Detection Network) approach which achieved the detection accuracy close to the DSSD while improving the detection speed. Also, Jeong et al. [23], Lee et al. [24], Cao et al. [25] and Zheng et al. [26] proposed other improved SSD approaches for object detection, but the existing research on the improvements of YOLO series approaches are still less.

Therefore, we propose a Dense Connection and Spatial Pyramid Pooling Based YOLO object detection approach for improving YOLOv2 by optimizing the connection structure of the backbone network and introducing the multi-scale local region feature extraction. This approach is more accurate than YOLOv2 while keeping the detection speed close to YOLOv2, higher than DSSD, YOLOv3, and STDN. The main contributions of this paper are as follows:

(1) We employ the dense connection structure of the convolutional layers to improve the backbone network of YOLOv2 for strengthening feature extraction and ensuring maximum information flow between layers in the network.

(2) An improved spatial pyramid pooling is introduced to collect and concatenate the local region features on different scales in the same convolutional layer for learning multi-scale object features more comprehensively.

(3) The improvements above are introduced in YOLOv2, and the cross-entropy which can effectively alleviate the vanishing-gradient problem is utilized instead of the mean squared error to represent object classification loss; a DC-SPP-YOLO approach is presented for ameliorating the detection accuracy with a fast detection speed.

This paper is organized as the following. Section 2 gives a brief review of the related works. In Section 3, we explain the proposed approaches in detail. Section 4 presents a series of experimental results and discussion. Finally, we make conclusions in Section 5.

2. Related Works

In recent years, the researches of object detection mainly focus on improving the structure of the backbone network and detecting the object on different scales.

2.1. Backbone network

As the feature extractor, the backbone network plays a significant role in object detection. The performance of the backbone network is directly related to the accuracy and speed of object detection.

The VGG network proposed by Simonyan and Zisserman [27] in 2014 employed a stack of

small convolutional kernels instead of a single large convolutional kernel to deepen the network, which got higher accuracy and were adopted as the backbone network by many popular object detection approaches such as Faster R-CNN and SSD. Natheless, as the number of convolutional layers increased, the VGG network whose convolutional layers were connected layer by layer would cause the vanishing-gradient problem, which restricted the further improvement of detection accuracy.

In 2015, Highway Network [28] and ResNet [14] adopted the "Skip-connection" idea to deepen the convolutional neural network further while alleviating the vanishing-gradient problem; subsequently, as the backbone network, ResNet was adopted by Faster R-CN, R-FCN, DSSD, and other approaches. Compared with the approaches using VGG network as the backbone network, these approaches with better feature extraction improved the detection accuracy significantly, while the detection speed was severely degraded because of the extreme deep network.

In 2017, Huang et al. [22] presented the DenseNet with the dense connection structure of convolutional layers, which was faster and more accurate than ResNet on the image recognition task while alleviating the vanishing-gradient problem further. The STDN proposed by Zhou et al. [21] in 2018 used DenseNet-169 as the backbone network, and its object detection speed was significantly improved compared with the DSSD using the ResNet-101 as the backbone network.

The "Multi-path" employed by the Inception series network [29] [30] [31] [32] and the Xception network [33] was also one of the main ideas for backbone network design and improvement. In 2017, Li et al. [34] adopted the improved Xception network as the backbone network in the proposed Light-head R-CNN object detection approach and got better performance than YOLO and SSD on the COCO datasets.

Convolutional neural networks such as MobileNet [35], SqueezeNet [36], ShuffleNet [37] reduced parameters by compressing the network for higher speed. Even if the object detection approaches using compressed networks were less accurate than the approaches using larger backbone networks like ResNet and Inception, the detection speed of which was significantly increased, and object detection tasks on mobile terminals widely adopted them.

At present, using the advanced backbone networks based on "Skip-connection" or "Multi-path" instead of the VGG networks has become one of the main improvements on object detection tasks, where the accuracy is ameliorated by strengthening the feature extraction and reusing the object features, but the detection speed also decreases. The DenseNet has the advantages of alleviating the vanishing-gradient problem and reusing the object features so that the STDN using the DenseNet can improve the detection accuracy while maintaining a fast detection speed; however, the detection speed of STDN is still lower than that of YOLOv2. Therefore, adopting the dense connection will effectively improve the detection accuracy and speed of YOLOv2.

2.2. Multi-scale Detection

Multi-scale detection is one of the significant research contents on CNN-based object detection. In recent years, a variety of multi-scale object detection methods have been proposed, which are

mainly divided into two categories: independent detection on multiple feature maps extracted by different layers of the networks, and fusing multiple feature maps extracted by different layers of the networks.

The method of independent detection on multiple feature maps was first adopted in the SSD proposed by Liu et al. [17], which was demonstrated better for detecting small objects than detecting objects on the feature map extracted by coarser top layers of the network. In 2016, Cai et al. [38] improved the Faster R-CNN and detected objects on multi-scale feature maps, for which the receptive fields could adapt to multi-scale objects; this method got a good performance in the scene like automatic driving where the scales of the object changed considerably. Yang et al. [39] proposed an SDP (Scale Dependent Pooling) method, which pooled features from different convolutional feature maps according to the size of each proposal. In 2018, Li et al. [40] used the scale-aware mechanism to weight and combine the prediction results of Large-size Sub-network and Small-size Sub-network according to the size of input proposal, which got the state-of-the-art performance on pedestrian detection.

The method of fusing multiple feature map improves the accuracy of multi-scale detection by fusing information from different scale feature maps and different receptive fields. In 2014, the SPP (Spatial Pyramid Pooling Network) method presented by He K et al. [41] pooled arbitrary size feature maps into fixed-size feature vectors, for which the CNN model not only didn't need to fix the size of the input images but also became robust for detecting multi-scale objects by fusing the multi-scale features. In 2017, Chen et al. [42], Fu et al. [19] and Jeong et al. [23] proposed different methods of fusing the multi-scale feature maps to improve SSD and got better performances than SSD, for which the finer layers of the networks could utilize the contextual information learned from the coarser layers of the networks. Lin et al. [43] took one step ahead and proposed the FPN (Feature Pyramid Network) method, in which a top-down lateral connection structure was designed based on the multi-scale pyramid structure inherent in deep convolutional neural networks, for increasing the accuracy of multi-scale detection.

The multi-scale detection methods above detect objects independently on different feature maps or detect objects on multi-scale feature maps fused by utilizing the global features from different convolutional layers of the networks to improve the detection accuracy. However, these methods do not make full use of the local region features on different scales from the same convolutional layer, and it is still difficult to accurately detect small objects with rich local region features.

