Computers and Electrical Engineering 77 (2019) 398–408

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

A real-time object detection algorithm for video☆

Shengyu Lu[a], Beizhan Wang[a,*], Hongji Wang[a], Lihao Chen[b], Ma Linjian[a], Xiaoyan Zhang[c]

[a] Software School, Xiamen University, Siming South Road, Xiamen City, Fujian Province, 361005, China
[b] Beijing University of Posts and Telecommunication, West Tucheng Road, Beijing City, Beijing, 100876, China
[c] Tan Kah Kee College, Xiamen University, Siming South Road, Xiamen City, Fujian Province, 361005, China

ARTICLE INFO

Article history:
Received 10 August 2018
Revised 21 May 2019
Accepted 23 May 2019

Keywords:
Object detection
GoogleNet
YOLO
Real-time
Video

ABSTRACT

Deep learning technology has been widely used in object detection. Although the deep learning technology greatly improves the accuracy of object detection, we also have the challenge of a high computational time. You Only Look Once (YOLO) is a network for object detection in images. In this paper, we propose a real-time object detection algorithm for videos based on the YOLO network. We eliminate the influence of the image background by image preprocessing, and then we train the Fast YOLO model for object detection to obtain the object information. Based on the Google Inception Net (GoogLeNet) architecture, we improve the YOLO network by using a small convolution operation to replace the original convolution operation, which can reduce the number of parameters and greatly shorten the time for object detection. Our Fast YOLO algorithm can be applied to real-time object detection in video.

1. Introduction

In recent years, with the rapid development of computer vision, object detection (OD) as an important part of computer vision has been widely used in many fields. OD based on image processing extracts the features from images, and then obtains and analyzes the object information such as the category, position and direction. OD is widely used in many real-time situations such as video monitoring, abnormal behavior analysis and mobile robots. This approach can obtain much valuable information through feature extraction and analysis. However, the method faces many challenges, especially in terms of the high computational and memory requirements.

The traditional machine learning methods extract the object features from images and then input the features into a classifier [1]. The traditional methods of extracting features include the histogram of oriented gradient (HOG) method, scale invariant feature transform (SIFT) [2], etc. The methods of classification include the support vector machine (SVM), Bayesian, decision trees, etc. These methods mainly rely on prior knowledge. They are not real-time because they down sample constantly. In addition, these methods have few feature points and edge feature extraction is sometimes not accurate. The core of these methods is feature extraction, and the quality of feature extraction directly affects the method performance. However, in practical applications, these methods are mostly targeted at recognizing specific objects using small datasets and

☆ This paper is for regular issues of CAEE. Reviews processed and recommended for publication to the Editor-in-Chief by Area Editor Dr. E. Cabal-Yepez.

* Corresponding author.
E-mail addresses: wangbz@xmu.edu.cn, 3137349575@qq.com (B. Wang).

https://doi.org/10.1016/j.compeleceng.2019.05.009
0045-7906/

have poor generalization ability. Although machine learning methods are constantly being optimized, from the extraction of low-level features to the emergence of images, the most successful method is the deformable part model (DPM). However, this method has slow detection and depends on the geometric properties of the samples. At present, the traditional machine learning methods cannot meet the efficiency, performance, speed and intelligence requirements of data processing in OD technology.

With the advent of deep learning technology, the field of computer vision has developed rapidly. Deep learning technology has been applied for image recognition [3], and it has made great progress in object recognition in recent years. Deep learning technology can process and analyze features through the learning and imitation of the human brain's cognitive ability, which has a great effect in OD [4]. Different from the traditional feature extraction methods, deep convolutional neural networks can achieve a high level of accuracy by extracting the features using multilayer convolution operations. In addition, they are robust in terms of geometric transformation, deformation, and illumination and can overcome the difficulties caused by environmental changes. The deep learning methods can construct the feature description adaptively using the training data, and they are highly flexible and have high generalization ability. Among the deep learning methods, region-convolutional neural network(RCNN) [5], Faster region-convolutional neural network(Faster RCNN) [6], You Only Look Once (YOLO) [7] and single shot multibox detector(SSD) [8] are the most widely used methods in OD. However, the current OD methods based on deep learning still have problems due to the slow detection speed and high time consumption. In this paper, we propose a real-time video OD method. We introduce the Fast YOLO algorithm for OD theory in abstract and then introduce the Fast YOLO framework in detail, including the preprocessing procedure, model training and loss function. Next, we verify the performance of the Fast YOLO algorithm through some experiments. Considering the theory, proposed methods and experiments, we can achieves an excellent performance. In detail, The Section 2 and Section 3 include the background, related work and the main framework of our method. We first use the frame difference and background subtraction methods to preprocess the image. Then, we improve the YOLO algorithm by learning from the efficient convolution operation of the Google Inception Net (GoogLeNet) and use a small convolution operation in the YOLO Network to conduct the convolution operation. In Section 4, the experiments are described, and the experimental results prove that our method can greatly improve the speed of OD compared with the original YOLO algorithm. In the conclusion, we summarize the Fast YOLO algorithm, considering its application to OD in video. Our method can be applied for real-time OD in video with great significance.

