Hindawi
Scientific Programming
Volume 2021, Article ID 4447271, 11 pages
https://doi.org/10.1155/2021/4447271

Research Article

A Decision Support System for Power Components Based on Improved YOLOv4-Tiny

Yangyang Tian,[1] Wandeng Mao,[1] Shaoguang Yuan,[1] Diming Wan,[1] and Yuanhui Chen[2]

[1]*State Grid Henan Electric Power Research Institute, Zhengzhou 450000, China*
[2]*Shandong University of Science and Technology, Qingdao 266000, China*

Correspondence should be addressed to Wandeng Mao; 1054354086@qq.com

Received 31 August 2021; Accepted 18 November 2021; Published 28 December 2021

Academic Editor: Zeeshan Pervez

The traditional image object detection algorithm applied in power inspection cannot effectively position power components, and the accuracy of recognition is low in scenes with some interference. In this research, we proposed a data-driven power detection method based on the improved YOLOv4-tiny model, which combined the ResNet-D module and the adjusted Res-CBAM to the backbone network of the existing YOLOv4-tiny module. We replaced the CSPOSANet module in the YOLOv4-tiny backbone network with the ResNet-D module to reduce the FLOPS required by the model. At the same time, the adjusted Res-CBAM whose feature fusion ways were replaced with stacking in the channels was combined as an auxiliary classifier. Finally, the features of five different receptive scales were used for prediction, and the display of the results was optimized by merging the prediction boxes. In the experiment, 57134 images collected on the power inspection line were processed and labeled, and the default anchor boxes were re-clustered, and the speed and accuracy of the model were evaluated by video and validation set of 3459 images. Processing multiple pictures and videos collected from the power inspection projects, we re-clustered the default anchor box and tested the speed and accuracy of the model. The results show that compared with the original YOLOv4-tiny model, the accuracy of our method that can position objects under occlusion and complex lighting conditions is guaranteed while the detection speed is about 13% faster.

1. Introduction

As an infrastructure related to the national economy and people's livelihood, the power system is very important to modern society [1, 2]. Therefore, it is a very essential task to monitor whether the power components work safely and reliably. The traditional manual detection method requires people to work for a long time, and its effect is related to the experience and working status of the staff. The system is not capable of continuous monitoring and is less reliable. To maintain the normal operation of power equipment and improve the safety and reliability of the power system, the related algorithms have begun to be applied in the power industry [3, 9].

In power inspection, the background of the images is complex and has high timeliness requirements. Traditional target detection algorithms mainly use artificially designed features, such as Haar classifier [10, 11], cascaded classifier [12], SIFT (scale-invariant feature transform) [3, 4, 13–15], HOG (histogram of oriented gradients) [16], DPM (deformable part model) [17], SVM (support vector machine) [18], and so on, or combine them [19, 20]. These methods' [3–20] disadvantages are as follows: (1) Their feature extractors are manually selected and not robust to changes in complex practical application scenarios. (2) Their region selection strategies are based on sliding windows, with high time complexity and window redundancy, which affects the accuracy and speed of detections.

With the rapid development of neural networks and artificial intelligence technology, there have been a large number of deep learning-based object detection algorithms applied to power inspection [5–9]. These methods can be roughly divided into two categories. One is two-stage methods that search for the boxes first and then perform classification and regression, whose accuracy is higher but speed is slow, such as the R-CNN series algorithm based on Region Proposal (R-CNN [21], fast R-CNN [22], and faster

R-CNN [23]. The other is the one-stage methods, using only one CNN to predict the categories and positions of different targets, such as SSD [24], RetinaNet [25], RefineDet [26], and YOLO (You Only Look Once) [27] series mentioned later. Although this type of method is lower in accuracy, it can achieve real-time detection.

There is a potential contradiction that higher accuracy is incompatible with the faster detection speed. In spite of the intrinsic difficulty in object detection algorithm, the application of UAV power real-time inspection is challenging owing to the interconnection and overlapping of the power components, illumination changes, complex background, diversity, and randomness of relative position and motion between target and camera.

Aiming at the real-time detection requirements of actual power project scenes, we propose a data-driven power detection method based on the improved YOLOv4-tiny. In detail, the Res-CBAM block was reconstructed and the short-cut in ResNet-D block was adjusted, and then the two blocks were combined in the backbone of YOLOv4-tiny. The results show that our method can adapt to scenes with multiple obstructions and is faster while ensuring accuracy.

2. Materials and Methods

2.1. UAV Image Dataset. The dataset in this paper was collected from the UAV images of power inspection, including 53675 in the training set and 3459 in the verification set. The power inspection images and videos were taken by the UAV with high resolution, diverse random angles, and fast video transition. Mark the key connection parts in the dataset, including three categories: wire connecting tower (dx_gt), tower connecting insulator (gt_jyz), and insulator connecting wire (jyz_dx). When labeling, try to cover the target area with the label box as much as possible and reduce other background information about the label box at the same time (Figure 1).