3. Dense Connection and Spatial Pyramid Pooling Based YOLO

In this paper, a DC-SPP-YOLO object detection approach is proposed. This approach employs dense connection to improve the backbone network of YOLOv2, introduces an improved spatial pyramid pool to extract the multi-scale local region features of the objects, utilizes the cross-entropy to represent the classification loss and obtains a new loss function, constructs and trains the model to detect the objects.

3.1. YOLOv2 Approach

The YOLOv2 object detection approach divides the input image into $S\times S$ grids; each grid predicts K bounding boxes, the confidence $\text{Pr(Object)}*\text{IoU}_{\text{pred}}^{\text{truth}}$ that bounding box contains objects and the conditional probabilities $\text{Pr(Class}_i|\text{Object)}$ that objects belong to C classes; where the $\text{IoU}_{\text{pred}}^{\text{truth}}$ is the Intersection-over-Union between the predictions and the ground truth. So, the class-specific confidence of each bounding box is

$$\text{Pr(Class}_i|\text{Object)}*\text{Pr(Object)}*\text{IoU}_{\text{pred}}^{\text{truth}} = \text{Pr(Class}_i)*\text{IoU}_{\text{pred}}^{\text{truth}} \tag{1}$$

The Eq. (1) represents the degree of coincidence between the predicted box and the ground truth and the probability that the object belongs to each class. Therefore, the predictions of YOLOv2 are encoded as an $S\times S\times(K\times(5+C))$ tensor.

The backbone network of the YOLOv2 extracts the object features by the down-sampling convolutional structure that is similar to the VGG network. The input of the lth layer in the convolutional neural network is represented as x^l, the weight of the convolution kernel is w^l, the bias parameter is b^l, $*$ represents convolution, the intermediate variable is $y^l = x^{l-1}*w^l+b^l$, the activation function is $f(.)$, and the loss function is $L(.)$. When convolutional neural network forward propagates, the relationship between the lth layer and the l-1th layer is represented as

$$x^l = f\left(y^l\right) = f\left(x^{l-1}*w^l+b^l\right) \tag{2}$$

When convolutional neural network backpropagates, the gradient of the loss function is

$$\delta^{l-1} = \frac{\partial L}{\partial y^{l-1}} = \frac{\partial L}{\partial y^l}\cdot\frac{\partial y^l}{\partial y^{l-1}} = \delta^l * \text{rot}180\left(w^l\right)\odot f'(x^{l-2}*w^{l-1}+b^{l-1}) \tag{3}$$

In Eq. (3), $\text{rot}180(.)$ represents the 180° counterclockwise rotation of the weight parameter matrix, $\odot$ is the Hadamard product. As the gradient propagates layer by layer in the network, the gradient represented by the product of the derivative of the activation functions and the weight parameters will become smaller and smaller. For example, the derivative of the Sigmoid activation function is $\left|f'(y^{l-1})_{\text{Sigmoid}}\right|\leq 1/4$, the initialized weights are usually less than 1; the gradient will vanish when it backpropagates in the network. Finally, the vanishing-gradient problem appears and results in low detection accuracy.

Besides, for multi-scale detection, the "Fine-Grained Features" strategy employed in YOLOv2 focuses on fusing the global features from different layers of the network but does not fully utilize the multi-scale local region features from the same convolutional layer, which restricts the improvement of detection accuracy.

3.2. Improved Dense Connection in YOLOv2

Considering the low ability of the backbone network on feature extraction and the vanishing-gradient problem in backpropagation, we employed the dense connection structure of convolutional layers to improve the accuracy of YOLOv2 by strengthening the feature extraction ability while ensuring the maximum information flow in the network.
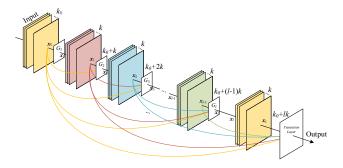
Fig. 1. The Dense Connection of the Convolutional Layers.

The dense connection structure of DC-SPP-YOLO in which the feature maps of the first l-1 layers are concatenated together and utilized as the input of the lth layer is shown in Fig. 1.

$$x^l = f(y^l) = f([x^0, x^1, \cdots, x^{l-1}] * w^l + b^l) \tag{4}$$

When convolutional neural network backpropagates, the gradient of the loss function is

$$\delta^{l-1} = \delta^l * \text{rot}180(w^l) \odot f'([x^0, x^1, \cdots, x^{l-2}] * w^{l-1} + b^{l-1}) \tag{5}$$

Compared with the derivative term $f'(x^{l-2} * w^{l-1} + b^{l-1})$ of the activation function in Eq. (3), the derivative term $f'([x^0, x^1, \cdots, x^{l-2}] * w^{l-1} + b^{l-1})$ of the activation function in equation (5) always contains the input x^0 and the output feature maps of the previous layers. Therefore, each layer of the CNN can obtain the input features and the gradient can be calculated directly from the loss function; which strengthens feature propagation in the network, alleviates the vanishing-gradient problem, and increases the detection accuracy.

Each convolutional layer of the DC block (Dense Connection block) in DC-YOLO (Dense Connection Based YOLO) outputs k concatenated feature maps; the lth layer of the DC block outputs $k_0 + k \times (l$-1) concatenated feature maps, where the number of the input feature maps x^0 is k_0. Considering that excessive increase of the densely connected convolutional layers may lead to a decrease in detection speed; only the last convolutional block which can extract the richer semantic features in the backbone network of YOLOv2 is improved to be a DC block. As shown in Fig. 2, the DC block has four dense connection units, each unit consists of a 3×3 convolutional layer and a 1×1 convolutional layer, and the increments of feature maps are set to 256, 512, 512, and 512 respectively.
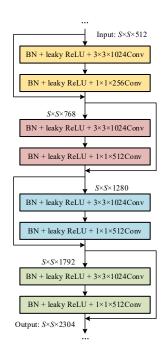
Fig. 2. The DC Block in DC-YOLO.

BN (Batch Normalization) [30] is added before the 3×3 convolutional layer of each dense unit to solve the "internal covariate shift" and alleviate the vanishing-gradient problem and speed up the model training. The leaky ReLU activation function

$$y_i = \begin{cases} x_i & \text{if } x_i \geq 0 \\ \dfrac{x_i}{a_i} & \text{if } x_i < 0 \end{cases}, \quad a_i \in (1, +\infty) \tag{6}$$

is utilized for the nonlinearization of convolution; when the input is greater than 0 ($f'(y^{l-1})_{\text{leaky ReLU}} = 1$), the vanishing-gradient problem can be alleviated; when the input is less than 0 ($f'(y^{l-1})_{\text{leaky ReLU}} > 0$), the dead neuron can be reduced compared to the ReLU activation function.

