Digital Signal Processing 102 (2020) 102756

Contents lists available at ScienceDirect

Digital Signal Processing

www.elsevier.com/locate/dsp

Faster-YOLO: An accurate and faster object detection method

Yunhua Yin [a,b,*], Huifang Li [a], Wei Fu [b],

[a] School of Electronics and Information, Northwestern Polytechnical University, XI'AN, 710129, China
[b] School of Electronics and Control Engineering, North University of China, Taiyuan, 030051, China

ARTICLE INFO

Article history:
Available online 4 May 2020

Keywords:
Object detection
ELM-LRF
ELM-AE
YOLO
Real-time processing

ABSTRACT

In the computer vision, object detection has always been considered one of the most challenging issues because it requires classifying and locating objects in the same scene. Many object detection approaches were recently proposed based on deep convolutional neural networks (DCNNs), which have been demonstrated to achieve outstanding object detection performance compared to other approaches. However, the supervised training of DCNNs mostly uses gradient-based optimization criteria, in which all parameters of hidden layers require multiple iterations, and often faces some problems such as local minima, intensive human intervention, time-consuming, etc. In this paper, we propose a new method called Faster-YOLO, which is able to perform real-time object detection. The deep random kernel convolutional extreme learning machine (DRKCELM) and double hidden layer extreme learning machine auto-encoder (DLELM-AE) joint network is used as a feature extractor for object detection, which integrating the advantages of ELM-LRF and ELM-AE. It takes the raw images directly as input and thus is suitable for the different datasets. In addition, most connection weights are randomly generated, so there are few parameter settings and training speed is faster. The experiment results on Pascal VOC dataset show that Faster-YOLO improves the detection accuracy effectively by 1.1 percentage points compared to the original YOLOv2, and an average 2X speedup compared to YOLOv3.

1. Introduction

In recent several years, deep learning has been actively applied in various fields of computer vision, like image classification [1–6], object detection [7–12], etc. As one of the most challenging issues in the computer vision, object detection has become a hot topic of recent research. Compared with object recognition tasks, object detection is more complicated and difficult. The purpose of object detection is to locate different objects in the same scene and designates labels to the bounding boxes of the objects. Two main problems need to be solved in object detection: where is the object in the image (namely location problem) and what is the object (namely category problem). Therefore, we can think of the object detector as a fusion of object location and object recognition.

In traditional object detection approaches, sliding windows [7] of different sizes was used to look for objects at different locations as candidate regions, and then a classifier decides whether there are objects and designates labels to all possible boxes in the scene. With the development of deep neural networks, especially convolutional neural network (CNN), object detection has reached an impressive improvement. R-CNN [12] has made a huge breakthrough in object detection, and has started the upsurge of object detection based on deep learning. R-CNN transforms the object detection into a classification problem very intuitively, which use CNN model for feature extraction and classification and has achieved a good detection effect. In order to overcome the limitation of repeatedly using CNN networks to extract image features in the R-CNN model, Fast R-CNN [13] has proposed a Region of Interest (RoI) pooling layer based on SPP-Net [14]. Both R-CNN and Fast R-CNN used a method called selective search (SS) to limit the number of bounding boxes, but the SS method takes a lot of time. In order to alleviate the time-consuming problem caused by the SS method, Faster R-CNN [15] directly generates potential bounding boxes to be detected through a Region Proposal Network (RPN) to save a lot of computation.

Different from the region proposal method used in two-stage detectors, The YOLO [16] (you only look once) treats the object detection task as a single regression problem which relieves the computational complexity. The core idea is to use the entire image as the input to the network, and directly return the bounding box coordinates and class probabilities in the output layer. It turns out that YOLO has a significant speed advantage over Faster R-CNN, but the localization accuracy of YOLO is significantly lower

* Corresponding author at: School of Electronics and Control Engineering, North University of China, Taiyuan, 030051, China.
E-mail addresses: yinyunhua@nuc.edu.cn, yinyunhua@mail.nwpu.edu.cn (Y. Yin).

https://doi.org/10.1016/j.dsp.2020.102756
1051-2004/

than the Faster R-CNN. SSD [17] is another end-to-end object detection method that follows the idea of direct regression bounding box and classification probability in YOLO. Unlike YOLO, SSD uses a multi-scale feature mapping strategy for prediction. Different scale feature maps are used to detect objects of different size. Additional convolution operations are performed on these layers to obtain all possible bounding boxes and categories. SSD maintains a high recognition speed and can also raise mAP to a higher level, but the detection effect on small size and blurry object is poor and recall rate for smaller objects is slow.

Motivated by region proposal network (RPN) for predicting bounding boxes in Faster R-CNN, YOLOv2 [18] makes some improvement, such as high resolution classifier, batch normalization, convolution with anchor boxes, dimension clusters and fine-grained features, etc. To further improve the speed and accuracy, YOLOv3 [19] has made some application improvements on the basis of YOLOv2, including multi-scale detection, multi-label classification and the improved feature extraction network (Darknet-53). YOLOv3 has a good balance in detection accuracy and detection speed, therefore it can be said that YOLOv3 is one of the state-of-the-art object detection method at present.

Besides, YOLOv3 adopts a new network structure (Darknet-53), which is mainly composed of convolutional layer (multiple 3×3 and 1×1 convolutional layers), Batch Normalization and shortcut connection. Although Darknet-53 network has been proved to be more efficient than ResNet [20], it is essentially a deep convolutional neural networks (DCNNs), which is too complicated and redundant. Excessive parameters can lead to complex training, increase the demand for data, and slow down the detection speed. YOLOv3 can achieve good performance on a powerful GPU, but the training process is still very time-consuming. Under the condition of embedded computing devices with limited computing power and limited memory hardware, using this method to perform real-time object detection is still very difficult. Therefore, real-time object detection system remains a big challenge to tackle.

Although DCNNs can automatically extract the distinguishing features and also has achieved better performance in object recognition, the training of CNN usually relies on traditional gradient-based learning algorithms (back-propagation algorithm), and thus CNN requires parameter iterative adjustment and suffers from trivial issues associated with BP, including slow training speed, local minima, intensive human intervention [21,22]. In order to overcome the problems caused by the BP algorithm, Huang et al. has proposed the extreme learning machine (ELM) [23,24], which is an efficient training algorithm for single-hidden layer feedforward neural network (SLFN). ELM improves the learning speed by means of randomly generating input weights and hidden biases without parameters adjustment control, which is different from traditional gradient descent-based methods. Therefore, fast learning speed and good generalization performance become outstanding advantages of ELM [25]. Especially, the ELM auto-encoder (ELMAE) [26,27] has been proposed to perform unsupervised feature learning, with which the input data could be projected to a different dimensional space. In order to extract deep higher-level feature information, various Multilayer ELM learning algorithms [27,28] has been proposed. Inspired by biological learning mechanisms, the local receptive fields based extreme learning machine (LEM-LRF) [29] has been proposed, which is implemented by introducing the local receptive field (LRF) concept similar to CNN networks. The connection between the input layer and a hidden layer node is randomly generated according to a continuous probability distribution. Comparing to traditional deep learning methods, ELM-LRF has better performance in object recognition [29–31]. Recently, some improved algorithms for ELM-LRF based on deep learning have been proposed. An effective multi-modal local receptive field extreme learning machine (MM-ELM-LRF) is proposed by Li et al. [32], which is applied to RGB-D object recognition. Qi et al. [33] proposes a hierarchical LRF-based ELM (HLRF-ELM) for a hyperspectral image (HSI) classification, which can be developed to jointly learn the feature representation and the reinforcement learning strategy for active object recognition.

In order to realize the real-time object detection on embedded devices, this paper proposes an accurate and faster object detection method called Faster-YOLO. A new DRKCELM and DLELM-AE joint network structure is proposed, which can replace Darknet-53 as a feature extractor. Different the Darknet-53, the joint network have a simple network structure and extracts the features effectively without parameter adjustment. Faster-YOLO not only simplifies the network structure, but also improves the speed of real-time target detection, and making it suitable for embedded devices with limited computing, memory, and power requirements.

The major contributions of this paper are as follows:

(1) In order to further improve the performance of object detection system, this paper introduces an accurate and faster object detection method called Faster-YOLO. Faster-YOLO is faster than the previous state-of-the-art detection algorithm.
(2) In order to improve the training speed and reduce the computational complexity, a new DRKCELM and DLELM-AE joint network is proposed, which replaces Darknet-53 as a feature extractor. The new network can not only extract more abstract and representative feature representation, but also simplify the network training process. It avoids tedious iterative fine-tuning of parameters and greatly improves training efficiency.
(3) We perform the experimental validation on generic object detection datasets (i.e. PASCAL VOC [34]). The experimental results show that the Faster-YOLO achieves rather promising performance compared to other latest approaches based on DCNNs (like Faster R-CNN, SSD, YOLOv2, YOLOv3) for object detection.

