electronics

MDPI

Article

Object Detection Algorithm Based on Improved YOLOv3

Liquan Zhao * and Shuaiyang Li

Key Laboratory of Modern Power System Simulation and Control & Renewable Energy Technology, Ministry of Education (Northeast Electric Power University), Jilin 132012, China; shuaiyangli@126.com

* Correspondence: zhaoliquan@neepu.edu.cn; Tel.: +86-150-432-01901

Received: 10 February 2020; Accepted: 23 March 2020; Published: 24 March 2020

Abstract: The 'You Only Look Once' v3 (YOLOv3) method is among the most widely used deep learning-based object detection methods. It uses the k-means cluster method to estimate the initial width and height of the predicted bounding boxes. With this method, the estimated width and height are sensitive to the initial cluster centers, and the processing of large-scale datasets is time-consuming. In order to address these problems, a new cluster method for estimating the initial width and height of the predicted bounding boxes has been developed. Firstly, it randomly selects a couple of width and height values as one initial cluster center separate from the width and height of the ground truth boxes. Secondly, it constructs Markov chains based on the selected initial cluster and uses the final points of every Markov chain as the other initial centers. In the construction of Markov chains, the intersection-over-union method is used to compute the distance between the selected initial clusters and each candidate point, instead of the square root method. Finally, this method can be used to continually update the cluster center with each new set of width and height values, which are only a part of the data selected from the datasets. Our simulation results show that the new method has faster convergence speed for initializing the width and height of the predicted bounding boxes and that it can select more representative initial widths and heights of the predicted bounding boxes. Our proposed method achieves better performance than the YOLOv3 method in terms of recall, mean average precision, and F1-score.

Keywords: deep learning; object detection; YOLOv3 method; k-means cluster

1. Introduction

Object detection is an important and challenging field in computer vision, one which has been the subject of extensive research [1]. The goal of object detection is to detect all objects and class the objects. It has been widely used in autonomous driving [2], pedestrian detection [3], medical imaging [4], industrial detection [5], robot vision [6], intelligent video surveillance [7], remote sensing images [8], etc.

In recent years, deep learning techniques have been applied in object detection [9]. Deep learning uses low-level features to form more abstractive high-level features, and to hierarchically represent the data in order to improve object detection [10]. Compared with traditional detection algorithms, the deep learning-based object detection method based has better performance in terms of robustness, accuracy and speed for multi-classification tasks.

Object detection methods based on deep learning mainly include region proposal-based methods, and those based on a unified pipeline framework. The former type of method firstly generates a series of region proposals from an input image, and then uses a convolution neural network to extract features from the generated regions and construct a classifier for object classes. The region-based convolution neural network (R-CNN) method [11] is the earlier method used to introduce convolution neural networks into the field of object detection. It uses the selection search method to generate region proposals from the input images, and uses a convolution neural network to extract features from the generated region proposals. The extracted features are used to train the support vector machine. Based on the R-CNN method, Fast R-CNN [12] and Faster R-CNN [13] were also proposed to reduce training time and improve the mean average precision. Although region proposal-based methods have higher detection accuracy, the structure of the method is complex, and object detection is time-consuming. The latter type of method (based on a unified pipeline framework) directly predicts location information and class probabilities of objects with a single-feed forward convolution neural network from the whole image, and does not require the generation of region proposals and post-classification. Therefore, the structure of the unified pipeline framework approach is simple and can detect objects quickly; however, it is less accurate than the region proposal-based approach. The two kinds of methods have different advantages and are suitable for different applications. In this paper, we mainly discuss the unified pipeline framework-based approach.

Researchers have proposed a wide range of unified pipeline framework-based methods in recent years, one of which is the You Only Look Once v2 (YOLOv2) method [14]. YOLOv2 uses batch normalization to improve convergence and prevent overfitting, and anchor boxes to predict bounding boxes, in order to increase the recall. Other innovations include a high-resolution classifier, direct location prediction, dimension cluster, and multi-scale training, all of which lend greater detection accuracy. Pedoeem and Huang recently proposed a shallow real-time detection method for Non-GPU computers based on the YOLOv2 method [15]; their method reduces the size of input image by half in order to speed up the detection speed, and removes the batch normalization of shallow layers in order to reduce the amount of model parameters. Shafiee et al. have proposed the Fast YOLO method, whereby YOLOv2 can be applied to embedded devices [16]: this employs an evolutionary deep intelligence framework to generate an optimized network architecture. The optimized network architecture can be used in the motion-adaption inference framework to speed up the detection process and thus reduce the energy consumption of the embedded device. Simon et al. have developed a complex-YOLO method [17], which uses a specific complex regression strategy to estimate multi-class 3D boxes in Cartesian space for detecting RGB images; the authors report a significant improvement in the speed of 3D object detection. Liu et al. have developed the Single Shot MultiBox Detector (SSD) method [18], which generates multi-scale feature maps in order to detect objects of different sizes. This method strikes a careful balance between speed and accuracy of detection, but the expression ability of the feature map is insufficient in the shallow layer. In order to enhance the expression ability of shallow feature maps, Fu et al. have proposed the Deconvolutional Single Shot Detector (DSSD) method [19], which uses the ResNet extraction network (generating better features) [20], a deconvolution layer and skip connection in order to improve the expression ability of shallow feature maps. In order to improve the detection accuracy of the SSD method for small objects, Qin et al. have proposed a new SSD method based on the feature pyramid [21]. Their method applies a deconvolution network in the high-level of the feature pyramid in order to extract semantic information, and expands the convolution network so as to learn low-level position information. Their method constructs a multi-scale detection structure so as to improve the detection accuracy of small objects. Redmon and Farhadi have proposed applying the YOLOv3 method for using binary cross-entropy loss for class predictions [22], which employs scale prediction to predict boxes at different scales, and thus improves the detection accuracy with regard to small objects.