Unlike the dense units with the "Bottleneck Layers" structure in DenseNet, each dense unit of DC-YOLO first extracts the object features by a 3×3 convolution to ensure that more abundant feature maps are used to improve the quality of object features obtained, and then a 1×1 convolution is employed to reduce the number of input feature-maps. It is because that the DC block is in the deeper layer of the network, where the features extracted are more abstract than that extracted by the first few layers of the network and the receptive field of each feature is also larger, using the larger convolution can extract the richer semantic features. Nevertheless, this design of connection also leads to an increase in the number of model parameters.

Therefore, the nonlinear mapping function of each dense unit can be represented as BN-leaky ReLU-Conv(3×3)-BN-leaky ReLU-Conv(1×1). The DC block with eight convolutional layers replaces the original laminated convolutional block with four convolutional layers, increasing the number of network layers in a small amount, but improving the detection accuracy while maintaining a fast detection speed.
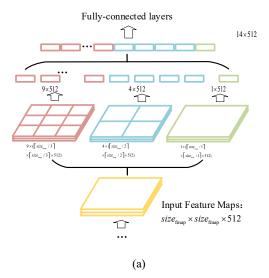
Table 1

8

The comparison of DC-YOLO and YOLOv2.

Method	BFLOP/s	mAP (%) on VOC 2007	Speed (fps)
YOLOv2	29.371	76.8	67
DC-YOLO	39.206	77.6	60.4

Table 1 uses BFLOP/s (Billion Floating Point Operations Per Second) as an evaluation index to compare the model complexity of YOLOv2 and DC-YOLO, and it also compares the detection accuracy and speed of YOLOv2 and DC-YOLO on the PASCAL VOC 2007 dataset (see Section 5 for specific experimental settings). As shown in Table 1 that the object detection accuracy of DC-YOLO is 77.6% on the PASCAL VOC 2007 dataset, which is 0.8% higher than that of the YOLOv2. Although the model complexity of DC-YOLO is increased by 9.835 BFLOP/s compared to YOLOv2, the detection speed only reduces by about 6.6 fps, which means that DC-YOLO still maintains a fast detection speed.

3.3. Improved Spatial Pyramid Pooling in YOLOv2

The multi-scale prediction of YOLOv2 and YOLOv3 focuses on concatenating the global features of multi-scale convolutional layers while ignores the fusion of multi-scale local region features on the same convolutional layer. Consequently, this paper designs a new space pyramid pooling block and introduces it into YOLOv2 for pooling and concatenating the multi-scale local region features, then the global and local multi-scale features are utilized together to improve the accuracy of object detection.
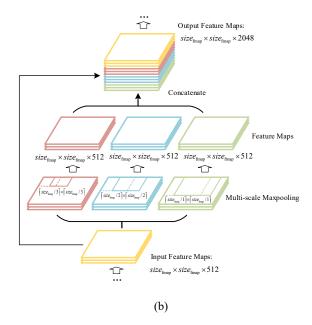

(a)

(b)

Fig. 3. The Spatial Pyramid Pooling (a) and the Improved Spatial Pyramid Pooling (b).

As shown in Fig. 3(a), the classical spatial pyramid pooling divides the input feature map into $a_i = n_i \times n_i$ bins according to the scales that represent different layers of the feature pyramid, where a_i represents the number of bins in the ith layer of the feature pyramid. The feature maps are pooled by the sliding windows of which the size is the same as that of the bins, and the $a_i \cdot d$ - dimensional feature vector where d is the number of filters is obtained to be the input of the fully connected layer.

Our new spatial pyramid pooling block with three max-pooling layers illustrated in Fig. 3(b) is introduced between the DC block and the object detection layer in the network. The 1×1 convolution is utilized to reduce the number of input feature maps from 1024 to 512. After that, the feature maps are pooled in different scales; $size_{pool} \times size_{pool}$ represents the size of the sliding windows, $size_{fmap} \times size_{fmap}$ represents the size of the feature maps, then

$$size_{pool} = \lceil size_{fmap} / n_i \rceil \tag{7}$$

We let n_i = 1, 2, 3 and pool the feature maps by the different sliding windows of which the sizes are $\lceil size_{fmap} / 3 \rceil \times \lceil size_{fmap} / 3 \rceil$, $\lceil size_{fmap} / 2 \rceil \times \lceil size_{fmap} / 2 \rceil$ and $\lceil size_{fmap} / 1 \rceil \times \lceil size_{fmap} / 1 \rceil$ respectively. The stride of pooling is all 1, and the padding is utilized to keep a constant size of the output feature maps, then we get three feature maps with the sizes of $size_{fmap} \times size_{fmap} \times 512$.

Different from the traditional spatial pyramid pooling presented by He K et al. [41], our new SPP block (Spatial Pyramid Pooling block) in SPP-YOLO (Spatial Pyramid Pooling Based YOLO) does not resize the feature maps into feature vectors with the fixed size. Instead of that, we concatenate the three feature maps pooled with the sizes of $size_{fmap} \times size_{fmap} \times 512$ and the input feature maps of the SPP block so that we would get $size_{fmap} \times size_{fmap} \times 2048$ feature maps which extract and converges the multi-scale local region features as the output for object detection.

Table 2

The comparison of SPP-YOLO and YOLOv2.

10

Method	BFLOP/s	mAP (%) on VOC 2007	Speed (fps)
YOLOv2	29.371	76.8	67
SPP-YOLO	29.746	77.5	65.2

Table 2 compares the model complexity of YOLOv2 and SPP-YOLO, and it also compares the detection accuracy and speed of YOLOv2 and DC-YOLO on the PASCAL VOC 2007 dataset (see Section 5 for specific experimental settings). As shown in Table 2 that the object detection accuracy of DC-YOLO is 77.5% on the PASCAL VOC 2007 dataset, which is 0.7% higher than that of the YOLOv2. Although the model complexity of DC-YOLO is increased by 0.375 BFLOP/s compared to YOLOv2, the detection speed only reduces by about 1.8 fps, which means that SPP-YOLO improves detection accuracy without significantly increasing the model complexity and reducing the detection speed.