The paper is organized as follows. Section 2 provides an overview of YOLO and ELM-LRF. Section 3 describes the methodology about Faster-YOLO in detail. Section 4 is experimental results. Section 5 is conclusions.

2. Related work

2.1. YOLO

YOLO (including its improved version YOLOv2 [18], YOLOv3 [19], SSD [17], RRC [35]) is the representative of the one-stage detection methods without a distinct region proposal stage, which treats object detection task as a single regression problem. The core idea is to use the entire image as the input of the network and straight from image pixels to bounding box coordinates and class probabilities. The feature map of the YOLO output layer is designed for predictions of bounding box coordinates, the confidence score, and the class probabilities, and thus YOLO enables the detection of multiple objects in a neural network, you can end to end to realize object detection performance. Therefore, the detection speed is much faster than that of conventional methods. However, owing to the limitation of 7×7 grid, localization error is large and the detection accuracy is low. YOLO does not detect small objects well, especially dense small objects because a grid can only predict one object.

To address these problems, YOLOv2 [18] has been proposed. YOLOv2 improves the detection accuracy through the following measures: (1) Batch Normalization (BN) [36] technique. By adding batch normalization on all convolutional layers, YOLOv2 improves the detection accuracy. Besides, dropout [37] can be removed from the model without overfitting with batch normalization. (2) High

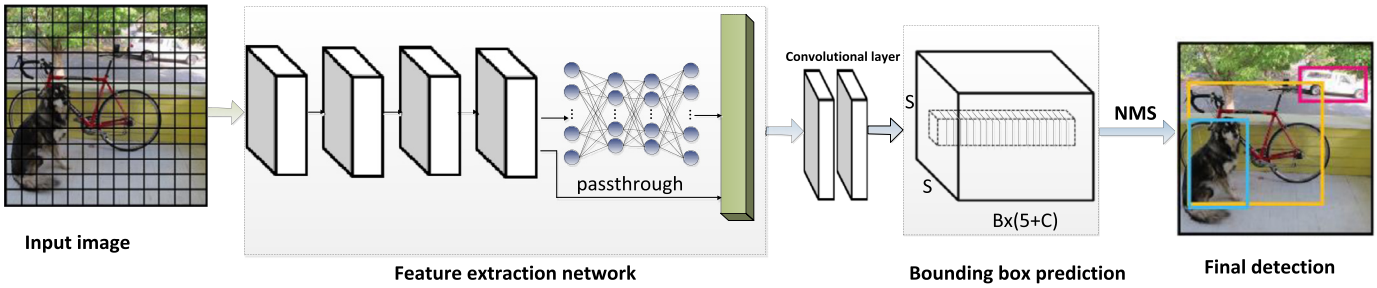


Fig. 1. Faster-YOLO pipeline. It mainly includes the following parts: Input image, Feature extraction network, Bounding box prediction and Final detection result.

Resolution Classifier. The size 224×224 of the input image in original YOLO is replaced by the size 448×448 for training the classifier network. Higher resolutions classifier makes the detector working better. (3) Convolutional with anchors boxes. The fully connected layers is removed and replaced by anchor boxes to predict the boundary boxes. The adoption of the anchor boxes makes a slight decrease in accuracy but improves the recall, and the chance to detect all the ground truth objects has been greatly increased. (4) Dimension clusters. On the training set bounding boxes, K-means clustering method is used to automatically find good priors. To alleviate errors caused by different size boxes, the IOU (Intersection Over Union) scores are used for the clustering instead of the Euclidean distance. (5) Direct location prediction. Predicting location coordinates relative to the location of the grid cell instead of the offsets to the anchors, which makes the network more stable and easier to learn. (6) Fine-grained features. For improving the capability of detecting small objects, YOLOv2 adds a pass-through layer, which combines the high resolution features with the low resolution features. (7) Multi-scale training. Multi-scale training regime helps the network to learn how to make prediction across a variety of input dimensions. This means that detections of different resolutions can be predicted using the same network.

However, the detection accuracy of YOLOv2 is still low for small or dense objects. Therefore, YOLOv3 has made some improvements on the basis of YOLOv2. (1) The multi-label classification. YOLOv3 uses independent logistic classifiers instead of softmax classifier for the multi-label class predictions. During the training phase, YOLOv3 use binary cross-entropy loss instead of the general mean square error for the class predictions. (2) Different bounding box prediction. The objectness score is set to 1 if the bounding box prior overlaps a ground truth object by more than others. If the bounding box prior overlaps a ground truth object by more than a chosen threshold, the prediction will be ignored. Therefore, YOLOv3 has only one bounding box anchor for each ground truth object. (3) Predictions across scale. YOLOv3 can predict boxes at three different scales and then extracts features from those scales using feature pyramid networks. The prediction result of the network is a 3-d tensor, which encodes bounding box, objectness score, and class predictions. (4) A new deep CNN feature extractor named Darknet-53. Darknet-53 combines ideas of other CNNs [38–40]. It is a 53 layered CNN including successive 3×3 and 1×1 convolutional layers, which uses skip connections network inspired from ResNet [40]. Experimental results have shown that YOLOv3 performs well, and has fewer floating point operations and faster speeds.

2.2. Local receptive fields based ELM (ELM-LRF)

By introducing the local receptive field (LRF) concept in neuroscience, Huang et al. [29] have proposed a new biologically inspired ELM framework. The local receptive fields based Extreme learning machine (ELM-LRF) is a special type of Extreme learning machine (ELM), and the main characteristics are the randomness of hidden neurons and combinatorial hidden nodes. The locally densely connection between the input layer and the hidden layer nodes in the architecture is randomly generated according to the continuous probability distribution. Therefore, this random connection constitutes a local receptive field. ELM-LRF extracts features from the input data through a single-layer convolutional layer and generates even more abstract representations by combinatorial nodes, which is similar to convolutional neural network (CNN). The LRF-ELM algorithm can randomly generate the input weights and directly calculate the output weighting. There is no need to use BP algorithm for adjusting the parameters of the network, so LRF-ELM is more efficient comparing the traditional CNN algorithm.

3. Faster-YOLO object detection model

In order to further simplify the network structure and decrease computational and memory requirements, this paper proposes a Faster-YOLO object detection method based on YOLOv2 and YOLOv3. This section describes the Faster-YOLO method in detail, including the specific network architecture for feature extraction and object detection, the derived loss function and our efficiency design for real-time performance.

3.1. Model architecture

Faster-YOLO inherits the characteristics of YOLO end-to-end operation and directly predicts the bounding box and object class. As shown in Fig. 1, Faster-YOLO mainly includes four parts: input image, feature extraction network, bounding box prediction and final detection result.

(1) Input image. The size of input image is 416×416. It divides the image into an $S \times S$ grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object.

(2) Feature extraction network. Using the DRKCELM and DLELM-AE joint network as a feature extractor for classification and detection.

(3) Bounding box prediction. Each grid cell predicts B bounding boxes and confidence scores for those boxes. Each bounding box is responsible for predicting four values of t_x, t_y, t_w, t_h and confidence t_o. Concerning the detected object, Each grid cell also predicts C conditional class probabilities (t_{ci}, $i = 1, 2, \ldots, C$). At last, we get a predicted tensor, which is used for regression. Faster-YOLO's final prediction is a 3-d tensor: $S \times S \times B \times (5 + C)$.

(4) Final detection result. The non-maximum suppression (NMS) algorithm is used to obtain the final object detection results.