In this Section 1, we have reviewed recent developments related to object detection. In Section 2, we outline the concepts and processes of the YOLOv3 object detection method, and in Section 3 we describe our proposed method. In Section 4, we illustrate and discuss our simulation results.

2. YOLOv3

The YOLOv3 method considers object detection as a regression problem. It directly predicts class probabilities and bounding box offsets from full images with a single feed forward convolution neural network. It completely eliminates region proposal generation and feature resampling, and encapsulates all stages in a single network in order to form a true end-to-end detection system.

The YOLOv3 method divides the input image into $S \times S$ small grid cells. If the center of an object falls into a grid cell, the grid cell is responsible for detecting the object. Each grid cell predicts the position information of B bounding boxes and computes the objectness scores corresponding to these bounding boxes. Each objectness score can be obtained as follows:

$$C_i^j = P_{i,j}(Object) * IOU_{pred}^{truth} \tag{1}$$

whereby C_i^j is the objectness score of the jth bounding box in the ith grid cell. $P_{i,j}(Object)$ is merely a function of the object. The IOU_{pred}^{truth} represents the intersection over union (IOU) between the predicted box and ground truth box. The YOLOv3 method uses binary cross-entropy of predicted objectness scores and truth objectness scores as one part of loss function. It can be expressed as follows:

$$E_1 = \sum_{i=0}^{S^2} \sum_{j=0}^{B} W_{ij}^{obj} [\hat{C}_i^j \log(C_i^j) - (1 - \hat{C}_i^j) \log(1 - C_i^j)] \tag{2}$$

whereby S^2 is the number of grid cells of the image, and B is the number of bounding boxes. The C_i^j and $\hat{C}_i^j$ are the predicted abjectness score and truth abjectness score, respectively.

The position of each bounding box is based on four predictions:t_x, t_y, t_w, t_h, on the assumption that (c_x, c_y) is the offset of the grid cell from the top left corner of the image. The center position of final predicted bounding boxes is offset from the top left corner of the image by (b_x, b_y). Those are computed as follows:

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \end{aligned} \tag{3}$$

whereby $\sigma()$ is a sigmoid function. The width and height of the predicted bounding box are calculated thus:

$$\begin{aligned} b_w &= p_w e^{t_w} \\ b_h &= p_h e^{t_h} \end{aligned} \tag{4}$$

whereby p_w, p_h are the width and height of the bounding box prior. They are obtained by dimensional clustering.

The ground truth box consists of four parameters (g_x, g_y, g_w and g_h), which correspond to the predicted parameters b_x, b_y, t_w and t_h, respectively. Based on (3) and (4), the truth values of $\hat{t}_x$, $\hat{t}_y$, $\hat{t}_w$ and $\hat{t}_h$ can be obtained as follows:

$$\begin{aligned} \sigma(\hat{t}_x) &= g_x - c_x \\ \sigma(\hat{t}_y) &= g_y - c_y \\ \hat{t}_w &= \log(g_w / p_w) \\ \hat{t}_h &= \log(g_h / p_h) \end{aligned} \tag{5}$$

The YOLOv3 method uses the square error of coordinate prediction as one part of loss function. It can be expressed as follows:

$$\begin{aligned} E_2 \ &= \sum_{i=0}^{S^2} \sum_{j=0}^{B} W_{ij}^{obj} [(\sigma(t_x)_i^j - \sigma(\hat{t}_x)_i^j)^2 + (\sigma(t_y)_i^j - \sigma(\hat{t}_y)_i^j)^2] \\ &+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} W_{ij}^{obj} [((t_w)_i^j - (\hat{t}_w)_i^j)^2 + ((t_h)_i^j - (\hat{t}_h)_i^j)^2] \end{aligned} \tag{6}$$

3. Proposed Method

Before developing the YOLOv3 model, it is necessary to determine the width and height of bounding box priors (p_w and p_h in (4) and (5), respectively), as they directly affect the performance of the YOLOv3 method. In the YOLOv3 method, it uses k-means clustering algorithm to select the representative width and height of bounding box priors to avoid consuming much time in adjusting the width and height. The complexity of k-means clustering method is expressed as $O(n^{kd})$ for the data based on d dimension and k cluster centers, whereby n is the number of data. The larger the dataset, the more time-consuming the modelling process. In addition, the YOLOv3 method is sensitive to the initial cluster center. To overcome this problem, we apply the AFK-MC^2 method [23] in order to estimate the width and height of bounding box priors.