3.4. Dense Connection and Spatial Pyramid Pooling Based YOLO

（1）DC-SPP-YOLO Model

The network of DC-SPP-YOLO consisting of five laminated convolution-pooling blocks, a dense connection block with four dense units, a spatial pyramid pooling block with three max-pooling layers and a multi-scale object detection block is shown in Figure 4.

Firstly, the five laminated convolution-pooling blocks decrease the features maps' size to 1/32 of the input image's size and increase the number of features maps to 512 by extracting and gathering the image features. After that, the DC block with four dense units composed by 3×3 and 1×1 densely connected convolutional layers, in which the increments of feature maps are set to 256, 512, 512, and 512 respectively, strengthens the feature extraction and outputs 2304 concatenated feature maps; then the number of output feature maps is reduced by 3×3×1024 filters.

The SPP block with three max-pooling layers is introduced after the DC block for concatenating the local region features extracted and converged by multi-scale pooling. The 1×1 convolution is adopted before the pooling to reduce the number of input feature maps from 1024 to 512. After that, feature maps are pooled by the sliding windows of which the sizes are $\left\lceil size_{\text{fmap}} / 3 \right\rceil \times \left\lceil size_{\text{fmap}} / 3 \right\rceil$, $\left\lceil size_{\text{fmap}} / 2 \right\rceil \times \left\lceil size_{\text{fmap}} / 2 \right\rceil$ and $\left\lceil size_{\text{fmap}} / 1 \right\rceil \times \left\lceil size_{\text{fmap}} / 1 \right\rceil$ respectively, then we concatenate the feature maps pooled and the input feature maps of the SPP block to get $size_{\text{fmap}} \times size_{\text{fmap}} \times 2048$ feature maps as the outputs of the SPP block.
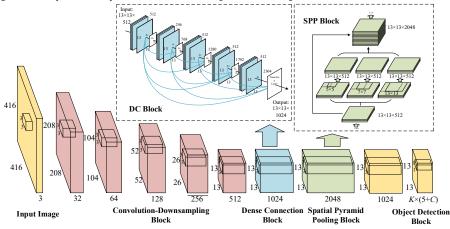
The last part of the network is object detection block, in which the output feature maps of DC block with higher resolution are reconstructed and concatenated with the output feature maps of SPP block with lower resolution. Then the feature maps above are convoluted by the $1\times1\times(K\times(5+C))$ convolution to obtain $S\times S\times(K\times(5+C))$ feature maps for object detection. Table 3 shows the parameter settings of the DC-SPP-YOLO network and the output of each layer when the size of the input image is $416\times416\times3$.

Table 3

The network's parameters of DC-SPP-YOLO

Layers	Parameters		Output	Layers	Parameters		Output
	Filters	Size / Stride			Filters	Size / Stride	
Conv 1	32	3×3/ 1	416×416×32	**DC Block**	**1024**	**3×3/1** **×4**	**13×13×2304**
Maxpool 1		2×2 / 2	208×208×32	**Conv 14-21**	**256 or 512**	**1×1 / 1**	
Conv 2	64	3×3 / 1	208×208×64	**Conv 22**	**1024**	**3×3 / 1**	**13×13×1024**
Maxpool 2		2×2 / 2	104×104×64	**Conv 23**	**512**	**1×1 / 1**	**13×13×512**
Conv 3	128	3×3 / 1	104×104×128	**SPP Block**		**5×5 / 1**	
Conv 4	64	1×1 / 1	104×104×64	**Maxpool 6-8**		**7×7 / 1** **Concat**	**13×13×2048**
Conv 5	128	3×3 / 1	104×104×128			**13×13 / 1**	
Maxpool 3		2×2 / 2	52×52×128	**Conv 26**	**512**	**1×1 / 1**	**13×13×512**
Conv 6	256	3×3 / 1	52×52×256	Conv 27	1024	3×3 / 1	13×13×1024
Conv 7	128	1×1 / 1	52×52×128	Reorg Conv13		/ 2	13×13×256
Conv 8	256	3×3 / 1	52×52×256	Concat -1, -2			13×13×1280
Maxpool 4		2×2 / 2	26×26×256	Conv 30	1024	3×3 / 1	13×13×1024
Conv 9-12	512	3×3 / 1 ×2 1×1 / 1	Conv 31	Conv31	$K*5+C$	1×1 / 1	13×13×$(K*5+C)$
Conv 13	512	3×3 / 1	26×26×512	Detection			
Maxpool 5		2×2 / 2	13×13×512				

（2）Loss Function

The predictions of DC-SPP-YOLO for each bounding box can be represented as $\boldsymbol{b} = [b_x, b_y, b_w, b_h, b_c]^T$; where (b_x, b_y) is the center coordinates of the box, b_w and b_h are the width and height of the box, b_c is the confidence. The offsets t_x, t_y from the top-left corner of the image to the grid center in b_x, b_y and the confidence b_c are constrained to $[0, 1]$ by the sigmoid function. Similarly, the ground truth of the bounding box can be represented as $\boldsymbol{g} = [g_x, g_y, g_w, g_h, g_c]^T$. The classification result of each bounding box is $\boldsymbol{Class} = [Class_1, Class_2, …, Calss_C]^T$, then the ground truth of the classification is $\Pr(Class_l)_{l \in C}$, and the predicted probability that the object belongs to the l class is $\widehat{\Pr}(Class_l)_{l \in C}$.

In this paper, a new loss function is constructed for the CNN model training, which adopts the mean squared error of the coordinate regression and the cross-entropy of object classification to represent the loss of object detection. Compared with only using the mean squared error to represent both of the coordinate regression loss and the object classification loss in YOLOv2, using the cross-entropy to represent the object classification loss can alleviate the vanishing-

gradient problem and make the model training robust. The new loss function constructed is shown in Eq. 8.