Our method has some advantages and disadvantages. Our approach improves the original YOLO algorithm and the detection speed is very fast. However, our method is limited to some extent, and it is not effective for detecting small and dense target objects. Although the YOLO algorithm can reduce the probability of the background being regarded as an object, it also results in a lower recall rate.

The organization of this paper is as follows. Section 1 introduces the object detection and our Fast YOLO algorithm. Section 2 reviews the background and related research. Section 3 introduces the improvements of the Fast YOLO algorithm in detail. Section 4 shows the performance results of our method and the baseline methods in the experiments. Section 5 presents the conclusions and future work.

2. Background and related work

With the development of deep learning technology, it has been widely used in OD due to its powerful feature extraction ability. In recent years, research on OD algorithms based on deep learning has attracted the attention of many scholars. In 2014, the RCNN model was proposed by Ross Girshick's team. Multiple candidate regions were selected from the input images and a CNN was used to extract the regional features. Then, the optimal regions were obtained using the non-maximal suppression method [5]. To solve the problems of the complex training steps and the high time consumption of the RCNN, Kaiming He et al. proposed the spatial pyramid pooling(SPP) Net, which reduced the time consumption by performing a convolution operation on the whole graph, and then mapped the location information of the candidate region to a feature map. In 2015, Ross Girshick's team improved the RCNN and designed the Faster RCNN, which was mainly used to solve the problem of repeating the computation of the candidate windows. They innovatively proposed the region proposal network(RPN) to generate the candidate regions, which enabled the regional nomination, classification and regression to share the convolution features and significantly improved the efficiency and performance of OD [9]. In 2016, Joseph Redmon et al. proposed the YOLO algorithm which can obtain the candidate regions of images using uniform segmentation, greatly improving the detection speed and ensuring high accuracy [7]. In addition, many other scholars have proposed advanced methods for OD. Yadav et al. proposed a novel method based on motion detection when there is a dynamic scene object in the background [10]. This work developed a background subtraction method based on the statistical p parameter and achieved excellent performance. Lavanya Sharma et al. provided a detailed overview of a proposed video surveillance system that uses the background subtraction method to detect moving objects, which can be applied to security for cities and towns or IP cameras can be used to capture video sequences at home [11]. Mehran Yazdi et al. presented a survey of the advanced methods for detecting moving objects in the video sequences captured by mobile cameras, and they also presented the challenges and key concerns in the field [12]. Lavanya Sharma et al. proposed a powerful background subtraction method for solving the problem of illumination changes based on motion [13]. Dileep Kumar Yadav et al. proposed a robust method for moving OD in video frames and developed a background model based on the trim mean. It can reduce the dynamic component of the noise or background [14]. Lavanya Sharma et al. proposed a new background subtraction method based on the threshold

value of the linear discriminant. This threshold value is updated automatically at runtime for each pixel of each successive frame [15].

2.1. The GoogLeNet model

Unlike the convolution layers of a traditional neural network, the GoogLeNet model reduces the number of parameters and controls the amount of computation [16], which achieves good classification performance and greatly improves the detection speed. The Google Inception Network is a deep neural network, consisting of 22 layers, 21 convolutional layers and one fully connected layer. A major innovation in the design of Google Inception Network is the multi-scale convolution applied to the local densification of sparse matrix operations and the parallel processing and composition of the features, which can greatly improve the speed.

The Inception module removes the full connected layer and replaces it with a global average pooling layer (resizing the image to 1×1), which makes the model training faster and reduces the probability of overfitting. The method of replacing the fully connected layer with the global average pooling layer is adopted from the network in network (NIN) model architecture. The design of the inception module during the inception stage improves the parameter utilization efficiency. The inception module is like a small network in a large network whose structure can be repeatedly superimposed to train the large network.

2.2. YOLO

YOLO is an end-to-end method of OD based on a non-regional candidate [17]. Unlike other OD methods such as the Fast RCNN, YOLO does not divide the object recognition task into multiple processes, such as object region prediction and class prediction. The YOLO algorithm integrates both of tasks into a single neural network model to achieve fast detection with high accuracy.

YOLO converts the OD to extract the features from images directly and calculate the bounding boxes and probabilities of the categories; it then obtains the object categories and position information. YOLO adopts a single convolutional neural network to predict the multiple bounding boxes and probabilities of the categories. Compared with traditional OD methods, the YOLO algorithm has these advantages: (1) the process of YOLO is simple and the detection speed is very fast. In this paper, we improved the convolution operation of convolutional neural network based on the GoogLeNet architecture by using a small convolution operation to replace the original convolution operation. Our method can optimize the YOLO algorithm to improve the speed of OD. (2) In the training and prediction processes, YOLO extracts and uses the feature information of the whole graph, which greatly reduces the error rate compared with local detection. (3) The YOLO algorithm can learn the important features accurately and the general information of the objects universally.