For the purpose of convenience, we suppose that the width and height of ground truth boxes are $\varphi = \{(w_1, h_1), (w_2, h_2), \cdots, (w_n, h_n)\}$. Firstly, we randomly select a couple of width and height values (w_i, h_i) as one initial cluster center c_1 from the set φ. To obtain the other $k-1$ initial cluster centers, we repeat the following procedure for $k-1$ times in order to build $k-1$ Markov chains with length m. The procedure begins by computing all proposal distributions, or $q(\varphi_j)$. Each $q(\varphi_j)$ is calculated as follows:

$$q(\varphi_j) = \frac{d(\varphi_j, c_1)}{\sum\limits_{i=1}^{n} d(\varphi_i, c_1)} + \frac{1}{n} \tag{7}$$

whereby $\varphi_j \in \varphi$, $j = 1, 2, \cdots, n$, c_1 is the first initial cluster center. The AFK-MC^2 method directly uses the Euclidean distance to compute the distance between two parameters. In this paper, we use the intersection over union method to compute distance. This is expressed as:

$$d(\varphi_j, c_1) = \min(1 - IOU(\varphi_j, c_1)) \tag{8}$$

whereby $IOU(\varphi_j, c_1)$ is the intersection over union betwee j-th bounding boxes $\varphi_j = (w_j, h_j)$ and the first initial cluster center $c_1 = (w_i, h_i)$. It is used to measure the overlap between φ_j and c_1. If the $IOU(\varphi_j, c_1)$ is larger, it means that there are more overlaps between φ_j and c_1. $d(\varphi_j, c_1)$ is distance from φ_j to the initial cluster center c_1.

Secondly, we randomly select a couple of width and height values φ_i as an initial point of the Markov chain. For the other points in the same Markov chain, we select a candidate φ_t from set φ based on proposal distribution $q(\varphi)$ from set φ, and compute the sampling probability $p(\varphi_t)$ as follows:

$$p(\varphi_t) = \frac{d(\varphi_t, C)}{\sum\limits_{i=1}^{n} d(\varphi_i, C)} \tag{9}$$

whereby C is the set of selected cluster centers, $d(\varphi_t, C)$ is minimum value of $d(\varphi_t, c_i)(i = 1, 2, \cdots, k)$. We compute the distance from candidate φ_t cluster center in set C using equation (8), and select the minimum value as $d(\varphi_t, C)$. Based on the sampling probability and proposal distribution of φ_t, we can compute the acceptance probability that φ_t can be accepted as the next point in the Markov chain. This can be expressed as follows:

$$\alpha(\varphi_t, \varphi_{t-1}) = \min(1, \frac{p(\varphi_t)}{p(\varphi_{t-1})} \times \frac{q(\varphi_{t-1})}{q(\varphi_t)}) = \min(1, \frac{d(\varphi_t, C)}{d(\varphi_{t-1}, C)} \times \frac{q(\varphi_{t-1})}{q(\varphi_t)}) \tag{10}$$

whereby φ_{t-1} is the current point in the Markov chain. If the acceptance probability $\alpha(\varphi_t, \varphi_{t-1})$ is greater than the threshold $N \in Unif(0, 1)$, then φ_t can be accepted as the next point in the Markov chain. Otherwise, φ_{t-1} is also used as the next point in the Markov chain. Therefore, we can construct a Markov chain with length m. Based on the above procedure, we can construct $k-1$ different Markov chains with length m and use the last point of every Markov chains as the initial cluster center. The obtained initial cluster centers of the Markov chains and the randomly set initial cluster center form k initial cluster centers $C = [c_1, c_2, \cdots c_k]$.

In constructing a Markov chain, each candidate point requires to calculate the distance between the candidate point and the selected cluster centers, if the selected candidate point is a point that has been selected as the cluster center, the distance between the candidate point and the selected cluster centers is 0, and the acceptance probability of the candidate point is 0. The candidate point will not be used as a point in the Markov chain. This avoids using the selected cluster center as one point of Markov chain in constructing different Markov chains. Therefore, the selected initial cluster centers are different.