$$\begin{aligned}
L(\boldsymbol{b}, \boldsymbol{Class}) = &\ \lambda_{\text{noobj}} \sum_{i=0}^{S} \sum_{j=0}^{S} \sum_{k=0}^{K} 1_{ijk}^{\text{noobj}} \cdot \left((g_{c_{ij}} - b_{c_{ijk}}) \cdot \nabla_{\sigma}(b_{c_{ijk}})\right)^2 \\
&+ \lambda_{\text{obj}} \sum_{i=0}^{S} \sum_{j=0}^{S} \sum_{k=0}^{K} 1_{ijk}^{\text{obj}} \cdot \left((g_{c_{ij}} - b_{c_{ijk}}) \cdot \nabla_{\sigma}(b_{c_{ijk}})\right)^2 \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S} \sum_{j=0}^{S} \sum_{k=0}^{K} 1_{ijk}^{\text{obj}} \cdot \left(((g_{x_{ij}} - b_{x_{ijk}}) \cdot \nabla_{\sigma}(b_{x_{ijk}}))^2\right. \\
&\qquad\qquad\qquad + ((g_{y_{ij}} - b_{y_{ijk}}) \cdot \nabla_{\sigma}(b_{y_{ijk}}))^2 \\
&\qquad\qquad\qquad \left. + (g_{w_{ij}} - b_{w_{ijk}})^2 + (g_{h_{ij}} - b_{h_{ijk}})^2 \right) \\
&+ \lambda_{\text{class}} \sum_{i=0}^{S} \sum_{j=0}^{S} \sum_{k=0}^{K} 1_{ijk}^{\text{obj}} \sum_{l=1}^{C} (-\Pr_{ij}(Class_l) \log(\widehat{\Pr}_{ijk}(Class_l))) \\
&+ \lambda_{\text{prior}} \sum_{i=0}^{S} \sum_{j=0}^{S} \sum_{k=0}^{K} 1_{ijk}^{\text{prior}} \cdot \left(((prior_{x_{ijk}} - b_{x_{ijk}}) \cdot \nabla_{\sigma}(b_{x_{ijk}}))^2\right. \\
&\qquad\qquad\qquad + ((prior_{y_{ijk}} - b_{y_{ijk}}) \cdot \nabla_{\sigma}(b_{y_{ijk}}))^2 \\
&\qquad\qquad\qquad \left. + (prior_{w_{ijk}} - b_{w_{ijk}})^2 + (prior_{h_{ijk}} - b_{h_{ijk}})^2 \right)
\end{aligned} \tag{8}$$

In the loss function above, if the maximum of $\text{IoU}_{\text{pred}}^{\text{truth}}$ is greater than the threshold $\text{IoU}_{\text{thres}}$, $1_{ijk}^{\text{obj}} = 1$, $1_{ijk}^{\text{noobj}} = 0$; otherwise $1_{ijk}^{\text{obj}} = 0$, $1_{ijk}^{\text{noobj}} = 1$. The gradient of the sigmoid function is $\nabla_{\sigma}(.)$. Since only the maximum of $\text{IoU}_{\text{pred}}^{\text{truth}}$ is taken as the prediction result of each grid among the K anchor boxes, we calculate the loss between these bounding boxes above and those bounding boxes which do not provide useful predictions to improve the stability of model training. When the number of trained samples is less than N_{prior}, $1_{ijk}^{\text{prior}} = 1$, the predictions of the prior box can be represented as $\boldsymbol{Prior} = [Prior_x, Prior_y, Prior_w, Prior_h]^{\text{T}}$; otherwise, $1_{ijk}^{\text{prior}} = 0$. Besides, the hyperparameters λ_{noobj}, λ_{obj}, λ_{coord}, λ_{class} and λ_{prior} are the weight coefficients on each part of the loss function respectively.

4. Object Detection Using DC-SPP-YOLO

The object detection process based on DC-SPP-YOLO which includes dataset construction, model training, and object detection is shown in Fig. 5.
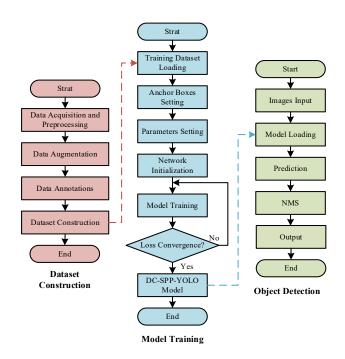
Fig. 5. The object detection process based on DC-SPP-YOLO.

Firstly, data augmentation methods such as random crop, scale augmentation, PCA jittering, are used to preprocess the training images for improving object detection performance and preventing the model from over-fitting. The k-means clustering is run for anchor boxes generation instead of hand-picked priors, and the IoU (Intersection-over-Union) between the bounding boxes of training samples and clustering centroids is utilized for constructing the distance metric

$$dist_{\text{centroid}}^{\text{box}} = 1 - \text{IoU}_{\text{centroid}}^{\text{box}} \tag{9}$$

Then, the training parameters are set, the convolutional neural network is loaded, the loss function is constructed with the sum of squared errors loss on regression and the binary cross-entropy loss on classification, the weights of the model are updated iteratively to make the loss function converge, and the DC-SPP-YOLO model is obtained for object detection.

Finally, input the test samples, load the trained DC-SPP-YOLO model, and detect the object; the algorithm's flowchart of DC-SPP-YOLO for object detection is shown in Fig. 6.

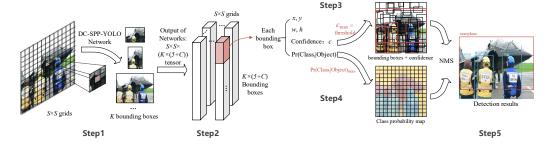**Fig. 6.** The flow chart of object detection using DC-SPP-YOLO.

Step1: Divide the input image into $S \times S$ grids; each grid generate K bounding boxes according to the anchor boxes.

Step2: Use the convolutional neural network to extract object features and predict the $\boldsymbol{b} = [b_x, b_y, b_w, b_h, b_c]^{\text{T}}$ and the $\boldsymbol{Class} = [Class_1, Class_2, \ldots, Calss_C]^{\text{T}}$.

Step3: Compare the maximum confidence $\mathrm{IoU}_{\mathrm{pred}}^{\mathrm{truth}}$ of the K bounding boxes with the threshold $\mathrm{IoU}_{\mathrm{thres}}$;

    if $\mathrm{IoU}_{\mathrm{pred}}^{\mathrm{truth}} > \mathrm{IoU}_{\mathrm{thres}}$, the bounding box contains the object;

    else, the bounding box does not contain the object.

Step4: Choose the category with the highest predicted probability as the object category.

Step5: Adopt NMS (Non-Maximum Suppression) to perform a maximum local search for suppressing redundant boxes, output and display the results of object detection.

5. Experiments

5.1. Experimental Setup and Implementation

（1）Experimental Condition

Our experiments were run on a Windows 10 PC with an Intel Xeon E5-2643 3.3GHz CPU, 32GB memory and an NVIDIA GTX 1080Ti GPU with 11.00 GB memory. The program was developed on the Visual Studio 2017 platform by C/C++ language, and the deep learning framework Darknet was used.