3. Methods

3.1. Image preprocessing

The purpose of image preprocessing is to intercept each frame image in the video and separate the background and target objects after removing the random noise from the image. The current methods include the optical flow, background difference [18] and frame difference methods [19],etc. These methods have simple steps and a fast detection speed. However, these methods also have some disadvantages. The background difference method needs to obtain the background images in advance with is sensitive to environmental factors such as light changes, which causes it easily to make wrong judgments. The frame difference method uses to make the difference between two consecutive frames or several consecutive frames in the video. If the environmental change between two frames is weak, the frame difference method can obtain the moving object accurately. However, objects with a slow speed, may not be detected, and environmental changes such as light variation have a large influence on the results. In this paper, we use a combination of the frame difference and background difference methods to separate the image background.

A video is approximately 25 frames per second, so the time between two adjacent frames is very short. The background and moving objects pixels change significantly, so we can assume that the brightness of a particular pixel follows a Gaussian distribution [20]. Notably, (x,y) represents the coordinates of a pixel, and $B(x,y) \sim N(\mu, \sigma 2)$ is satisfied, where μ represents the mean difference of the Gaussian distribution, and $\sigma 2$ represents the variance of the Gaussian distribution.

$$\mu(x,y) = \frac{1}{N}\sum_{i=0}^{N-1} F_i(x,y) \tag{1}$$

$$\sigma(x,y) = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1}[F_i(x,y) - \mu(x,y)]^2} \tag{2}$$

The Gaussian model is trained by pixel points, and these two parameters are optimized in the training progress. The difference operation is performed on the pixel points in the background regions of the news and previous image frames to obtain different values. If this value exceeds the threshold value, it is determined that an object has been detected.

The detection of the motion regions uses the frame difference method. By comparing the difference value between adjacent frames and the threshold value, we can determine the changeable regions. The formula is defined as follows:

$$D_k(x, y) = | f_k(x, y) - f_{k-1}(x, y) | \tag{3}$$

We use the frame difference method to remove the similarity between the two frames, and then adopt the background difference method to detect the objects. The Gaussian model is used to fit the pixel points.

3.2. Fast YOLO

In the Google Inception Network, the fully connected layer is replaced with a pooling layer and the inception module is designed to improve the efficiency of parameter utilization. Fig. 1 shows the structure of the Google Inception Network. There are four branches in the network. The first branch convolves the input by 1*1 convolution. The 1*1 convolution organizes the information across the channels and reduces the dimensions of the output channels to improve the network. As seen in Fig. 1, the 1*1, 3*3, and 5*5 convolution layers, and the 3*3 max pooling layer are completely entirely before the input, and the stack is the output. When an image input, the network will choose the right path. If we perform the convolution kernel operation in the output of previous layer, the computational complexity of the 5*5 convolution kernels is too large, which may cause the thickness of the feature maps to be too large. In order to reduce the computational load and speed up the operation, we add a 1*1 convolution layer before the 3*3 convolution, 5*5 convolution and max pooling layers to reduce the thickness of the feature maps. The main framework of the Google Inception Network is as follows:

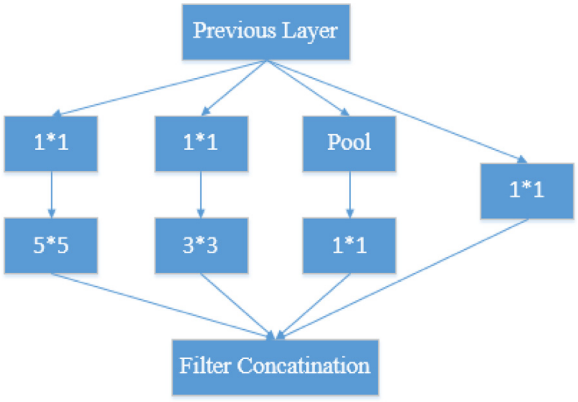

Fig. 1. The framework of Google Inception Network.

YOLO is an improved CNN [21] that, adopts a convolutional neural network to extract features, and then uses the full connected layer to predict the object information. The model includes 24 convolutional layers and 2 fully connected layers. The convolutional layers, mainly use 1×1 convolution to reduce the channels. The convolutional layers and the fully connected layer use the ReLU activation function, and the last layer uses a linear activation function.

As shown in Fig. 2, the YOLO model includes 24 layers. The $L^{(3)}$ to $L^{(24)}$ layers and are similar to the $L^{(1)}$ and $L^{(2)}$ layers. There are two convolutional layers and the max-pooling layers appear alternately in each layer. The convolution kernels of all the convolution layers size=3, stride=1 and pad=1. All the max-pooling layers have size=2 and stride=2. The output layer is a tensor of 7×7 $(5 \times 5 + 5)$ dimensions.