Thirdly, we randomly select S points from set φ to form set Φ, and compute the distances between the each point in the Φ and k cluster centers. If one point is closest to one cluster center, we assign the point to the cluster at which the cluster center is located. Therefore, we can construct k clusters using S points and k cluster centers. We use the all-points mean in every cluster as the new cluster center, i.e.,

$$p_{wi} = \frac{1}{|H|}\sum_{j=1}^{H} w_{i,j} \tag{11}$$

$$p_{hi} = \frac{1}{|H|}\sum_{j=1}^{H} h_{i,j} \tag{12}$$

whereby (p_{wi}, p_{hi}) is the new cluster center in the new cluster ϑ_i, $i = 1, 2, \cdots, k$, and is the number of points in the cluster ϑ_i, $\vartheta_i = [(w_{i,1}, h_{i,1}), (w_{i,2}, h_{i,2}), \cdots, (w_{i,H}, h_{i,H})]$. Next, we reselect S points from set φ and compute the distances between the points and k new cluster centers. We also construct the new cluster according the computed distance and use equations (11) and (12) to obtain a new cluster center. If the new cluster center is invariant, we can obtain the final cluster center. The flowchart of our proposed method is shown in Figure 1.

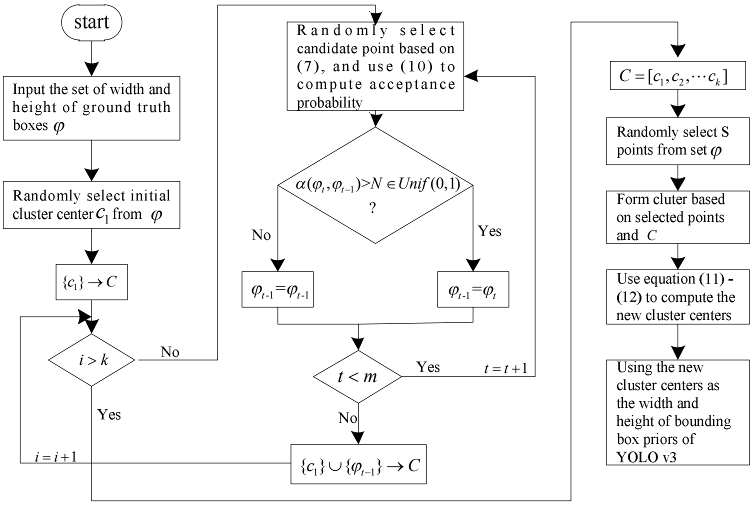

start
Input the set of width and height of ground truth boxes φ
Randomly select initial cluster center c_1 from φ
$\{c_1\} \rightarrow C$
$i > k$
No
Yes
$i = i + 1$
Randomly select candidate point based on (7), and use (10) to compute acceptance probability
$\alpha(\varphi_t, \varphi_{t-1}) > N \in Unif(0,1)$?
No
Yes
$\varphi_{t-1} = \varphi_{t-1}$
$\varphi_{t-1} = \varphi_t$
$t < m$
Yes
$t = t + 1$
No
$\{c_1\} \cup \{\varphi_{t-1}\} \rightarrow C$
$C = [c_1, c_2, \cdots c_k]$
Randomly select S points from set φ
Form cluter based on selected points and C
Use equation (11) - (12) to compute the new cluster centers
Using the new cluster centers as the width and height of bounding box priors of YOLO v3

Figure 1. Flowchart of our proposed method.

The k-means method used in YOLOv3 randomly selects k couples of width and height values as initial cluster centers, so the k-means method is sensitive to the initial cluster center. Secondly, it requires computing the distances between all points and k cluster centers, and this will consume a large amount of time for large-scale detection dataset in adjusting cluster centers. Our proposed method only randomly selects a couple of width and height values as one initial cluster, and then selects $k-1$ cluster centers by constructing $k-1$ different Markov chains of length m. Therefore, our proposed method reduces the sensitivity to the initial cluster centers. Besides, we only randomly select S points instead of all points from set φ, and compute distance between the S points and k cluster centers. It requires a shorter running time compared to the k-means method, especially for large-scale detection datasets. Therefore, using the YOLOv3 method, we can use the cluster centers as the width and height of bounding box priors so as to realize object prediction.

4. Simulation and Discussion

In this paper, we used two datasets, PASCAL VOC (Pattern Analysis, Statical Modeling and Computational Learning Visual Object Classes) and MS COCO (Microsoft Common Objects in Context) [24]. The PASCAL VOC is a standardized dataset for image classification and object detection. The images contained in the PASCAL VOC dataset are from real scenes. These objects can be divided into twenty classes. There are 9963 images in the datasets which contain 24,640 annotated objects. The MS COCO is an authoritative and important benchmark tool used in the field of object recognition and detection. It is also used in the YOLOv3 method. It contains 117,264 training images and more than 5000 testing images with 80 classes. The Ubuntu 18.04 system is used for the simulations, and the method employs an Intel Xeon E5-2678 v3 CPU. The GPU is NVIDIA GeForce GTX 1080Ti, and the deep learning framework is PyTorch. The size of each image is 416×416. The learning rate, momentum and decay are 0.001, 0.9 and 0.0005, respectively. The number of training images is 64 per batch. Our YOLOv3 model uses three output feature maps with different scales to detect differently sized objects, and we have tested it on 3, 6, 9, 12, 15 and 18 candidate cluster centers.