3.2. RKCELM

In order to make full use of the advantages of CNN in feature extraction and the fast training speed of ELM algorithm based

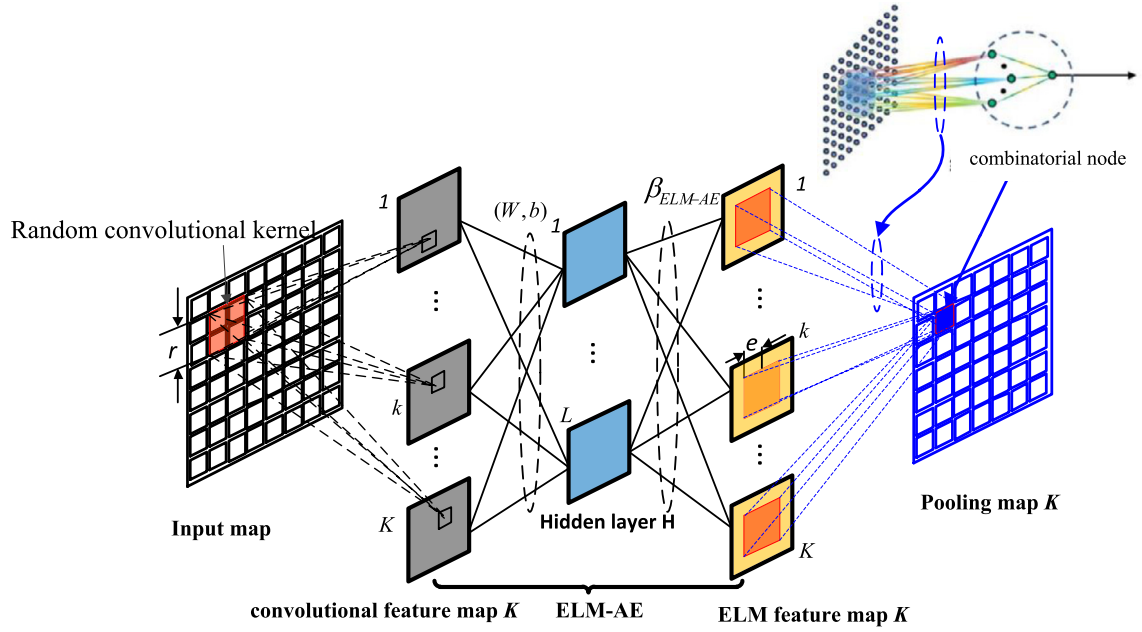


Fig. 2. The structure of the RKCELM model.

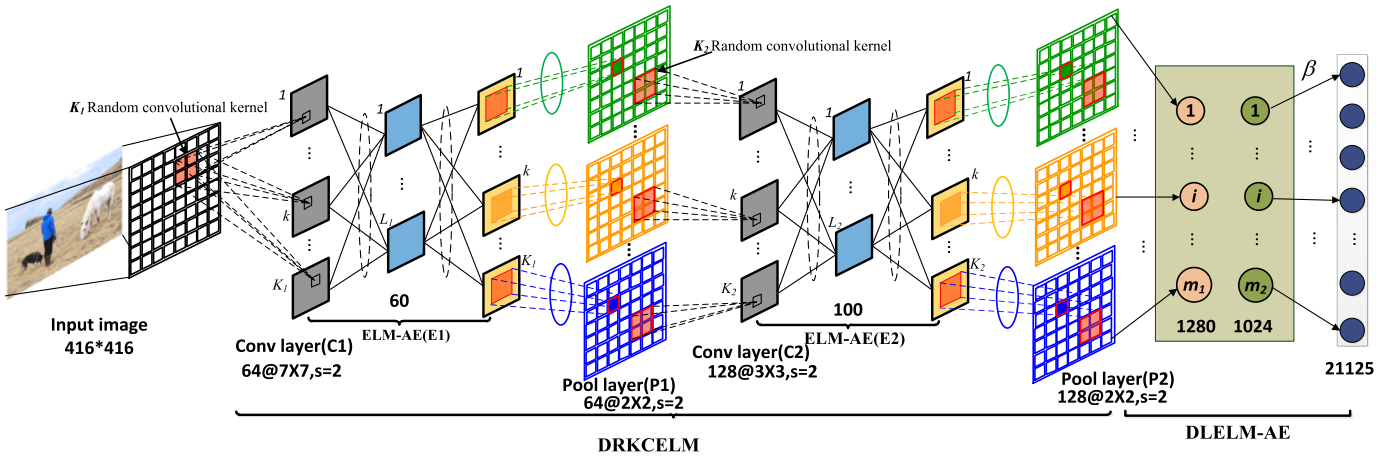


Fig. 3. The DRKCELM and DLELM-AE joint network architecture.

on ELM-LRF, this paper proposes a random kernel convolutional extreme learning machines (RKCELM) which combines ELM auto-encoder (ELM-AE) and convolutional neural network (CNN). ELM-AE [26] is an unsupervised learning method, which is helpful for the stacked network to obtain more abstract representations without losing salient information. The main idea is that combining the fast model with the complex model to give full play to their advantages.

The convolution kernel performs convolution operations on all input maps, and the shared convolution kernel parameters can be randomly generated according to the Gaussian distribution. It is equivalent to using a random convolution kernel by local connection with input images for extracting low-level feature information. Then, the ELM-AE is fully connected between the hidden layer and the convolutional feature map, and representative features are extracted by encoding and reconstruction. Finally, the square-root pooling is used in the form of a combined node, which makes the RKCELM network has translation invariance. The structure of the RKCELM model is shown in Fig. 2, which mainly includes the following parts: random convolution kernel, convolutional layer, ELM-AE layer and combinatorial pooling layer.

Therefore, RKCELM not only has the significant feature extraction ability of CNN, but also has the advantages of ELM-AE's faster learning and good generalization performance, which can be used repeatedly as an independent learning network unit to realize more complex function approximation.

3.3. Feature extractor network

Although the Darknet (DCNN used in YOLOv2 and YOLOv3) has excellent feature extraction performance, the network structure is too complicated and has too many parameters. In order to simplify the network structure, this paper replaces the Darknet with the DRKCELM and DLELM-AE joint network, as shown in Fig. 3. The joint network consists of a two layer RKCELM and a double hidden layer extreme learning machine auto-encoder (DLELM-AE). RKCELM is responsible for low-level feature extraction and DLELM-AE is responsible for high-level feature extraction. The parameters of the entire training process are randomly generated and do not need to be adjusted.

According to architecture of the DRKCELM and DLELM-AE joint network in Fig. 3, the training stage is divided into the following parts:

(1) Generate the random convolutional kernel A

For making the learned features more robust, Gaussian probability distribution is used to generate K different random convolutional kernel matrices A^{init}. The size of random convolutional kernel is $r \times r$.

$$\hat{A}_{init} \in R^{r^2 \times K}, \quad \hat{a}_k \in R^{r^2}$$
$$\hat{A}_{init} = [\hat{a}_1, \hat{a}_2, ..., \hat{a}_k]$$
$$k = 1, 2, ..., K_1 \tag{1}$$

Then, orthogonalize matrix $\hat{A}$ using singular value decomposition (SVD) method. $\hat{a}_k$ (column-wisely) are the orthonormal basis of $\hat{A}$. The purpose of orthogonalization is to make the weight distribution more broadly and evenly.

(2) convolutional layer C1

The size of input RGB image is $d \times d$, the size and the number of random convolutional kernel are $r \times r$ and K_1 respectively, the size of the feature map should be $(d - r + 1) \times (d - r + 1)$. The convolutional node (i, j) in the k-th feature map $C_{i,j,k}$ is calculated as:

$$C^{(1)}_{i,j,k}(x) = \sum_{m=1}^{r}\sum_{n=1}^{r} x_{i+m-1,j+n-1} \cdot a_{m,n,k}$$
$$i, j = 1, ..., (d - r + 1), \quad k = 1, 2, ..., k_1 \tag{2}$$

(3) ELM-AE layer E1

Owing to the desirable property of ELM-AE in dimension reduction, in this work, we exploit ELM-AE for feature extraction and feature aggregation. Calculating the first layer of ELM-AE hidden layer output $H^{(1)}$. The weight output $\beta^{(1)}$ is as shown:

$$H^{(1)} = G\left(A, b, C^{(1)}\right) \tag{3}$$
$$\beta^{(1)} = \left(\frac{I}{\lambda} + H^{(1)T} H^{(1)}\right)^{-1} H^{(1)T} C^{(1)} \tag{4}$$

where, $A^T A = I_{L_1}$, $b^T b = 1$, $A = [a_1, ..., a_{L_1}]$, $b = [b_1, ..., b_{L_1}]$, $G(\cdot)$ is activation function.

Calculating the feature output of ELM-AE layer:

$$E^{(1)} = G\left(\beta^{(1)}, C^{(1)}\right) \tag{5}$$

where $G(\cdot)$ is activation function. λ denotes the regularization parameter to control the balance between the training error and the model complexity.