YOLO can use the whole image to train the model directly, and does not enquire the extraction of all the regions from the images. The labels of images include both the categories and positions. YOLO divides the images into $S*S$ grids and then, for each object and grid, it calculates the probability that the center of the objects falls within the grids. If the probability exceeds a threshold value, it is determined that there is an object in the grid. B boundary boxes are built for the grids with objects, and the confidence level of each box is computed simultaneously. The confidence level reflects the probability that the boundary box contains the object. The formula is defined as follows:

$$P_b = P_r(Object) * IOU \tag{4}$$

$$IOU = \frac{area(BB_{dt} \cap BB_{gt})}{area(BB_{dt} \cup BB_{gt})} \tag{5}$$

Where, $P_r(Object)$ represents the probability that the boundary box contains an object, BB_{gt} represents the reference boundary box based on the training labels, BB_{dt} represents the detection boundary box and $area(\cdot)$ represents the area.

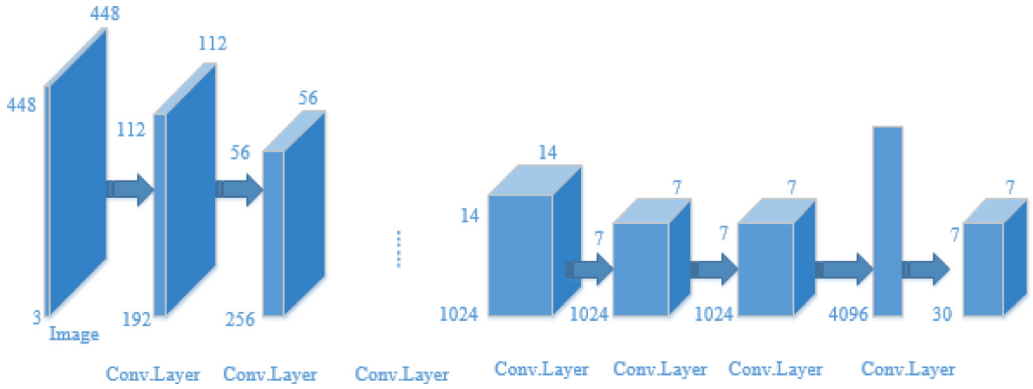

Fig. 2. The framework of YOLO.

There are five parameters in each boundary box: *x, y, w, h* and the confidence level. Where, the center position of the boundary box relative to the original grid is represented by *(x, y), w* represents the width of the boundary box, and *h* represents the height of the boundary box. Each grid also predicts $P_r(Class_i|object)$, which is the probability of the object belonging to one of the class C, and it denotes the probability that the center of class *i* falls within the grid. Finally, the network outputs tensors of $S*S*(B*5+C)$ dimensions.

For each frame in the video stream, we divide the image into $S*S$ grids. We use five different sizes for the bounding boxes for the center of each grid, and then extract the features of the boxes and predict the probability of the categories for each bounding box. Based on the multiple bounding boxes in each grid, we predict the probability of the categories and take the category with the highest probability as the category of the object. Therefore, our method can overcome the issue of overlapping of objects. When objects overlap in the image, different categories of vehicles will appear in different bounding boxes with the highest probability. Therefore, we can obtain the category and location information for different vehicles even if the objects overlap.

3.3. The loss function

We use the sum of squared errors as the loss function. The YOLO algorithm divides the images into $S*S$ grids and the output of each grid has dimensions of $(B*5+C)$ and includes the position information, the confidence level of the boundary box, and the class probabilities.

Our loss function is defined as the sum of squared errors and consists of three types of error information. The loss function is defined as follows:

$$\text{Loss} = \sum_{i=0}^{s^2} (coordError + iouError + classError) \tag{6}$$

For the size and position errors of the boundary frame, the effect of the former is more noticeable. Therefore, we replace *w, h* with $\sqrt{w}, \sqrt{h}$. The coordError represents the sum of the squared errors of the position information and it is defined as follows:

$$\text{coordError} = \sum_{j=1}^{B} I_{ij}^{obj}\left[\left(x_i - \hat{x}\right)^2 + \left(y_i - \hat{y}\right)^2\right] + \sum_{j=1}^{B} I_{ij}^{obj}\left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2\right] \tag{7}$$

Where, we let I_i represent whether there is an object at the center of grid *i*. If there is an object in the grid, I_i is 1, otherwise it is 0. I_{ij} denotes whether there is an object in the boundary box *j* of grid *i*. If so it is 1; otherwise it is 0.

In detail, we compute the value of *IOU* value of the current bounding box and all the reference bounding boxes, and select the object with the maximum *IOU* as the object detected by the current bounding box. The *iouError* represents the sum of the squared errors of the confidence level and it is defined as follows:

$$\text{iouError} = \sum_{j=0}^{B} I_{ij}^{obj}\left(C_i - \widehat{C}_i\right)^2 + \sum_{j=0}^{B} I_{ij}^{nobj}\left(C_i - \widehat{C}_i\right)^2 \tag{8}$$

Fig. 3. The simple images.

The classError represents the sum of squared errors of the classes and it is defined as:

$$classError = I_i^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \tag{9}$$

Obviously, these factors have different effects on the accuracy of object recognition. In addition, many grids do not contain objects, and the confidence level of the boundary boxes built by them is 0. In general, the gradient used in the training process of these grids without objects is higher than that of the grids containing objects, which will result in unstable training and divergence. Therefore, we set different weights for different grids. We let α_{coord} represent the weight of the grids that contain the center of an object, which is equal to 5. α_{noobji} represents the weight of the grids that have no object, which is equal to 0.5.