（2）Datasets

We demonstrated the effectiveness of DC-SPP-YOLO compared with state-of-the-art methods, especially with YOLOv2, on PASCAL VOC dataset and UA-DETRAC dataset.

The PASCAL VOC datasets used for the object detection experiments were set as follows. The experimental datasets contained 32487 images in which the objects belonged to twenty categories; the VOC 2007 trainval dataset and the VOC 2012 trainval dataset were used to train the DC-SPP-YOLO model; the VOC 2007 test dataset and the VOC 2012 test dataset were used to test the performance of DC-SPP-YOLO. As with the object detection approaches such as YOLOv2, the $\mathrm{IoU}_{\mathrm{pred\ thres}}^{\mathrm{truth}}$ was set to 0.5. The result of experiments on the PASCAL VOC 2012 test dataset was given by PASCAL VOC Challenge Evaluation Server.

The UA-DETRAC dataset used for the object detection experiments contained 82088 vehicle images taken by traffic cameras, in which the objects belonged to four categories. There were 20522 images for model training, 20522 images for validation and 41044 images for the test.

（3）Model Training

The training parameters of DC-SPP-YOLO were set as follow: the parameter a_i of leak ReLU was 10; the hyperparameters λ_{noobj}, λ_{obj}, λ_{coord}, λ_{class} and λ_{prior} of the loss function were 1, 5, 1, 1 and 0.1; the N_{prior} was 12800. The adaptive moment estimation (Adam) was adopted to update the weights of the network; the momentum was 0.9, the decay was 0.0005, the batch size was 64; the initial learning rate was 0.001, and the learning rate on the 400th epoch and the 500th epoch was reduced to 0.1 times of the original.

（4）Evaluation

The object detection accuracy was measured by mAP (mean Average Precision) when $\mathrm{IoU}_{\mathrm{thres}} = 0.5$, and the detection speed was represented by fps (frames per second).

5.2. Experiments on PASCAL VOC 2007

The object detection results of DC-SPP-YOLO on the PASCAL VOC 2007 test dataset are shown in Table 4 and Table 5; Table 4 shows the detailed experimental results of DC-SPP-YOLO on the PASCAL VOC 2007 test dataset, Table 5 Compare the detection accuracy and speed of DC-SPP-YOLO with the state-of-the-art approaches. When the size of the input image is 416×416 pixels, the DC-SPP-YOLO approach is represented as DC-SPP-YOLO 416, and other approaches are also represented as described above.

Table 4

The detection results of DC-SPP-YOLO on PASCAL VOC2007 test dataset.

Method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP (%)	Speed (fps)
DC-SPP-YOLO 416	80.0	84.9	76.0	68.0	53.8	87.6	83.9	90.1	62.5	84.1	75.8	88.6	87.3	85.7	77.0	54.3	81.7	80.1	88.3	78.7	78.4	56.3
DC-SPP-YOLO 544	83.1	85.9	77.2	69.5	59.7	88.5	86.3	89.9	62.6	86.0	78.3	87.6	88.0	86.7	80.1	54.3	81.3	80.4	87.6	79.4	79.6	38.9

Table 5

The comparison of accuracy and speed on PASCAL VOC2007 test dataset

Method	Year	Base network	mAP (%)	Speed (fps)	GPU
Faster RCNN[10]	2015	VGG16	73.2	7	Titan X
Faster RCNN[14]	2016	ResNet-101	76.4	2.4	K4
SSD 300[17]	2016	VGG16	77.5	46	Titan X
SSD 512[17]	2016	VGG16	79.5	19	Titan X
DSSD 321[19]	2017	ResNet-101	78.6	9.5	Titan X
DSSD 513[19]	2017	ResNet-101	81.5	5.5	Titan X
STDN 300[21]	2018	DenseNet-169	78.1	41.5	Titan Xp
STDN 513[21]	2018	DenseNet-169	80.9	28.6	Titan Xp
YOLO[16]	2016	Darknet19	63.4	45	Titan X
YOLOv2 416[18]	2017	Darknet19	76.8	67	Titan X
YOLOv2 544[18]	2017	Darknet19	78.6	40	Titan X
YOLOv3 416[20]	2018	Darknet53	79.3	39	GTX 1080 Ti
DC-SPP-YOLO 416	2018	Darknet19	78.4	56.3	GTX 1080 Ti
DC-SPP-YOLO 544	2018	Darknet19	79.6	38.9	GTX 1080 Ti

Since Redmon J and Farhadi A [20] did not give the test results of YOLOv3 on the PASCAL VOC dataset, we used their open source code from *https://pjreddie.com/darknet/yolo/* for object detection experiments on the PASCAL VOC 2007 test dataset. Then we used the experimental results of YOLOv3 as a control group for the experimental results of DC-SPP-YOLO.

As shown in Table 4 and Table 5, at 56.3 fps, the mAP of DC-SPP-YOLO 416 is 78.4%, which is 1.6% higher than that of YOLOv2 416; at 38.9 fps, the mAP of DC-SPP-YOLO 544 is 79.6%, which is 1.0% higher than that of YOLOv2 544; the accuracy improvement above only slightly decreases the detection speed. Although the mAP of YOLOv3 416 is 79.3% in our experiments, the speed of YOLOv3, which is as fast as DC-SPP-YOLO 544 but lower than DC-SPP-YOLO 416 and YOLOv2 416, has been damaged due to the larger backbone network Darknet53 with residual

connection structure.

We compare the detection performance of DC-SPP-YOLO with the state-of-the-art approaches in Table 5. Evidently, the performance of DC-SPP-YOLO is better than that of Faster R-CNN and YOLO on Pascal VOC 2007 test dataset; the DC-SPP-YOLO 544 is not only more accurate than SSD 512 and but also runs twice as fast as SSD 512. Even though the mAP of DC-SPP-YOLO 544 is 1.9% lower than that of DSSD 513, its detection speed is much faster than that of DSSD 513 (more than seven times faster); which is due to the fact that the detection speed of the DSSD approach is severely constrained by the extremely deep backbone network (ResNet-101) and the inefficient feature fusion. STDN 513 adopts the DenseNet-169 backbone network to improve the speed of DSSD 513, but STDN 513 is still about a third slower than DC-SPP-YOLO 544 even though STDN 513 has a 1.3% higher mAP than DC-SPP-YOLO 544. As shown in Fig. 7, considering both the detection accuracy and speed, the general performance of our approach is better than that of STDN.
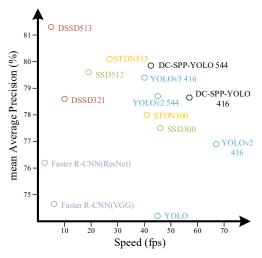

Fig. 7. The accuracy and speed on PASCAL VOC2007.