(4) Pooling layer P1

The combinatorial pooling is used to formulate the combinatorial node. And the pooling map is of the same size with the feature map. $e^{(1)}_{i,j,k}$ denote node (i, j) in the k-th output of ELM-AE, $h^{(1)}_{p,q,k}$ denote combinatorial node (p, q) in the k-th pooling map.

$$h^{(1)}_{p,q,k} = \sqrt{\sum_{i=p-e}^{p+e}\sum_{j=q-e}^{q+e}\left(e^{(1)}_{i,j,k}\right)^2}$$
$$p, q = 1, ..., (d - r + 1), \quad k = 1, 2, ...K_1 \tag{6}$$

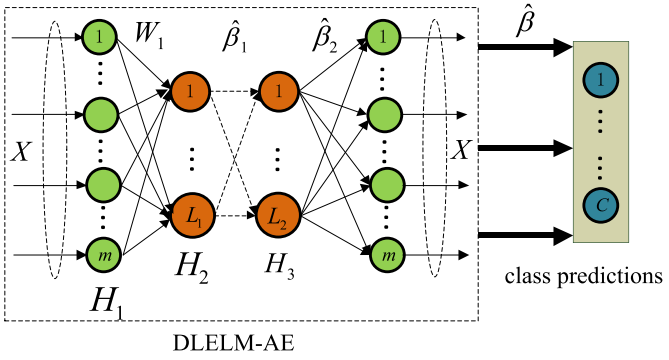


Fig. 4. DLELM-AE architecture.

Pooling size e is the distance between the center and the edge of the pooling area.

(5) convolution layer C2

The size and the number of the second layer random convolutional kernel are $r \times r$ and K_2 respectively. Convolutional feature map is calculated as:

$$c^{(2)}_{i,j,k}(x) = \sum_{m=1}^{r}\sum_{n=1}^{r} h^{(1)}_{i+m-1,j+n-1} \cdot a_{i,j,k},$$
$$i, j = 1, ..., (d - r + 1), \quad k = 1, 2, ..., K_2 \tag{7}$$

(6) ELM-AE layer E2

Calculating the second ELM-AE layer weight output and feature output as:

$$H^{(2)} = G\left(A, b, C^{(2)}\right) \tag{8}$$
$$\beta^{(2)} = \left(\frac{I}{\lambda} + C^{(2)T} C^{(2)}\right)^{-1} C^{(2)T} H^{(2)} \tag{9}$$
$$E^{(2)} = G\left(\beta^{(2)}, C^{(2)}\right) \tag{10}$$

where, $A^T A = I_{L_2}$, $b^T b = 1$, $A = [a_1, ..., a_{L_2}]$, $b = [b_1, ..., b_{L_2}]$, $G(\cdot)$ is activation function.

(7) Pooling layer P2

The second pooling feature map is calculated as:

$$h^{(2)}_{p,q,k} = \sqrt{\sum_{i=p-e}^{p+e}\sum_{j=q-e}^{q+e}\left(e^{(2)}_{i,j,k}\right)^2}$$
$$p, q = 1, ..., (d - r + 1), \quad k = 1, 2, ...K_2 \tag{11}$$

(8) DLELM-AE

DLEM-AE is used to obtain a high-level feature representation of the input image, as shown in Fig. 4.

The hidden layer output and weight of DLELM-AE are shown as follows:

$$H_j = g(W_j H_{j-1} + B_j)$$
$$W_j^T W_j = I, \qquad B_j^T B_j = I \qquad j = 1, 2 \tag{12}$$
$$W_j = \hat{\beta}_j$$

$g(\cdot)$ is activation function.

Table 1
Parameter settings of the DRKCELM and DLELM-AE joint network.

Layer	Layer name	Size/stride	Output
L0	Input layer	416 × 416, 3 channels	–
L1	Convolution layer	7 × 7/64, s-2	208 × 208
L2	ELM-AE	64	208 × 208
L3	Combined pooling layer	2 × 2/2	208 × 208
L4	Convolution layer	3 × 3/128, s-2	104 × 104
L5	ELM-AE	128	104 × 104
L6	Combined pooling layer	2 × 2/2	104 × 104
L7	DLELM-AE	H1: 1280, H2: 1024	1024
L8	Output layer	–	13 × 13 × 125

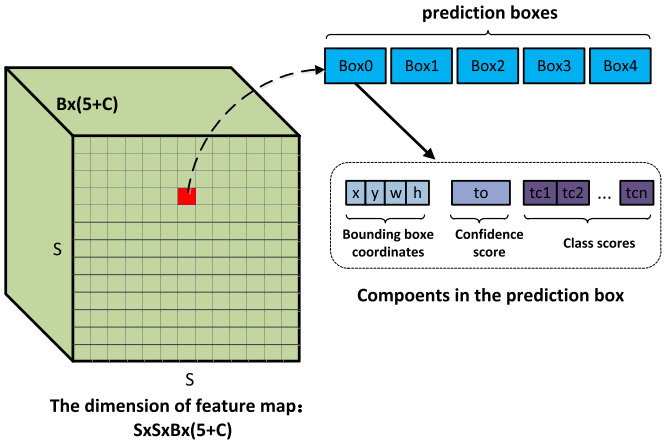


Fig. 5. Attributes of prediction feature map.

The final output weight matrix can be expressed as:

$$\begin{cases} \hat{\beta} = H^T(\frac{I}{\lambda} + HH^T)T, N \le K_2(d-r+1)^2 \\ \hat{\beta} = (\frac{I}{\lambda} + HH^T)H^T T, N > K_2(d-r+1)^2 \end{cases} \tag{13}$$

The specific parameters of the Faster-YOLO network are shown in Table 1.

3.4. Prediction feature map

As shown in Fig. 5, the output layer of Faster-YOLO is the prediction feature map, which is a 3-d tensor of dimensions: $S \times S \times B \times (5 + C)$. $S \times S$ represents the grid size of the input image. Each grid cell predicts B bounding boxes, where each prediction box includes bounding box coordinates (x, y, w, h) and the confidence score (t_o), and class scores $(t_{ci}, i = 1, 2, ..., C)$. For evaluating the YOLO on PASCAL VOC, setting the values: $S = 13$, $B = 5$, $C = 20$ (PASCAL VOC has 20 classes). So the final prediction is a $13 \times 13 \times 125$ tensor.

3.5. Bounding box prediction

Similar to YOLOv2 and YOLOv3, our system uses dimension clustering as anchor boxes to predict bounding boxes. The prediction of the network includes 4 coordinates (t_x, t_y, t_w, t_h) for each bounding box. (c_x, c_y) is the offset of the grid from the top left corner of the image. (p_w, p_h) is width and height of the bounding box prior. λ_w and λ_h represent the ratio of the width and height of minimum anchor boxes and the input image. $\sigma(\bullet)$ is sigmoid function. The corresponding predictions are shown in Fig. 6.

$$b_x = \big(\sigma(t_x) + c_x\big)/S$$
$$b_y = \big(\sigma(t_y) + c_y\big)/S$$
$$b_w = p_w \lambda_w \sigma(t_w)$$

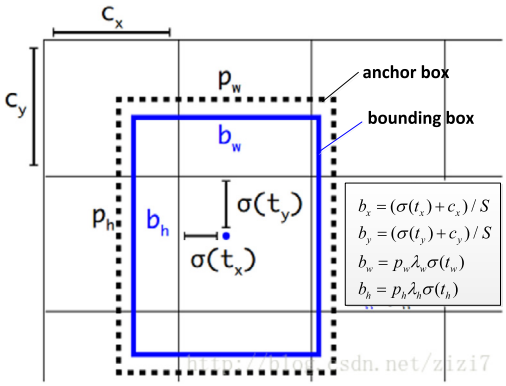


Fig. 6. Bounding boxes with dimension priors and location prediction.

$$b_h = p_h \lambda_h \sigma(t_h)$$
$$p_r(Object) * IOU(b, object) = \sigma(t_o) \tag{14}$$

Faster-YOLO uses independent logistic classifiers to predict an objectness score for each bounding box, which is similar to YOLOv3. The objectness score will be set to 1 if the bounding box prior overlaps a ground truth object by more than others. If the bounding box prior is not the best but IOU between the predicted box and the ground truth more than a chosen threshold, the prediction will be ignored.