We use Avg IOU (Average Intersection over Union) between the boxes that are generated by using cluster centers and all ground truth boxes in order to measure the performance of each cluster method. This can be expressed as follows:

$$\text{Avg IOU} = \frac{1}{N}\sum_{j=1}^{N}(\max_{i\in[1,\cdots,k]}\frac{C_i \cap \Psi_j}{C_i \cup \Psi_j}) \tag{13}$$

whereby N is the number of ground truth boxes (that is, $\Psi = [\Psi_1, \Psi_2, \cdots, \Psi_N]$), k is the number of cluster centers, and C_i is the box generated using the width and height in the cluster centers. The larger the Avg IOU value, the better the clustering effect. We also use recall, mean value of average precision (mAP), and F1-score to measure the performance of different methods. The recall is the ratio of the number of objects that are successfully detected and the number of samples that contain the detected objects. The mAP is the mean value of average precision for the detection of all classes. The F1-score is the harmonic mean of precision and recall, the maximum value is 1 and the minimum value is 0.

Below, we compare the performance of the proposed cluster method and AFK-MC^2 method in terms of estimating the initial width and height of predicting boxes. The length of the Markov chain used in simulations for two methods is 200. We have also tried to increase and decrease the length of the Markov chain. When the length is increased, the Avg IOU and running time are also increased. When the length is decreased, the Avg IOU and running time are also decreased. For simplicity, we used 200 as the length of the Markov chain that is also used in [23]. On the MS COCO datasets, the Avg IOUs obtained by our proposed method and the AFK-MC^2 method are shown in Figure 2: it can be seen that our proposed method has a larger Avg IOU than that of the AFK-MC^2 method for a different number of cluster centers. This means that the proposed method has better performance than the AFK-MC^2 method in terms of estimating the initial width and height of predicting boxes.

In order to compare the detection performance of the original YOLOv3 method and that based on our proposed cluster method, we use the same cluster center number that is used in the former method. The cluster center number is 9, and the results on the MS COCO and PASCAL VOC datasets are shown in Tables 1 and 2, respectively. In Table 1, the Avg IOU values for proposed cluster method and k-means method used in YOLOv3 are 60.44 and 59.88, respectively. The running time for proposed cluster method and k-means method used in YOLOv3 are 1183.083s and 3.972 s, respectively. The running time of k-means used in YOLOv3 is about 297 times that of our proposed cluster method. This shows that the proposed cluster has a larger Avg IOU and smaller running time than the k-means method used in YOLOv3. In Table 2, the Avg IOU values for proposed cluster method and k-means method used in YOLOv3 are 67.34 and 67.45, respectively. The running time for the proposed cluster method and the k-means method used in YOLOv3 are 19.337 s and 0.239 s, respectively. The running time of k-means used in YOLOv3 is about 81 times that of our proposed cluster method. This also shows that the proposed cluster has a larger Avg IOU and smaller running time than the k-means method used in YOLOv3. The k-means method used in YOLOv3 requires computing the distance between all points and k cluster centers. This will consume a large amount of time for large-scale dataset detection. While we only randomly select S points instead of all points from set φ, and compute distance between the S points and k cluster centers. Therefore, it requires a smaller running time compared with the k-means method, especially for large-scale detection dataset. The size of the MS COCO dataset is larger than PASCAL VOC dataset, so the difference of running time between our proposed method and the k-means method used in YOLOv3 is larger for the MS COCO dataset than for the PASCAL VOC dataset.

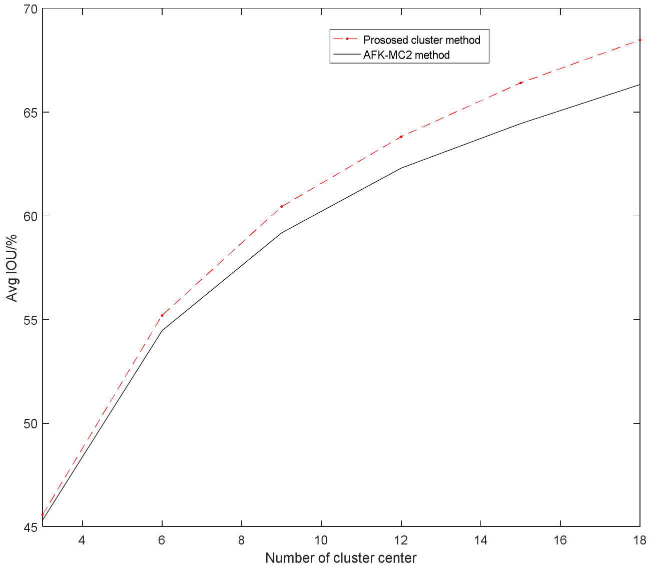


Figure 2. Avg IOU of the proposed cluster method and the AFK-MC^2 method on the MS COCO dataset.