Table 6

The path from YOLOv2 to DC-SPP-YOLO.

Improvement	YOLOv2 416			DC-SPP-YOLO 416
Dense Connecting		√		√
Spatial Pyramid Pooling			√	√
mAP (%)	76.8	77.6	77.5	78.4
Speed (*fps*)	67	60.4	65.2	56.3

The comparison of improvement on each component in DC-SPP-YOLO is discussed in Table 6. The mAP of DC-YOLO 416 in which the improved dense connection of convolutional layers is employed is 0.8% higher than that of YOLOv2, the mAP of SPP-YOLO 416 in which the improved spatial pyramid pooling block is introduced is 0.7% higher than that of YOLOv2. According to the results of the experiments, two improvements above improve the mAP of DC-SPP-YOLO 416 from 76.8% to 78.4%, which further verifies the effectiveness of our methods.

5.3. Experiments on PASCAL VOC 2012

Object detection experiments of which the results are shown in Table 7 and Fig. 8 were done on the PASCAL VOC 2012 dataset for testing the performance of DC-SPP-YOLO further. At 38.9 fps, the mAP of DC-SPP-YOLO 544 is 1.2% higher than that of YOLOv2 544. The APs (Average Precisions) of 19 classes in total 20 classes predicted by DC-SPP-YOLO 544 are higher than those predicted by YOLOv2 544, which demonstrates the detection accuracy of YOLOv2 is ameliorated by the improved dense connection and the improved spatial pyramid pooling.

Table 7

The detection results of DC-SPP-YOLO on PASCAL VOC2012 test dataset.

Method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP (%)
YOLO[16]	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8	57.9
YOLOv2 544[18]	86.3	82.0	74.8	59.2	51.8	79.8	76.5	90.6	52.1	78.2	58.5	89.3	82.5	83.4	81.3	49.1	77.2	62.4	83.8	68.7	73.4
DC-SPP-YOLO 544	86.9	82.5	75.7	60.1	52.9	82.5	78.4	91.0	52.8	80.2	60.8	89.4	83.5	85.5	82.5	49.5	79.8	63.9	83.7	68.3	74.6

Fig. 8. The object detection using DC-SPP-YOLO on PASCAL VOC2012 test dataset.

As shown in Table 8, this paper compares the object detection performance of the DC-SPP-YOLO with the state-of-the-art approaches on the PASCAL VOC 2012 dataset. The experimental results on PASCAL VOC 2012 are similar to those on PASCAL VOC 2007. DC-SPP-YOLO 544 gets 74.6% mAP, which is comparable to that of SSD 512, higher than that of Faster R-CNN and that of YOLOv2, lower than that of DSSD; while DC-SPP-YOLO 544 the speed of which is 38.9 fps runs much faster than Faster R-CNN, DSSD 513, faster than SSD 512, only a little slower than YOLOv2. In general, these results above demonstrate the effectiveness of DC-SPP-YOLO both in detection accuracy and speed further.

Table 8

The comparison of accuracy and speed on PASCAL VOC2012 test dataset.

Method	Year	Base network	mAP(%)	Speed(fps)	GPU
Faster RCNN[10]	2015	VGG16	70.4	7	Titan X
Faster RCNN[14]	2016	ResNet-101	73.8	2.4	Tesla K4
SSD 300[17]	2016	VGG16	72.4	46	Titan X
SSD 512[17]	2016	VGG16	74.9	19	Titan X
DSSD 321[19]	2017	ResNet-101	76.3	9.5	Titan X
DSSD 513[19]	2017	ResNet-101	80.0	5.5	Titan X
YOLO[16]	2016	Darknet19	57.9	45	Titan X
YOLOv2 544[18]	2017	Darknet19	73.4	40	Titan X
DC-SPP-YOLO 544	2018	Darknet19	74.6	38.9	GTX 1080Ti

5.4. Experiments for Vehicles Detection on UA-DETRAC

Vehicle detection is one of the significant applications of vision-based object detection approaches. In order to verify the detection performance of our approach in the real scene, we utilized the UA-DETRAC Trainval dataset to train the DC-SPP-YOLO 416 model and the YOLOv2 416 model. The results of vehicle detection in various environmental conditions are summarized in Table 9; the DC-SPP-YOLO 416 has a 2.25% higher mAP than YOLOv2 416 at 58.3 fps, which shows that the improved dense connection and the new spatial pyramid pooling are helpful to increase the accuracy.

Table 9

The comparison of accuracy and speed on UA-DETRAC dataset.

Method	mAP(%)	Speed(fps)	GPU
GP-FRCNN[45]	91.90	4.0	Tesla K40
EB[46]	89.57	11.0	Titan X
SSDR[47]	79.47	34.0	GTX 1080
RCNN-SC[47]	93.43	2.2	2×Tesla K80
FRCNN-Res[47]	82.90	1.0	2×Titan X
DFCN[47]	86.86	11.0	Titan X
YOLOv2 416[18]	85.48	67.8	GTX 1080 Ti
DC-SPP-YOLO 416	87.73	58.7	GTX 1080Ti

Comparing with the other approaches in Table 9, the mAP of DC-SPP-YOLO 416 is 8.26% higher than SSDR, 4.83% higher than FRCNN-Res, 0.87% higher than DFCN, 1.84% lower than EB, 3.27% lower than GP-FRCN and 5.7% lower than RCNN-SC; but DC-SPP-YOLO 416 is much faster than the approaches above, even if the SSDR is about two-fifths slower than DC-SPP-YOLO 416. Besides，as shown in Fig. 9, the object detection of DC-SPP-YOLO is robust in complex scenes such as variable lighting conditions, different weather, object occlusion, object blurring.

Fig. 9. The object detection using DC-SPP-YOLO on UA-DETRAC dataset.

6. Conclusions

In this paper, a DC-SPP-YOLO object detection approach is proposed for the problem that YOLOv2 has the backbone network with low ability of feature extraction and fails to make full use of multi-scale local region features. In DC-SPP-YOLO, the improved dense connection structure of the convolutional layers is utilized for strengthening feature extraction and ensuring maximum information flow in the network. Moreover, a new spatial pyramid pooling is designed and introduced to collect and concatenate the multi-scale local region features to learn object features comprehensively. The cross-entropy is adopted instead of the mean squared error to represent the object classification loss, which alleviates the vanishing-gradient problem and accelerates the model training. The experiments on the PASCAL VOC datasets and the UA-DETRAC datasets demonstrate that DC-SPP-YOLO is more accurate than YOLOv2 and is as good as the state-of-the-art approaches on object detection tasks.