Besides, the K-means algorithm is very sensitive to the selection of the initial clustering center and often requires an appropriate initial value setting. The selection of the initial clustering points has a great influence on the clustering results. Once the initial value is not selected well, effective clustering results may not be obtained, so this paper uses a more stable K-means ++ algorithm to determine our bounding box priors. It ensures that the selection of the initial point is sufficiently discrete, which further improves the rationality of the distribution of the a priori box. The distance metric is as follows:

$$d(box, centroid) = 1 - IOU(box, centroid) \tag{15}$$

$IOU(box, centroid)$ represents IOU scores between the bounding boxes and the cluster centroids, which are used for the clustering. On the VOC 2012 dataset we choose $K = 5$ as a good balance between model complexity and high recall, which is same as YOLOv2. The five different predictors are: (1.3221,1.73145), (3.19275,4.00944), (5.05587,8.09892), (9.47112,4.84053), (11.2364, 10.0071).

3.6. Object class prediction

Whether to choose a softmax classifier or multiple logistic classifiers depends on whether there is mutual exclusion between all categories in the dataset. When the label categories are mutually exclusive, it is suitable to select the softmax classifier; when the label categories are not completely mutually exclusive, it is suitable to choose multiple independent logistic regression classifiers. Overall, multi-class logistic regression classifiers have better generalization performance and simpler structure. In order to improve the versatility of the Faster-YOLO model, this paper still uses multiple logistic classifiers.

We define confidence as $P_r(Object) \times IOU(b, object) = \sigma(t_o)$, where $P_r(Object)$ represents the probability that the prediction bounding box b contains the object, and $IOU(b, object)$ represents the accuracy of the prediction bounding box b. If the confidence score is zero, which means there are no objects in the cell. Otherwise the confidence score is equal to the IOU between the predicted box and the ground truth.

In the test phase, if the prediction bounding box contains the object ($P_r(Object) = 1$), Faster-YOLO converts the category score t_{ci} into the conditional class probability $P(c_i|Object) = Logisic(t_{ci})$ with the trained logistic regression function. We can obtain class-specific confidence scores for each box by multiplying the conditional class probabilities and the individual box confidence predictions.

$$\begin{aligned} &P(c_i) \times IOU(b, object) \\ &= P(c_i|Object) \times P_r(Object) \times IOU(b, object) \\ &= Logistic(t_{ci}) \times \sigma(t_o) \end{aligned} \tag{16}$$

3.7. Fine-grained features

Faster-YOLO predicts detection results on a 13×13 feature map, but it may be insufficient for localizing smaller objects. For further improving the system's ability to detect small objects, we add a pass-through layer. By stacking adjacent features into different channels, the pass-through layer can combine the higher resolution features with the low resolution features. The $104 \times 104 \times 64$ feature map is turned into a $13 \times 13 \times 512$ feature map by the pass-through layer, and then we combine the original $13 \times 13 \times 1024$ features and the $13 \times 13 \times 512$ features into a $13 \times 13 \times 1536$ feature map. At the end, we perform 3×3 and 1×1 convolutions on this feature map to form a final $13 \times 13 \times 125$ tensor for detection predictions.

3.8. Loss function

The training process of Faster-YOLO is inherited from YOLOv2 and YOLOv3, and the objective function is adjusted according to the predictor output format to minimize the multi-task loss function and achieve optimal training of network parameters. The loss function contains coordinate error E_{Coord}, confidence error E_{Conf} and classification error E_{Obj}. The final multi-part loss function is as follows:

$$\begin{aligned} E &= E_{Coord} + E_{Conf} + E_{Obj} = \\ &= \lambda_{coord} \sum_{i=1}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} (2 - \hat{w}_i * \hat{h}_i) \Big[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \\ &\quad + \frac{(w_i - \hat{w}_i + h_i - \hat{h}_i)^2}{\hat{w}_i * \hat{h}_i} \Big] \\ &\quad + 0.01 * \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{noobj} \Big[(0.5 - \hat{x}_i)^2 + (0.5 - \hat{y}_i)^2 \\ &\quad + \frac{(1 - \hat{w}_i - \hat{h}_i)^2}{\hat{w}_i * \hat{h}_i} \Big] \\ &\quad + \lambda_{obj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} \big[\hat{C}_i \log(C_i) + (1 - \hat{C}_i) \log(1 - C_i) \big] \\ &\quad + \lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{noobj} \big[\hat{C}_i \log(C_i) + (1 - \hat{C}_i) \log(1 - C_i) \big] \\ &\quad + \sum_{i=0}^{s^2} \sum_{c \in class} \big[\hat{p}_i(c) \log\big(p_i(c)\big) \\ &\quad + \big(1 - \hat{p}_i(c)\big) \log\big(1 - p_i(c)\big) \big] \end{aligned} \tag{17}$$

We use sum-squared loss for coordinates coordinate error and binary cross-entropy loss for the confidence and the class loss. Where 1_{ij}^{obj} denotes that the j-th bounding box predictor in cell i is responsible for that prediction. λ_{coord} is penalty factor for coordinate prediction. λ_{noobj} and λ_{noobj} are confidence penalty factor. C_i is confidence scores in cell i. $p_i(c)$ is the conditional class probabilities in cell i.

In this paper, based on the original YOLO loss function, a series of improvement measures are carried out:

(1) The contributions of the coordinate error and classification error of the object to the network loss function are different. In order to increase the loss weight of the bounding box coordinate prediction, we set the factor $\lambda_{coord} = 2$.
(2) Due to the limited number of predictors for $1_{ij}^{obj} = 1$, the confidence error of the grid containing the object and the grid without the object has different effects on the loss function. The confidence loss weight of no objects should be reduced, so we set different scaling factors ($\lambda_{obj} = 5$, $\lambda_{noobj} = 0.5$).
(3) In order to alleviate the influence of errors of different sizes on the overall detection effect of the image, the coordinates and dimensions of the prediction boxes are normalized, and the coordinate error of the height and width is jointly calculated in the loss function.

3.9. Training

Training for classification network. We train the Classifier on the ImageNet 1000 class classification dataset using the DRKCELM and DLELM-AE joint network framework. During the training process, the weight parameters are randomly generated, avoiding the complicated parameter iterative adjustment. The size of training images is $448 \times 448 \times 3$.

Training for detection network. We modify this network for detection by adding on a 3×3 convolutional layers with 1024 filters and 1×1 convolutional layers with the number of outputs we need for detection. For VOC we predict 5 boxes with 5 coordinates and 20 classes per box. The ReLU function is used as the activation function. A pass-through layer is added from the DRKCELM output layer to the DELM-AE output layer so that our model can fuse the fine-grained features.

The size of training images is $416 \times 416 \times 3$. A starting learning rate is 0.001, weight decay is 0.0005 and momentum is 0.9. We train the network for 160 epochs on training datasets VOC 2007 and VOC 2012.

4. Experimental results and analyses

We evaluated Faster-YOLO on the PASCAL VOC 2007 + 2012 object detection benchmark. In addition, in order to improve the detection speed, we use GPU acceleration with OpenCV.

4.1. Faster-YOLO method performance

Fig. 7 compares the training process of the different methods. It can be seen that Faster-YOLO can better locate the grid cell of possible objects in the process of training. Due to the more accurate positioning of the coordinate points, the height and width loss of the bounding box are also reduced accordingly. There is a complementary relationship between the accuracy of the location and the accuracy of the classification. Therefore, the classification loss and the confidence loss are reduced. Benefitting from the unique DRKCELM and DLELM-AE joint feature extraction network, the Faster-YOLO converges faster during training.