Therefore, we improved the loss function and defined it as follows:

$$\begin{aligned} \text{Loss} &= \sum_{i=0}^{s^2} \left(\alpha_{coord} * coordError + \alpha_{noobj} * iouError + classError\right) \\ &= \alpha_{coord} \sum_{i=0}^{S_2} \sum_{j=0}^{B} I_{ij}^{obj} \left[\left(x_i - \hat{x}\right)^2 + \left(y_i - \hat{y}\right)^2 \right] + \alpha_{coord} \sum_{i=0}^{S_2} \sum_{j=0}^{B} I_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \right] \\ &\quad + \sum_{i=0}^{S_2} \sum_{j=0}^{B} I_{ij}^{obj} \left(C_i - \hat{C}_i\right)^2 + \alpha_{noobj} \sum_{i=0}^{S_2} \sum_{j=0}^{B} I_{ij}^{nobj} \left(C_i - \hat{C}_i\right)^2 + \sum_{i=0}^{S_2} I_i^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \end{aligned} \tag{10}$$

4. Experiments

In this section, we use several data sets and comparative methods to experiment. We use different evaluation metrics to demonstrate the effectiveness and adaptability of our method [22]. The main equipment in the experiment included the E5 processor, four-channel GPU (GTX-1080) and 64 GB of memory. We implement the algorithm by using Python in TensorFlow framework.

4.1. Data sets

Our datasets consist of vehicle monitoring videos from smart cities from the Xiamen municipal transportation bureau. The videos consist of several 416*416 color image. We preprocess the videos with the combined of the frame difference and background difference methods. We obtained 8000 1280×720 images. There are 6000 images in the training set and 2000 in the test set. In the training set, we mark 12,000 labels for several classes and statistic on five classes including cars, buses, trucks, motorcycles and pedestrians. We can see these five simple images as Fig. 3.

We use the images with labels to train the model and optimize the parameters of the improved YOLO algorithm. When the video is input, we capture each image frame in the video for preprocessing, and then input them into the improved YOLO algorithm. Through the YOLO network, we can extract the images features from the video and obtain the location and category information as shown in Fig. 4. We implement object detection for the vehicles in the video.

4.2. Evaluation metrics

We use the precision and recall rates and the frames recognized per second (FPS) to evaluate the experimental results. We compute the *IOU* of the detection boundary boxes and reference boundary boxes to judge whether the results are true or false.

$$IOU = \begin{cases} \geq 0.5 \ ture \\ < 0.5 \ false \end{cases} \tag{11}$$

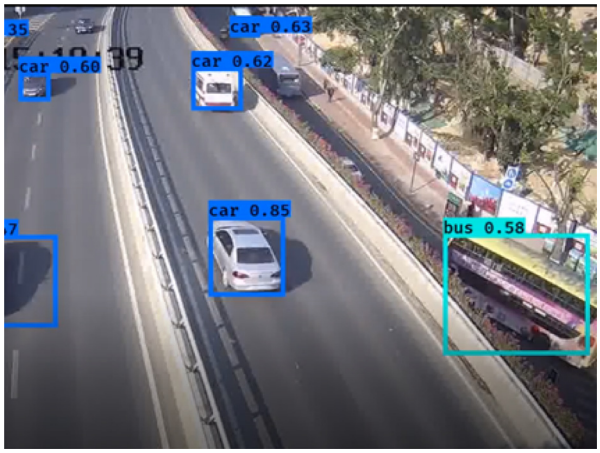

Fig. 4. Vehicle detection image.

Table 1
Experimental results of evaluation metrics.

Algorithm	Precision (%)	Recall (%)
Sliding window	70.59	72.91
CNN	80.82	78.38
RCNN	84.19	83.69
Faster R-CNN	83.96	82.65
YOLO	88.34	86.22
Fast YOLO	**88.45**	**86.64**
SSD	86.82	84.10

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

Where, *TP, FP, and FN* represent the number of real cases, false positive cases and false negative cases respectively.

4.3. Baseline approaches

The models compared to our model are as follows.

1. Sliding Window: In this model, the images are divided into several grids and the sliding window is used to extract the features of each grid and predict the probability of categories for the objects [23].
2. CNN: The Image features are extracted by a convolution operation based on the color, edge, texture, etc. [21].
3. RCNN: The images are divided the images into several local regions. These regions are input into the CNN to obtain the regional features, and then classifier is used to determine whether the region is an object or background [5].
4. Faster R-CNN: On the basis of the RCNN, the extracted feature regions are mapped to the feature map of the last convolutional layer of the CNN so that the image feature of images are only extracted once [6].
5. YOLO: The image is divided into several grids, and the probability of the center of an object falling within the grid is calculated. B boundary boxes are built to calculate the category probabilities of the objects [7].
6. SSD: Different scales of feature mapping are extracted from the output of different layers. The mess are divided into different scales to determine the object categories in the grids [8].