Acknowledgments

References

[1] S. Agarwal, J. O. D. Terrail, and F. Jurie, "Recent Advances in Object Detection in the Age of Deep Convolutional Neural Networks," *ArXiv180903193 Cs*, Sep. 2018.

[2] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object Detection with Discriminatively Trained Part-Based Models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.

[3] J. Zhang, K. Huang, Y. Yu, and T. Tan, "Boosted local structured HOG-LBP for object localization," in *2011 IEEE Computer Vision and Pattern Recognition (CVPR)*, Colorado Springs, CO, USA, 2011, pp. 1393–1400.

[4] J. Zhang, Y. Huang, K. Huang, Z. Wu, and T. Tan, "Data decomposition and spatial mixture modeling for part based model," in *Asian Conference on Computer Vision*, 2012, pp. 123–137.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *International Conference on Neural Information Processing Systems*, 2012, pp. 1097–1105.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, 2014, pp. 580–587.

[8] J. Han, D. Zhang, G. Cheng, N. Liu, and D. Xu, "Advanced Deep-Learning Techniques for Salient and Category-Specific Object Detection: A Survey," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 84–100, Jan. 2018.

[9] R. B. Girshick, "Fast R-CNN," in *2015 International Conference on Computer Vision (ICCV)*, Santiago, Chile, 2015, pp. 1440–1448.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[11] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object Detection via Region-based Fully Convolutional Networks," *Neural Inf. Process. Syst.*, pp. 379–387, 2016.

[12] J. Ma *et al.*, "Arbitrary-Oriented Scene Text Detection via Rotation Proposals," *IEEE Trans. Multimed.*, vol. PP, no. 99, pp. 1–1, 2017.

[13] B. Singh, H. Li, A. Sharma, and L. S. Davis, "R-FCN-3000 at 30fps: Decoupling Detection and Classification," in *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, Utah, USA, 2018, pp. 1081–1090.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 770–778.

[15] Z. Li, Y. Chen, G. Yu, and Y. Deng, "R-FCN++: Towards Accurate Region-based Fully Convolutional Networks for Object Detection," in *National Conference on Artificial Intelligence*, 2018, pp. 7073–7080.

[16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 779–788.

[17] W. Liu *et al.*, "SSD: Single Shot MultiBox Detector," in *2016 European Conference on Computer Vision (ECCV)*, Amsterdam, The Netherlands, 2016, vol. 9905, pp. 21–37.

[18] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017, pp. 6517–6525.

[19] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "DSSD : Deconvolutional Single Shot Detector," *ArXiv170106659 Cs*, Jan. 2017.

[20] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *ArXiv180402767 Cs*, Apr. 2018.

[21] P. Zhou, B. Ni, C. Geng, J. Hu, and Y. Xu, "Scale-Transferrable Object Detection," in *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, Utah, USA, 2018, pp. 528–537.

[22] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017, pp. 2261–2269.

[23] J. Jeong, H. Park, and N. Kwak, "Enhancement of SSD by concatenating feature maps for object detection," in *British Machine Vision Conference*, 2017.

[24] K. Lee, J. Choi, J. Jeong, and N. Kwak, "Residual Features and Unified Prediction Network for Single Stage Detection.," *ArXiv Prepr. ArXiv170705031*, 2017.

[25] G. Cao, X. Xie, W. Yang, Q. Liao, G. Shi, and J. Wu, "Feature-fused SSD: fast detection for small objects," *ArXiv Comput. Vis. Pattern Recognit.*, vol. 10615, Apr. 2018.

[26] L. Zheng, C. Fu, and Y. Zhao, "Extend the shallow part of single shot multibox detector via convolutional neural network," in *International Conference on Digital Image Processing*, 2018.

[27] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *Int. Conf. Learn. Represent.*, 2015.

[28] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Neural Information Processing Systems*, Montreal,Canada, 2015, pp. 2377–2385.

[29] C. Szegedy *et al.*, "Going deeper with convolutions," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2015, pp. 1–9.

[30] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[31] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 2818–2826.

[32] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," in *National Conference on Artificial Intelligence*, 2016, pp. 4278–4284.

[33] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," in *Computer Vision and Pattern Recognition*, 2017, pp. 1800–1807.

[34] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun, "Light-Head R-CNN: In Defense of Two-Stage Object Detector," *ArXiv Comput. Vis. Pattern Recognit.*, Jan. 2017.

[35] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.," *ArXiv Comput. Vis. Pattern Recognit.*, 2017.

[36] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," *ArXiv Comput. Vis. Pattern Recognit.*, Apr. 2017.

[37] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," in *Computer Vision and Pattern Recognition*, 2018, pp. 6848–6856.

[38] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A Unified Multi-scale Deep Convolutional Neural Network for Fast Object Detection," in *European Conference on Computer Vision*, 2016, pp. 354–370.

[39] F. Yang, W. Choi, and Y. Lin, "Exploit All the Layers: Fast and Accurate CNN Object Detector with Scale Dependent Pooling and Cascaded Rejection Classifiers," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2129–2137.

[40] J. Li, X. Liang, S. Shen, T. Xu, J. Feng, and S. Yan, "Scale-Aware Fast R-CNN for Pedestrian Detection," *IEEE Trans. Multimed.*, vol. 20, no. 4, pp. 985–996, Apr. 2018.

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015.

[42] Y. Chen, J. Li, B. Zhou, J. Feng, and S. Yan, "Weaving Multi-scale Context for Single Shot Detector," *ArXiv Comput. Vis. Pattern Recognit.*, Jan. 2017.

[43] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, 2017, pp. 936–944.

[44] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," in *2015 International Conference on Learning Representations (ICLR)*, 2015.

[45] S. Amin and F. Galasso, "Geometric proposals for faster R-CNN," in *Advanced Video and Signal Based Surveillance*, 2017, pp. 1–6.

[46] L. Wang, Y. Lu, H. Wang, Y. Zheng, H. Ye, and X. Xue, "Evolving boxes for fast vehicle detection," in *International Conference on Multimedia and Expo*, 2017, pp. 1135–1140.

[47] S. Lyu *et al.*, "UA-DETRAC 2017: Report of AVSS2017 & IWT4S Challenge on Advanced Traffic Monitoring," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2017, pp. 1–7.