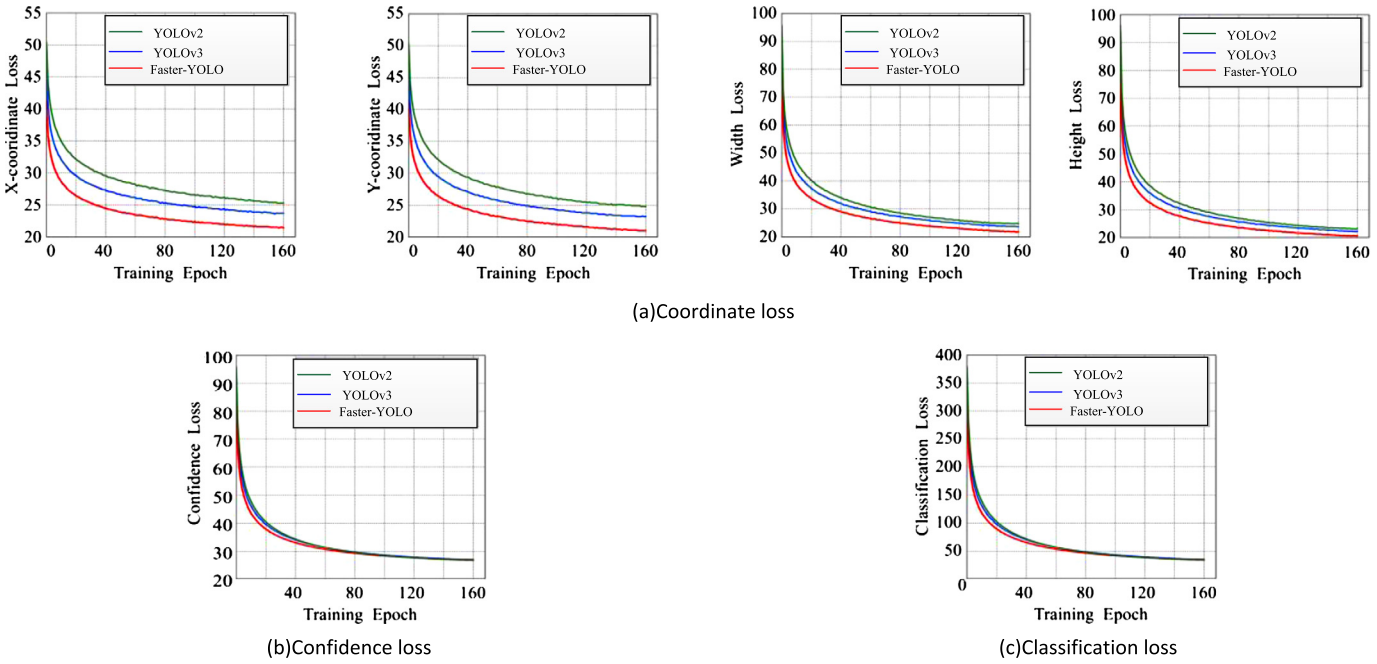


Fig. 7. Training process comparison of different methods. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

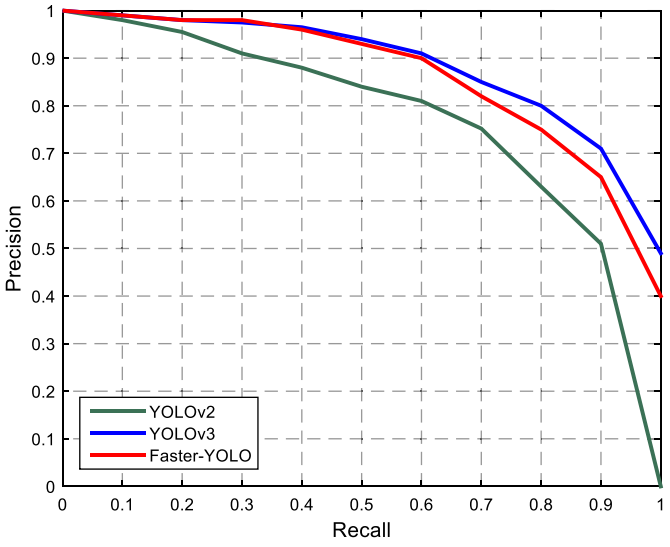


Fig. 8. Precision-Recall curve of different methods on the VOC dataset.

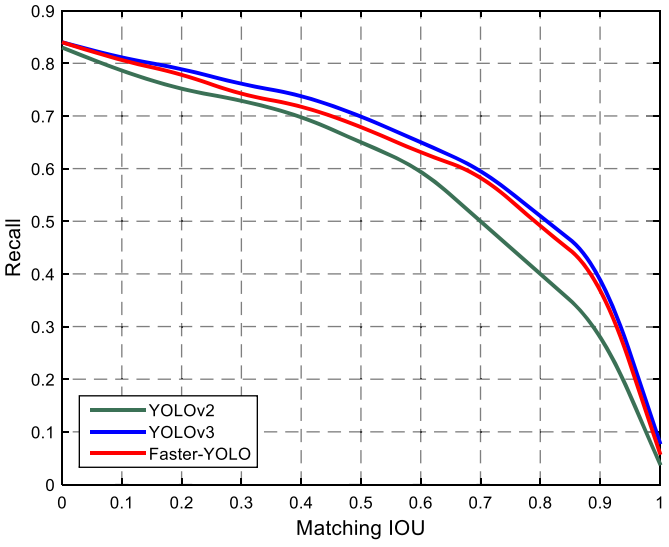


Fig. 9. Recall curves of different methods at different IOU threshold.

Fig. 8 shows the PR (Precision-Recall) curve for several different methods (YOLOv2, YOLOv3, and Faster-YOLO) on the PASCAL VOC dataset. It can be seen that Faster-YOLO's performance is between YOLOv2 and YOLOv3 by comparing the area under the curve (AUC). Compared with YOLOv2, Faster-YOLO has greatly improved the detection effect.

Fig. 9 is a comparison of the recall of the bounding box under different IOU thresholds. It can be seen that Faster-YOLO has a clear advantage over YOLOv2 algorithm and the performance is very close to YOLOv3. In general, the IOU threshold (TH) of the VOC dataset is set to 0.75 for all classes. It can be considered as a good result.

4.2. Comparison experiment of different detection model

Table 2 shows the performance of our Faster-YOLO and other algorithms (including Faster R-CNN, SSD, YOLOv2, YOLOv3) using

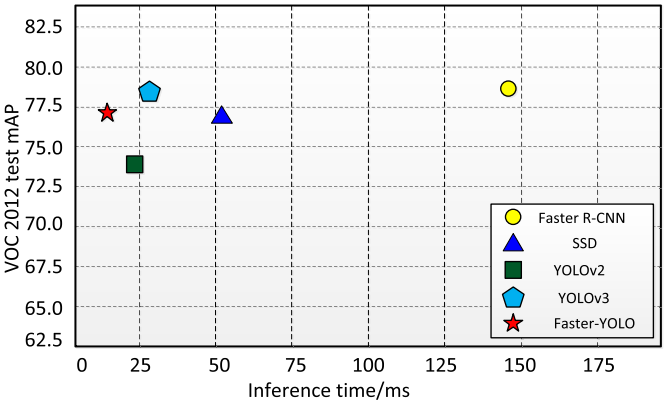


Fig. 10. Time-mAP comparison chart on VOC 2012.

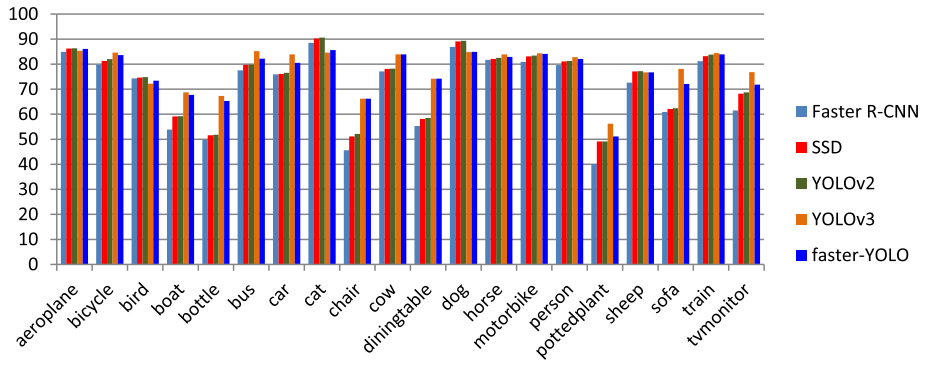


Fig. 11. Detection results of the different method on VOC 2012.

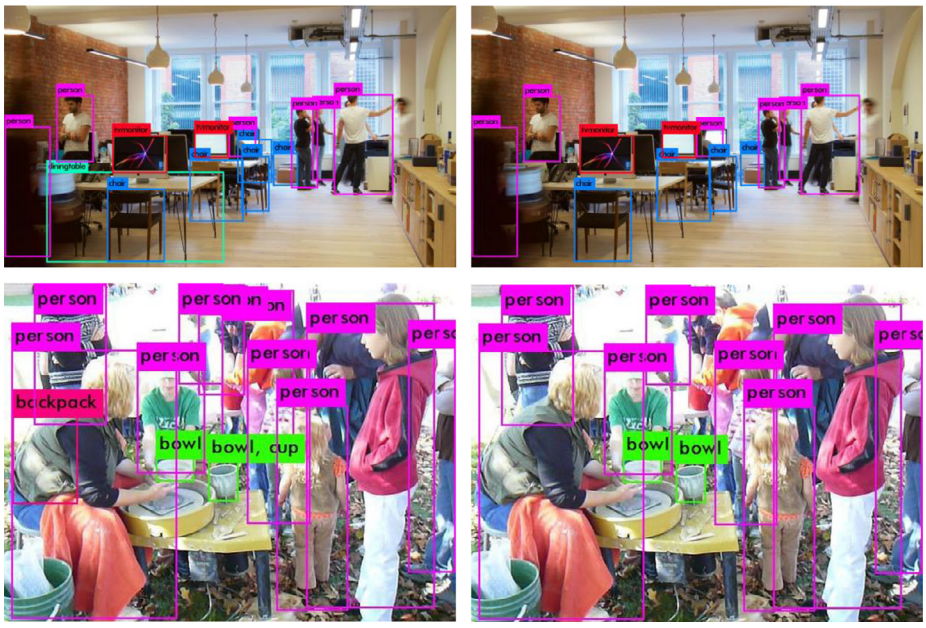


Fig. 12. Comparison of test results of different classifier.