2.2. YOLO. YOLO was proposed by Redmon et al. [27]. It is a one-stage real-time object detection algorithm. Compared with the two-stage model, such as the R-CNN series, the YOLO series model simplifies the object detection task into a single regression task of boxes and prediction of classes, and the position and category of the object boxes are inferred from the input. Due to the abandonment of the default anchor, the YOLO series model is faster than the two-stage model in detection speed and can achieve real-time detection, but it also sacrifices a certain model accuracy and the ability to recognize small objects.

The YOLO series model inspection process is described in Figure 2. In the training phase, the image is divided into certain grids, and each grid has some default anchor boxes. Among these anchor boxes corresponding to the grid including the center of the real box, the one that most matches the real box—with the largest IOU—is responsible for detecting the real box and obtaining the information about the real boxes, that is, position information, confidence probability, and classification information. When the image is input into the YOLO model, the positioning loss l_{loc}, the object confidence loss l_{conf}, and the classification loss l_{cls} can be calculated, so the total loss function $L = \lambda_1 l_{\text{loc}} + \lambda_2 l_{\text{conf}} + \lambda_3 l_{\text{cls}}$, where $\lambda_1, \lambda_2, \lambda_3$ are the weights. Finally, the model parameters are updated through backpropagation. In the prediction phase, the parameters trained model are not updatable, and some corresponding categories and confidence levels of some possible prediction boxes are obtained after the image input to the model. Finally, the results are obtained through the process of nonmaximum suppression (NMS).

YOLOv4 [28] is an improved version of YOLOv3 [29] of the YOLO series. It combines many small tricks on the basis of the former, such as CutMix, mosaic, DropBlock regularization, SPP, and so on. Further, YOLOv4-tiny [30] is a simplified version of YOLOv4. The number of features used for classification and regression detection is simplified to two, and the parameters are simplified from 60 million to 6 million with the application of the CSPOSANet (CSP) which is similar to the ResNet module. When the size of the input is $608 * 608 * 3$, the model structure is shown in Figure 3.

2.3. Improved YOLOv4-Tiny

2.3.1. ResNet-D. ResNet-D [31] is a modification to the ResNet [32] architecture shown in Figure 4. The motivation is that the $1 * 1$ convolution with stride 2 for the downsample ignores 3/4 of the input in the unmodified ResNet. Thus, this is modified such that downsampling can be done by the next $3 * 3$ convolutions in one path and by maxpooling in the short-cut path, and the loss of features caused by the simultaneous occurrence of $1 * 1$ convolution and stride 2 is avoided.

FLOPS [33] (floating-point operations per second) are used to measure hardware performance. When the input feature map is $152 * 152 * 64$, FLOPS required by the ResNet-D and CSPOSANet modules are shown in Table 1.

It can be known that the FLOPS required by the ResNet-D module are about one-tenth of CSPOSANet. In this article, we consider replacing the CSPOSANet module with ResNet-D to accelerate the model.

In particular, the short-cut in the ResNet-D means element addition, but in this paper, it means stacking on the channels, i.e., the output is $2n$-dimensional in the channel when the input is n-dimensional in the channel.

2.3.2. Res-CBAM. Woo et al. [34] proposed CBAM (convolutional block attention module) (Figure 5). Channel attention module will calculate a channel weight $M_C(F)$ whose size is (channel, 1, 1), that is, the values of height $*$ width pixels in one channel are multiplied by the same weight, mainly focusing on the information in different input channel information; spatial attention module will calculate a spatial weight $M_C(F')$ whose size is (1, height, width), which means that the values of different channels at the same pixel position are multiplied by the same weight, mainly focusing on the different location information entered. The specific calculation process is