Table 1. Comparison between original YOLOv3 and proposed cluster method on the MS COCO dataset: Avg IOU (%) and running time (s).

Method	Avg IOU	Running Time
k-means used inYOLOv3	59.88	1183.038
Proposed cluster method	60.44	3.972

Table 2. Comparison between original YOLOv3 and proposed cluster method on the PASCAL VOC dataset: Avg IOU (%) and running time (s).

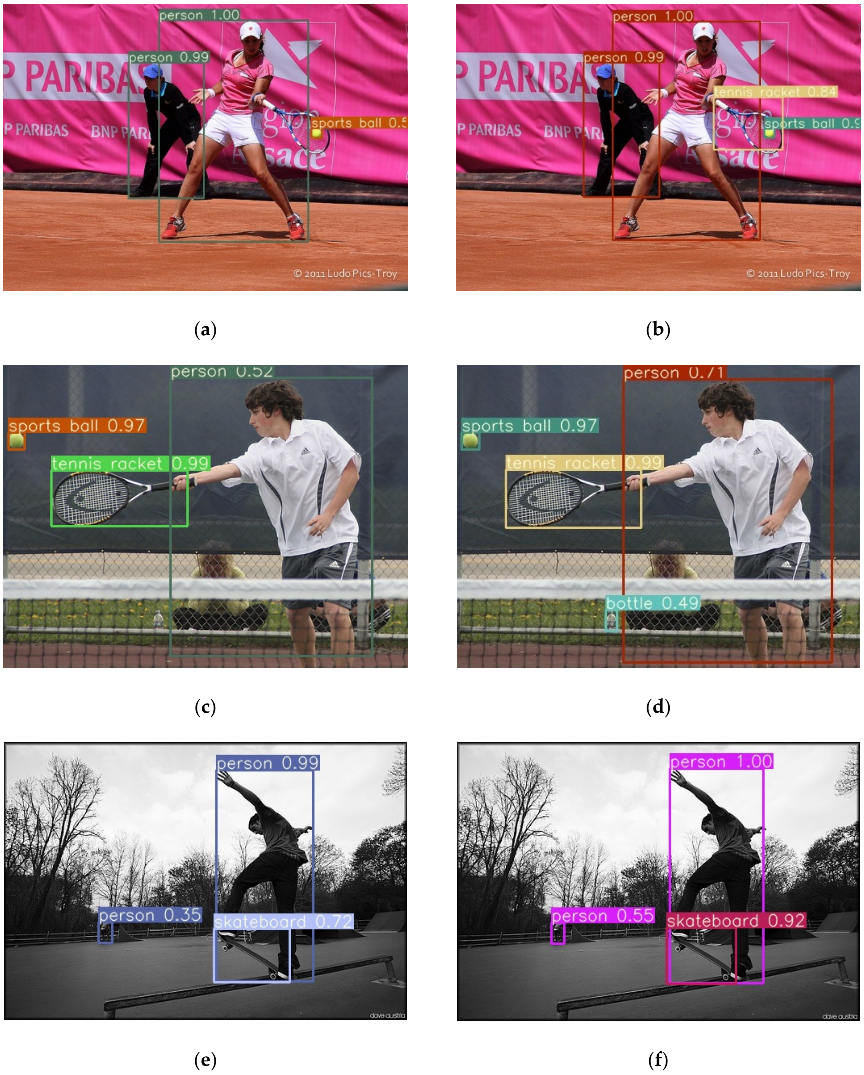
Method	Avg IOU	Running Time
k-means used inYOLOv3	67.34	19.377
Proposed cluster method	67.45	0.239

Table 3 shows the comparisons between the original YOLOv3 and improved YOLOV3 method (based on our proposed cluster method) on the MS COCO dataset: it can be seen that our YOLOv3 method produces larger recall, mAP and F1-score values, and therefore has better detection accuracy than the original YOLOv3 method.

Table 3. Comparison between YOLOv3 and proposed cluster method on the MS COCO dataset: Recall (%), mAP (%) and F1-score (%).