4.4. Results and analysis

In the experiments, we use five datasets to test the performance of these algorithms. We obtain the precision rate, recall rate and the frames of recognized per second (*f/s*) for the YOLO algorithm and the baseline methods for different datasets. We also calculate the average of each evaluation metric for the different datasets, as shown in Table 1 and Fig. 5 and Fig. 6.

Compared with the other algorithms, both the Fast YOLO and YOLO algorithms achieve better results, especially in terms of the recall rate. In the experiments, in order to ensure the real-time performance of YOLO, we set the grid parameter S to 8. Therefore, perhaps the possibility of a missed detection reduced the recall rate.

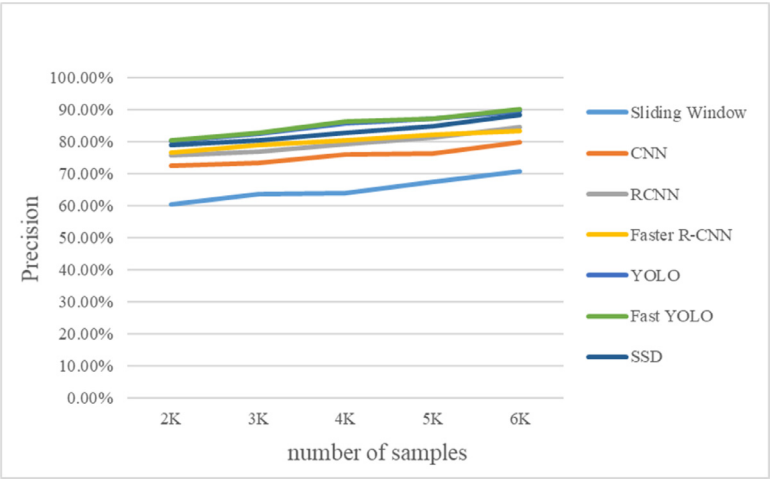

Fig. 5. The precision for the datasets.

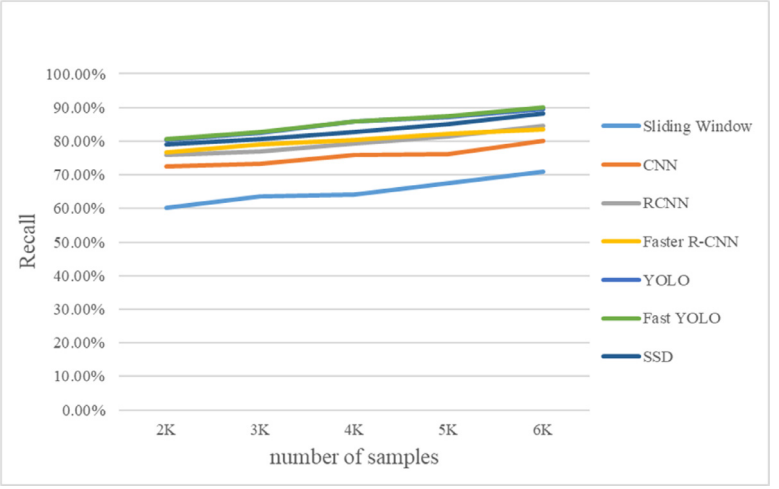

Fig. 6. The recall for the datasets.

Table 2
The precision rates of the categories.

Algorithm	car	trunk	bus	motor	person
Sliding window	65.35	70.26	68.65	68.34	72.16
CNN	81.88	75.38	79.98	80.21	72.31
RCNN	75.38	80.21	85.36	82.52	85.33
Faster R-CNN	76.24	84.32	85.32	74.81	80.25
YOLO	87.64	84.58	89.99	81.21	88.42
Fast YOLO	**87.88**	**84.76**	**90.12**	**80.87**	**88.64**
SSD	80.25	84.29	83.58	76.82	80.16

As shown in Fig. 5 and Fig. 6, the YOLO algorithm and other baseline methods tend to increase and stabilize in the precision rate and recall rates as the number of samples increases from 2000 to 6000. When the number of training samples is small, the model is underfit. As the number of samples increases, the model can better fit the sample distribution. When the number of samples reaches 6000, the model essentially begins to converge. The Fast YOLO algorithm always achieves the best results in the different cases.