Table 2
The performance comparison of the detection systems.

Method	Training set	Testing set	mAP (%)	FPS
Faster R-CNN ResNet [15]	VOC 2007 + 2012	VOC 2007	76.4	5
SSD500 [17]	VOC 2007 + 2012	VOC 2007	76.7	19
YOLOv2 [16]	VOC 2007 + 2012	VOC 2007	76.8	67
YOLOv3 [18]	VOC 2007 + 2012	VOC 2007	78.2	79
Faster-YOLO	VOC 2007 + 2012	VOC 2007	77.9	101

Table 3
Time performance comparison on VOC 2012.

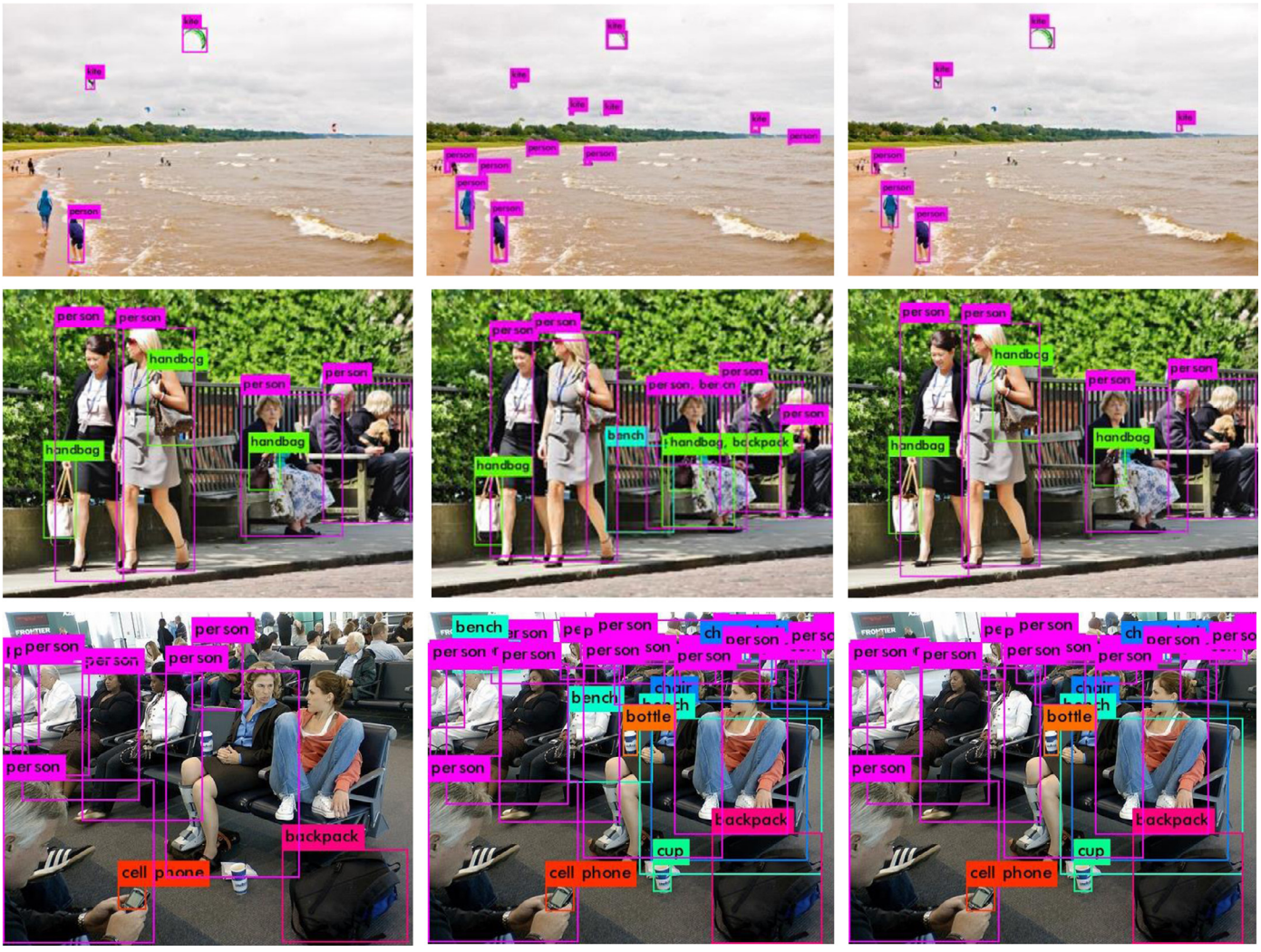
Method	Faster R-CNN	SSD	YOLOv2	YOLOv3	Faster-YOLO
Time (ms)	650	125	30	21	10

the VOC 2007 + 2012 training set and VOC 2007 testing set. Fig. 10 shows a comparison of time-mAP on the VOC2012 dataset.

It can be seen from the data in Table 2: (1) In terms of detection accuracy, Faster-YOLO are slightly improved, increasing by 1.1% comparing to YOLOv2, 1.2% higher than Faster SSD, 1.5% higher than Faster R-CNN, 0.3% lower than YOLOv3; (2) In terms of detection speed, Faster-YOLO is the highest, which is 22 FPS higher than the detection speed of YOLOv3, and 34 FPS higher than YOLOv2. This is because the network structure of the Faster-YOLO algorithm framework is greatly simplified compared to the DarkNet structure, and the number of iterations is reduced. (3) In general, the accuracy of the Faster R-CNN algorithm framework is high, but it does not reach real-time requirements. According to Table 2 and Fig. 10, compared with other detection algorithms, Faster-YOLO greatly improves the detection speed on the basis of ensuring detection accuracy, and better meets the application requirements of real-time systems.

The detection performance of different methods for different objects on the PASCAL VOC 2012 dataset can be seen from the Fig. 11, the average precision (AP) of Faster-YOLO for 20 different object categories is generally between YOLOv2 and YOLOv3, especially for the recognition of small objects, which is slightly improved compared to YOLOv2 and SSD. Compared with Faster R-CNN, the Faster-YOLO has a big improvement. But there is still a gap compared to YOLOv3. In categories such as bottle and tvmonitor, YOLO scores lower than YOLOv3 and slightly higher than YOLOv2. However, it is higher than YOLOv3 in the bird, cat, and dog categories, but lower than YOLOv2.

Time performance comparison on PASCAL VOC 2012 dataset is shown in Table 3. It can be seen that Fast-YOLO is the fastest object detection method. Time-consuming of Faster-YOLO is 10 ms, about half as much as that of the YOLOv3, one-third that of the YOLOv2. The detection speed is far faster than Faster R-CNN and



(a)Detection results using YOLOv2 (b)Detection results using YOLOv3 (c)Detection results using Faster-YOLO

Fig. 13. Test data randomly selected image object detection results.

SSD methods. This is just a picture test result, and the network training is carried out on a relatively large database, and it takes a certain time to test, so when in this case, time performance of Faster-YOLO will be more prominent. The results show that Faster-YOLO has a faster detection speed and fully meets the real-time application requirements.

4.3. Comparison of test results

In order to verify the influence of different classifiers on the detection results, the same image is detected in the Faster-YOLO model using the multi-class logistic classifier and the softmax classifier separately, as shown in Fig. 12. It can be seen that multi-class logistic classifiers can not only achieve the detection better performance than the softmax classifier, but also apply different data sets with overlapping labels.

The images were randomly selected from the VOC test data using YOLOv2, YOLOv3 and the Faster-YOLO network for object detection. The results are shown in Fig. 13. The three pictures selected from all the detection images as object detection results are very typical and effective, because they include tiny and blurry object objects, sparse objects, and dense objects.