Method	Recall	mAP	F1-Score
YOLOv3	70.5	53.2	60.6
Proposed cluster method	71.3	53.3	61.0

We also randomly selected five images from the test sets of the MS COCO dataset in order to test the performance of small object detection; the object detection results are shown in Figure 3. Subfigures (a), (c), (e), (g) and (i) show object detection results generated using the original YOLOv3 method, and subfigures (b), (d), (f), (h) and (j) show the object detection results generated using our proposed method. For the first image, the YOLOv3 method detected three objects, while our proposed method detected four objects (subfigures (a) and (b)). With the second image, the YOLOv3 method detected three objects, while our proposed method detected four objects (subfigures (c) and (d)). For the first and second image, our proposed method detected more objects, and it has higher scores in terms of detecting small objects. With the third image, the YOLOv3 method and our proposed method detected three objects (subfigures (e) and (f)), and our proposed method has higher scores in terms of detecting objects, especially small objects such as people in the distance and skateboards. With the fourth image, the YOLOv3 method and our proposed method detected two objects (subfigures (g) and (h)), and our proposed method has higher scores in terms of detecting objects, especially cups. With the fifth image, the YOLOv3 method and our proposed method detected three objects (subfigures (i) and (j)), and our proposed method has higher scores in terms of detecting objects, especially the giraffe in the distance. These ten sub-figures indicate that our proposed method has better performance in terms of detecting objects, especially for some small objects such as sports balls, tennis rackets, bottles, people in the distance, skateboards, cups, a giraffe in the distance, etc.

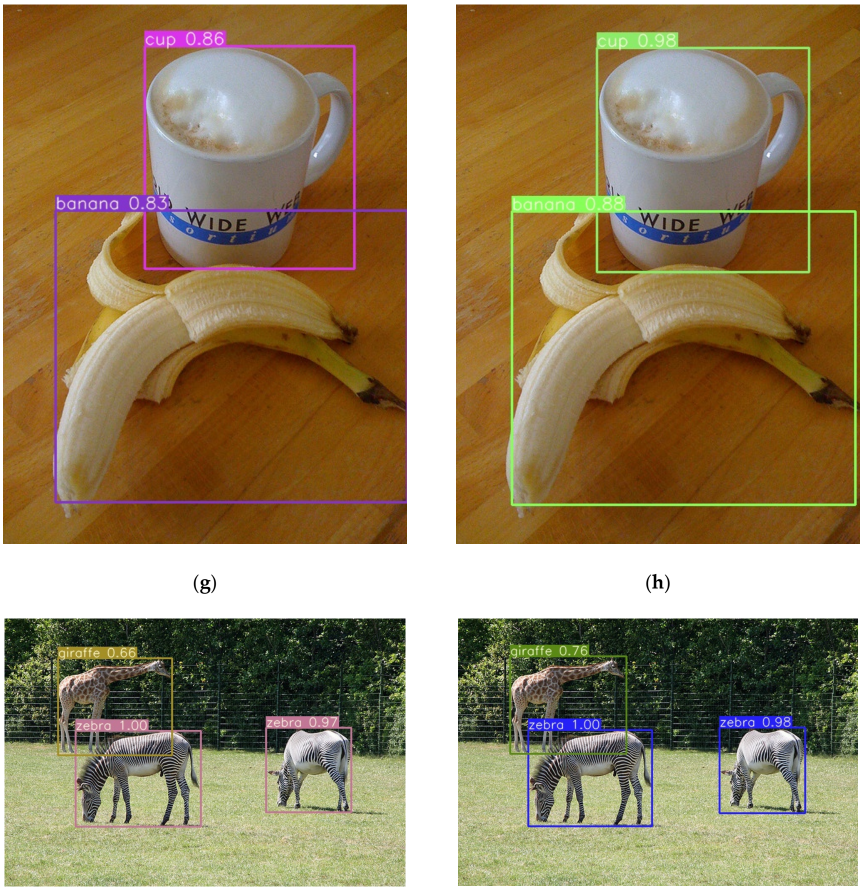


(**a**) (**b**)

(**c**) (**d**)

(**e**) (**f**)

Figure 3. *Cont.*



(**g**) (**h**)

(**i**) (**j**)

Figure 3. Object detection results using the YOLOv3 method (subfigures (**a**), (**c**), (**e**), (**g**) and (**i**), and object detection results using our proposed method (subfigures (**b**), (**d**), (**f**), (**h**) and (**j**)).

5. Conclusions

This paper proposes a new method for initializing the width and height of predicted bounding boxes. Our proposed method has a larger Avg IOU and smaller running time on the MS COCO dataset. The Avg IOU is 60.44%, which is 0.56% higher than original YOLOv3 method, and the running time is 1/297 that of the original YOLOv3 method. For the PASCAL VOC dataset, the average IOU is 67.45%, which is 0.13% higher than original YOLOv3 method, and the running time is 1/81 that of the original YOLOv3 method. It exhibits better performance in terms of initializing the width and height of predicted bounding boxes, as well as in terms of choosing the representative initial width and height. Besides, we randomly selected some images from the test set of the MS COCO dataset for detection. The object detection results indicate that our proposed method detected more objects in some test images. It also has better performance in terms of detecting small objects. Our proposed method also outperforms the original YOLOv3 method in terms of recall, mean average precision, and F1-score.