From Table 2, we can see the precision rate for various categories for all the algorithms. Compared with the other algorithms, the Fast YOLO and YOLO algorithms have a significantly higher precision for all the categories. From the categories recognition of the objects, the precision rates of cars and buses are significantly higher than that of trunks. This result has a

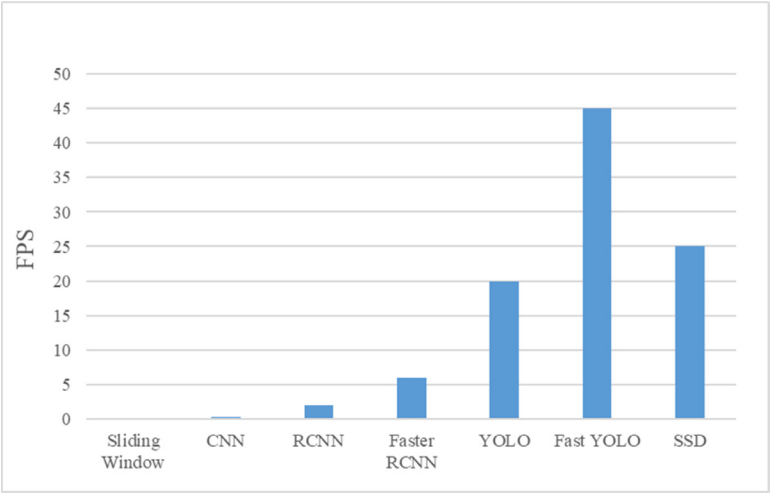

Fig. 7. Histogram of FPS.

lot to do with our datasets. Cars and buses appear more frequently than trunks in our monitoring video. With more sample tags input to train the model, which a more accurate model. can be implemented. The motors and persons are similar to cars and buses in that they appear frequently in our dataset.

From Fig. 7, we can see that the Fast YOLO algorithm can recognize 45 frames per second, far exceeding the ability of the other algorithms. Our method based on the YOLO network, greatly improves the detection speed. In general, the traffic monitoring video is 25 frames per second. Therefore, our method can implement the real-time detection of the traffic videos with the given equipment.

Currently, in the big data environment, we can collect massive samples of objects in advance to train the Fast YOLO, algorithm and retrain the trained model in the configuration files. When new samples are input, we can read the configuration file to generate the model directly and detect the objects, which can greatly reduce the detection time and the OD can be implemented in real-time.

In the IoT cloud environment, many sensors collect data and generate massive datasets for various application every day. The OD methods based on deep learning have a powerful feature extraction ability and can be applied to obtain high-value information in the IOT-Cloud environment. Our Fast YOLO algorithm can apply to the IoT cloud environment. We can use parallel computing method to train the model from the massive samples, and store the metamodel. Then we can use a depth-enhanced method to optimize the model continuously with update data from the Internet, which can enhance the detection accuracy. Combined with the previous reduction in the time consumption, our method can be applied in the IoT cloud environment to obtain valuable information and promote the development of businesses.

5. Conclusions and future work

In this paper, we present a fast object detection algorithm for video. We highlight the performance of our algorithm in terms of the speed of detection with outstanding efficiency of more than video frames per second and high accuracy of detection. Based on You Only Look Once (YOLO) network, we improve the convolution operation instead small convolution to reduce the amount of calculation and speed up detection significantly. With the image progressing, we eliminate the effects of environment and noise. Furthermore, we need not build any size of boxes as the bounding boxes of the background and reduce the analysis of the background.

We conduct experiments on videos from a vehicle monitoring dataset obtained from the Xiamen municipal transportation bureau and use the precision and recall rates and frames recognized per second (FPS) as the evaluation metrics. We use the image patches with marked labels from the vehicle monitoring video to train our model and obtain the object information. The results demonstrate that our method can perform better than the original YOLO algorithm and the other baseline methods. Our method can not only achieve a fast detection speed but also ensure the high accuracy. The number of images that our method can recognize exceeds that in video frames. Therefore, our method can implement object detection for videos in real time under good equipment conditions.

In the future, we will consider the problem of an imbalance between the positive and negative samples when most of samples are negative and adopt some methods such as resampling and voting to address this issue. In addition, we will consider multi-scale parallel processing for our model to improve the detection speed and optimize the structure of the network further, because we still face the problems of a large computational load and dense target objects.

The pseudo code of the Fast YOLO algorithm is as follows:

Algorithm 1
Fast YOLO.

```
Input: vehicle video parameters: α_coord, α_noobj
Procedure:
    Obtain each frame of the video
    Preprocess the images with the frame difference and background difference methods
    For each frame in the video
        Divide the image into S*S grids
        Build five different sizes of boxes as bounding boxes for the center of each grid
        Compute the probability that the boundary box contains the object
        Compute the probability that the center of the object falls within into the grid
        Compute the probability of the class for each bounding box
    End for
Output: boundary box (x, y. w, h, C), and the probability of classes
```

.

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.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.compeleceng.2019.05.009.