As can be seen from the detection results in Fig. 13, the overall object detection performance of the Faster-YOLO is between YOLOv2 and YOLOv3. Through fusing the shallow fine-grained features with the deep features and the improvement of the loss function, Faster-YOLO improves the detection accuracy and enhances the detection ability of dense tiny and blurry objects.

5. Conclusion

The paper proposes a new model (Faster-YOLO) for real-time object detection on embedded devices. Comparing other object detection methods, the network framework of our model is simple and easy to construct, and can be trained directly on raw images. As an improved version of YOLO, Faster-YOLO replaces Darknet with the DRKCELM and DLELM-AE joint network for training and testing, multiple independent logistic classifiers are used to convert multi-classification problems into regression problems. It simplifies the training process and greatly improves the detection speed while ensuring higher detection accuracy. In addition, this paper carries out a series of improvement measures based on the original YOLOv2 loss function. In order to alleviate the impact of object errors of different sizes on the overall detection effect of the image, the coordinates and dimensions of the prediction boxes are normalized, and the coordinate error of the height and width is jointly calculated in the loss function. Finally, the experimental results show that Faster-YOLO is faster than other detection methods and provides a balance between speed and accuracy. Faster-YOLO has a good generalization performance, which makes it very suit-

able for real-time applications that rely on fast, powerful object detection.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is funded by National Natural Science Foundation of China (Grant No. 61402368). The authors thank all the anonymous reviewers for their very helpful comments to improve the paper.

References

[1] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[2] A.A.M. Al-Saffar, H. Tao, M.A. Talab, Review of deep convolution neural network in image classification, in: 2017 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET), 2017, pp. 26–31.
[3] Y. Li, H. Zhang, X. Xue, et al., Deep learning for remote sensing image classification: a survey, Wiley Interdiscip. Rev. Data Min. Knowl. Discov. 8 (6) (2018) e1264.
[4] A. Ferreyra-Ramirez, C. Aviles-Cruz, E. Rodriguez-Martinez, et al., An improved convolutional neural network architecture for image classification, Pattern Recognit. 11524 (2019) 89–101.
[5] M.A. Kadhim, M.H. Abed, Convolutional neural network for satellite image classification, in: Studies in Computational Intelligence, Intelligent Information and Database Systems: Recent Developments, vol. 830, Springer, Cham, 2019, pp. 165–178.
[6] M.A.O. Ahmed, O. Reyad, B.A. El-Rahiem, An efficient deep convolutional neural network for visual image classification, in: The International Conference on Advanced Machine Learning Technologies and Applications (AMLTA2019), vol. 921, 2019, pp. 23–31.
[7] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, 2001, pp. I511–I518.
[8] X. Dai, Hybridnet: a fast vehicle detection system for autonomous driving, Signal Process. Image Commun. 70 (2019) 79–88.
[9] B. Benjdira, T. Khursheed, A. Koubaa, et al., Car detection using unmanned aerial vehicles: comparison between faster R-CNN and YOLOv3, in: The 1st Unmanned Vehicle Systems Conference in Oman, 2019, pp. 1–6.
[10] S. Kamada, T. Ichimura, An object detection by using adaptive structural learning of deep belief network, in: 2019 International Joint Conference on Neural Networks (IJCNN), 2019.
[11] B. Vaidya, C. Paunwala, Deep learning architectures for object detection and classification: towards smarter algorithms, in: Smart Techniques for a Smarter Planet, 2019, pp. 53–79.
[12] R.B. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Computer Vision and Pattern Recognition (CVPR), 2014, pp. 580–587.
[13] R.B. Girshick, Fast R-CNN, in: IEEE International Conference on Computer Vision, 2015, pp. 1440–1448.
[14] K. He, X. Zhang, S. Ren, et al., Spatial pyramid pooling in deep convolutional networks for visual recognition, IEEE Trans. Pattern Anal. Mach. Intell. 37 (9) (2014) 1904–1916.
[15] S. Ren, K. He, R.B. Girshick, J. Sun, Faster R-CNN: towards real-time object detection with region proposal networks, TPAMI 39 (6) (2017) 1137–1149.
[16] J. Redmon, S.K. Divvala, R.B. Girshick, A. Farhadi, You only look once: unified, real-time object detection, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779–788.
[17] W. Liu, D. Anguelov, D. Erhan, et al., SSD: single shot multibox detector, in: ECCV, 2016, pp. 21–37.
[18] J. Redmon, A. Farhadi, YOLO9000: better, faster, stronger, in: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 6517–6525.
[19] J. Redmon, A. Farhadi, YOLOv3: an incremental improvement, arXiv:1804.02767, 2018.
[20] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 3, 2016, pp. 770–778.
[21] G.-B. Huang, Q.-Y. Zhu, C.K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1) (2016) 489–501.
[22] G.-B. Huang, Q.-Y. Zhu, C.K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Trans. Neural Netw. 17 (4) (2006) 879–892.
[23] G.-B. Huang, Q.-Y. Zhu, C.K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: Proc. of the International Joint Conference Neural Networks (IJCNN), vol. 2, 2004, pp. 985–990.
[24] G.-B. Huang, H.-M. Zhou, X.-J. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, IEEE Trans. Syst. Man Cybern., Part B, Cybern. 42 (2) (2012) 513–529.
[25] X.-Y. Wang, M. Han, Multivariate time series prediction based on multiple kernel extreme learning machine, in: Proc. of the International Joint Conference Neural Networks (IJCNN), vol. 3, 2014, pp. 198–201.
[26] L.L.C. Kasun, H.-M. Zhou, G.-B. Huang, C.M. Vong, Representational learning with extreme learning machine for big data, IEEE Intell. Syst. 28 (6) (2013) 31–34.
[27] H. Zhou, G.-B. Huang, Z. Lin, H. Wang, Y.C. Soh, Stacked extreme learning machines, IEEE Trans. Cybern. 45 (9) (2015) 2013–2025.
[28] Y.-M. Yang, Q.M.J. Wu, Multilayer extreme learning machine with subnetwork nodes for representation learning, IEEE Trans. Cybern. 46 (11) (2015) 2570–2583.
[29] G.-B. Huang, Z. Bai, L.L.C. Kasun, M.V. Chi, Local receptive fields based extreme learning machine, IEEE Comput. Intell. Mag. 10 (2) (2015) 18–29.
[30] Z. Bai, L.L.C. Kasun, G.-B. Huang, Generic object recognition with local receptive fields based extreme learning machine, Proc. Comput. Sci. 53 (2015) 391–399.
[31] H. Liu, F. Li, X. Xu, et al., Active object recognition using hierarchical local-receptive-field-based extreme learning machine, Memetic Comput. 10 (2) (2018) 233–241.
[32] F.-X. Li, H.-P. Liu, X.-Y. Xu, F.-C. Sun, Multi-modal local receptive field extreme learning machine for object recognition, in: International Joint Conference on Neural Networks, 2017, pp. 1696–1701.
[33] Q. Lv, X. Niu, Y. Dou, et al., Classification of hyperspectral remote sensing image using hierarchical local-receptive-field-based extreme learning machine, IEEE Geosci. Remote Sens. Lett. 13 (3) (2016) 1–5.
[34] M. Everingham, S.M. Ali Eslami, et al., The Pascal visual object classes challenge: a retrospective, Int. J. Comput. Vis. 111 (1) (2015) 98–136.
[35] J. Ren, X. Chen, J. Liu, W. Sun, J. Pang, Q. Yan, Y.-W. Tai, L. Xu, Accurate single stage detector using recurrent rolling convolution, in: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 752–760.
[36] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning, vol. 1, 2015, pp. 448–456.
[37] N. Srivastava, G. Hinton, A. Krizhevsky, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (2014) 1929–1958.
[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, et al., Going deeper with convolutions, in: IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.
[39] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: International Conference on Learning Representations (ICLR), 2015.
[40] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

Yunhua Yin, (1979-), male, from Shanxi, a teacher from North University of China, a PhD candidate in Northwestern Polytechnical University now, mainly engaged in deep learning, object detection and recognition research work.

Huifang Li, (1962-), male, from Shanxi, Professor in Northwestern Polytechnical University, mainly engaged in Image detection and recognition, Microelectronics and microelectronics, Quantum information processing research work.

Wei Fu, (1979-), male, from Shanxi, Associate professor from North University of China, mainly engaged in deep learning, Image recognition research work.