Author Contributions: Conceptualization, formal analysis, investigation, and writing the original draft were performed by L.Z. and S.L. Experimental tests were performed by S.L. All authors have read and approved the final manuscript.

Funding: This research was funded by National Natural Science Foundation of China (61271115) and Science and Technology Innovation and Entrepreneurship Talent Cultivation Program of Jilin (20190104124).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hanchinamani, S.R.; Sarkar, S.; Bhairannawar, S.S. Design and Implementation of High Speed Background Subtraction Algorithm for Moving Object Detection. In Proceedings of the IEEE International Conference on Advances in Computing, Communications and Informatics, Jaipur, India, 21–24 September 2016; pp. 367–374.
2. Chen, X.; Ma, H.; Wan, J.; Li, B.; Xia, T. Multi-view 3D Object Detection Network for Autonomous Driving. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Hawaii, HI, USA, 21–26 July 2017; pp. 6526–6534.
3. Mao, J.; Xiao, T.; Jiang, Y.; Cao, Z. What Can Help Pedestrian Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Hawaii, HI, USA, 21–26 July 2017; pp. 3127–3136.
4. Christ, P.F.; Kaissis, G.; Ettlinger, F.; Kaissis, G.; Schlecht, S.; Ahmaddy, F.; Grün, F.; Menze, B.; Valentinitsch, A.; Ahmadi, S.-A.; et al. SurvivalNet: Predicting patient survival from diffusion weighted magnetic resonance images using cascaded fully convolutional and 3D Convolutional Neural Networks. In Proceedings of the IEEE International Conference on International Symposium on Biomedical Imaging, Melbourne, Australia, 18–21 April 2017; pp. 839–843.
5. Weimer, D.; Scholz-Reiter, B.; Shpitalni, M. Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection. *CIRP Ann.* **2016**, *65*, 417–420. [CrossRef]
6. Senicic, M.; Matijevic, M.; Nikitovic, M. Teaching the methods of object detection by robot vision. In Proceedings of the IEEE International Convention on Information and Communication Technology, Electronics and Microelectronics, Opatija, Croatia, 21–25 May 2018; pp. 558–563.
7. Sreenu, G.; Durai, M. Intelligent video surveillance: A review through deep learning techniques for crowd analysis. *J. Big Data* **2019**, *6*, 48–75. [CrossRef]
8. Li, K.; Cheng, G.; Bu, S.; You, X. Rotation-Insensitive and Context-Augmented Object Detection in Remote Sensing Images. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 2337–2348. [CrossRef]
9. Zhou, X.; Gong, W.; Fu, W.; Du, F. Application of deep learning in object detection. In Proceedings of the IEEE/ACIS 16th International Conference on Computer and Information Science, Wuhan, China, 24–26 May 2017; pp. 631–634.
10. Zhao, Z.; Zheng, P.; Xu, S.; Wu, X. Object Detection with Deep Learning: A Review. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3212–3232. [CrossRef] [PubMed]
11. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 24–27 June 2014; pp. 125–138.
12. Girshick, R. Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 127–135.
13. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans Pattern Anal Mach Intell.* **2017**, *39*, 1137–1149. [CrossRef] [PubMed]
14. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. *IEEE Trans. Pattern Anal.* **2017**, *29*, 6517–6525.
15. Pedoeem, J.; Huang, R. YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers. In Proceedings of the IEEE International Conference on Big Data, Seattle, WA, USA, 10–13 December 2018; pp. 2503–2510.
16. Shafiee, M.J.; Chywl, B.; Li, F.; Wong, A. Fast YOLO: A Fast You Only Look Once System for Real-time Embedded Object Detection in Video. *J. Comput. Vis. Image Syst.* **2017**, *3*, 171–173. [CrossRef]
17. Simon, M.; Milz, S.; Amende, K.; Gross, H.M. Complex-YOLO: Real-time 3D Object Detection on Point Clouds. In Proceedings of the IEEE International Conference on European Conference on Computer Vision, Munich, Germany, 8–14 September 2018; pp. 197–209.
18. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single shot multibox detector. In Proceedings of the IEEE International Conference on European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016; pp. 21–37.
19. Fu, C.Y.; Liu, W.; Ranga, A.; Tyagi, A.; Berg, A.C. DSSD: Deconvolutional single shot detector. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Hawaii, HI, USA, 21–26 July 2017; pp. 1–8.

20. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
21. Pinle, Q.; Chuanpeng, L.; Jun, C.; Chai, R. Research on improved algorithm of object detection based on feature pyramid. *Multimed. Tools Appl.* **2019**, *78*, 913–927.
22. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *IEEE Trans. Pattern Anal.* **2018**, *15*, 1125–1131.
23. Bachem, O.; Lucic, M.; Hassani, H.; Krause, A. Fast and Provably Good Seedings for k-Means. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–8 December 2016; pp. 55–63.
24. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. In Proceedings of the IEEE International Conference on European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; pp. 740–755.