References

[1] Ramík DM, Sabourin C, Moreno R, Madani K. A machine learning based intelligent vision system for autonomous object detection and recognition. Appl Intell 2014;40(Mar. (2)):358–75.
[2] Zhao W-L, Ngo C-W. Flip-Invariant SIFT for copy and object detection. IEEE Trans Image Process 2013;22(Mar. (3)):980–91.
[3] Lu S. An image retrieval learning platform with authentication system. In: 2018 13th International Conference on Computer Science & Education (ICCSE); 2018. p. 1–5.
[4] Girshick R, Donahue J, Darrell T, Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition; 2014. p. 580–7.
[5] Lou X, Cui B. Boundedness and exponential stability for nonautonomous RCNNs with distributed delays. Comput Math Appl 2007;54(Aug. (4)):589–98.
[6] Sun X, Wu P, Hoi SCH. Face detection using deep learning: an improved faster RCNN approach. Neurocomputing 2018;299(Jul.):42–50.
[7] Li J, Su Z, Geng J, Yin Y. Real-time detection of steel strip surface defects based on improved YOLO detection network. IFAC-Pap 2018;51(21):76–81.
[8] Shu Y, Duan W, Jiao C. SSD evolution model in HF etching of fused silica optics. Optik (Stuttg) 2019;181(Mar.):372–7.
[9] Couteaux V, et al. Automatic knee meniscus tear detection and orientation classification with Mask-RCNN. Diagn Interv Imaging 2019;100(Apr. (4)):235–42.
[10] Yadav DK, Singh K. Adaptive background modelling technique for moving object detection in video under dynamic environment. Int J Spatio-Temporal Data Sci 2019;1(Jan. (1)):4–21.
[11] Sharma L, Lohan N. Performance analysis of moving object detection using BGS techniques in visual surveillance. Int J Spatio-Temporal Data Sci 2019;1(Jan. (1)):22–53.
[12] Yazdi M, Bouwmans T. New trends on moving object detection in video images captured by a moving camera: a survey. Comput Sci Rev 2018;28(May):157–77.
[13] Sharma L, Yadav DK. Histogram-based adaptive learning for background modelling: moving object detection in video surveillance. Int J Telemed Clin Pract 2017;2(Feb.):74–92.
[14] Yadav DK, Singh K. A combined approach of kullback–leibler divergence and background subtraction for moving object detection in thermal video. Infrared Phys Technol 2016;76(May):21–31.
[15] Sharma L, Yadav DK, Singh A. 'Fisher's linear discriminant ratio based threshold for moving human detection in thermal video'. Infrared Phys Technol 2016;78(Sep.):118–28.
[16] Tang P, Wang H, Kwong S. G-MS2F: googLeNet based multi-stage feature fusion of deep CNN for scene recognition. Neurocomputing 2017;225(Feb.):188–97.
[17] Shinde S, Kothari A, Gupta V. YOLO based human action recognition and localization. Procedia Comput Sci 2018;133:831–8.
[18] Ramya P, Rajeswari R. A modified frame difference method using correlation coefficient for background subtraction. Procedia Comput Sci 2016;93:478–85.
[19] Zhang J, Cao J, Mao B. Moving object detection based on Non-parametric methods and frame difference for traceability video analysis. Procedia Comput Sci 2016;91:995–1000.
[20] Morinaga A, Hara K, Inoue K, Urahama K. Classification between natural and graphics images based on generalized Gaussian distributions. Inf Process Lett 2018;138(Oct.):31–4.
[21] Zha S, Luisier F, Andrews W, Srivastava N, Salakhutdinov R. Exploiting image-trained CNN architectures for unconstrained video classification. In: Procedings of the British Machine Vision Conference 2015; 2015. p. 60.1–60.13.
[22] Lu S, Chen H, Zhou X, Wang B, Wang H, Hong Q. Graph-Based collaborative filtering with MLP. Math Probl. Eng 2018;2018(Dec.):1–10.
[23] Sudowe P, Leibe B. Efficient use of geometric constraints for sliding-window object detection in video. In: Crowley JL, Draper BA, Thonnat M, editors. Computer vision systems, 6962. Berlin, Heidelberg: Springer Berlin Heidelberg; 2011. p. 11–20.

Shengyu Lu: He was born in Ganzhou, Jiangxi, China in 1995. He received a B.S. degree from the Software School, Nanchang University, China, in 2017. He is currently pursuing an M.S. degree at the Software School, Xiamen University, China. He was a visiting researcher at Shanghai Jiao Tong University. His research interests include computer vision, natural language processing and recommendation systems.

Beizhan Wang: He was born in XianYang, Shangxi, China in 1965. He received a B.S., M.S. and Ph.D. degree in College from the Computer Science, Northwestern Polytechnical University, China. Since 2004, he has been a professor with the Software School, Xiamen University, China. His research interests include computer vision and data mining.

Hongji Wang: He received a Ph.D. degree in Computer Software and Theory from the Chinese Academy of Sciences, China. Since 2008, he has been an associate Professor with the Software School, Xiamen University, China. His research interests include data mining and information security.

Lihao Chen: He was born in Maanshan, Anhui, China in 1993. He is currently pursuing a Ph.D. at the Science School, Beijing University of Posts and Telecommunications, China. His research interests include computer vision and high throughput calculation.

Ma Linjian: She was born in Longyan, Fujian, China in 1981. She received a B.S. and M.S. degree in Xiamen University, China. She has been an engineer with the Software School, Xiamen University, China. Her research interests include data mining.

Xiaoyan Zhang: She was born in XiAn, Shangxi, China in 1970. She received a Ph.D. degree in College from the Computer Science, Northwestern Polytechnical University, China. She has been an associate Professor with the Tan Kah Kee College, Xiamen University, China. Her research interests include big data analysis.