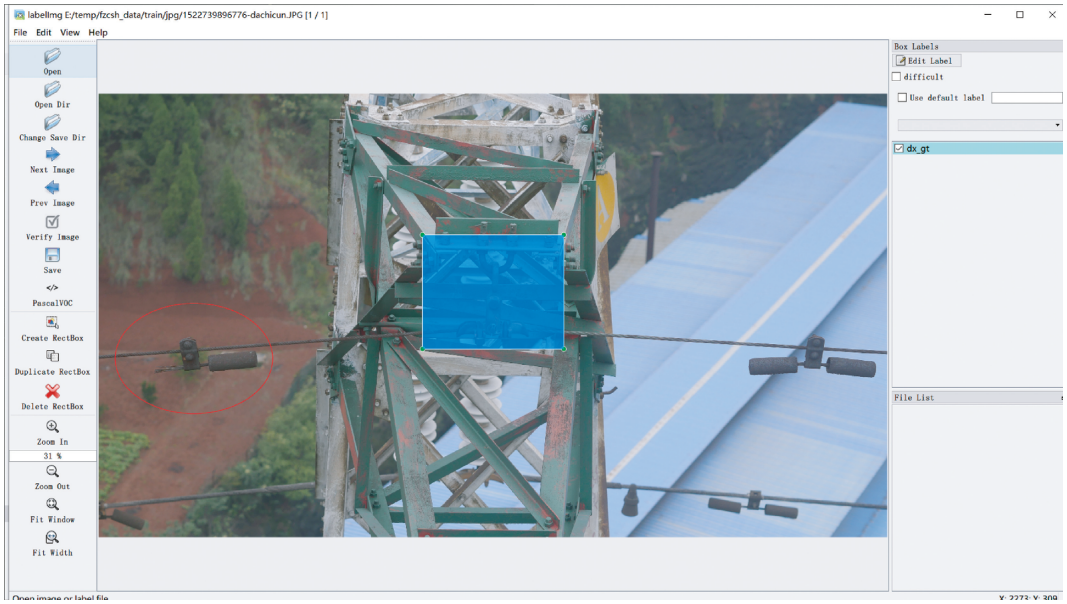

Figure 1: An example of labeling.

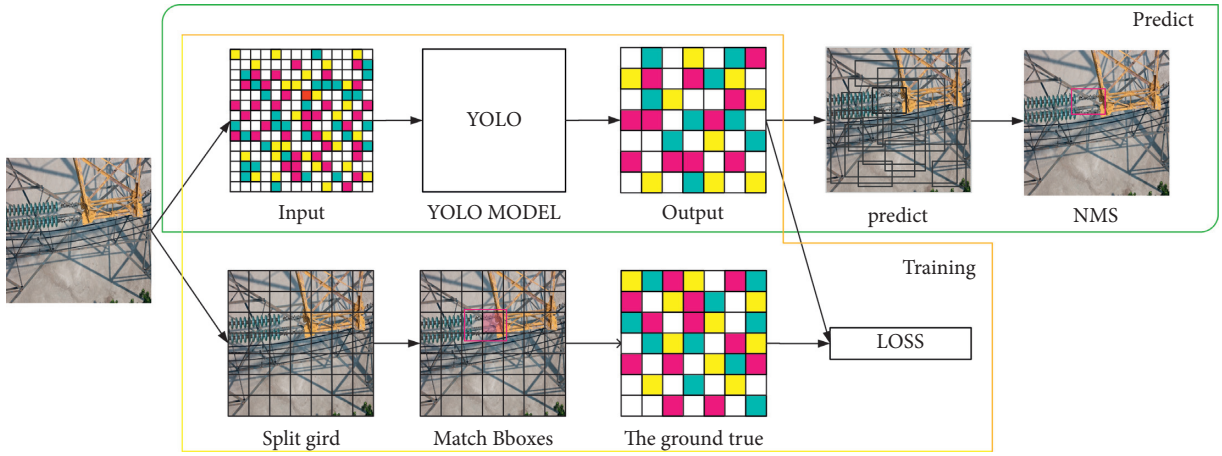

Figure 2: The process of YOLO series method.

$$F' = M_C(F) \otimes F,$$
$$F'' = M_S(F') \otimes F', \tag{1}$$

where F and F' are the inputs of the channel attention module and spatial attention module, respectively, and F' and F'' are the results of the channel attention module and spatial attention module, respectively. $M_C(\cdot)$ and $M_S(\cdot)$ represent channel attention and spatial attention, respectively.

We designed Res-CBAM based on CBAM, replacing the channel attention module with an upsampled average pooling layer and adjusting fusion method of features from point multiplication to stacking in the channel (Figure 6).

$$F' = M_C(F) \otimes F,$$
$$F'' = M_S(F') \otimes F'. \tag{2}$$

To combine the features extracted by the adjusted Res-CBAM and prevent the gradient explosion caused by the network module being too deep, two convolution operations and one convolution operation are added before and after the module, and the residual method is used to connect them.

2.3.3. Improved Model. In this article, we consider replacing the CSP module in the original YOLOv4-tiny with the ResNet-D module to speed up the model, but the ResNet-D module is small and may affect the accuracy. To balance the accuracy, the designed Res-CBAM is incorporated into the backbone network to assist in detection. The proposed backbone and model structure are shown in Figures 7 and 8.

2.4. Processing of Inputs and Outputs

2.4.1. Processing of Inputs. Taking into account the IO limitation on data training, the pictures are preprocessed to a certain extent in advance. We crop the image based on the

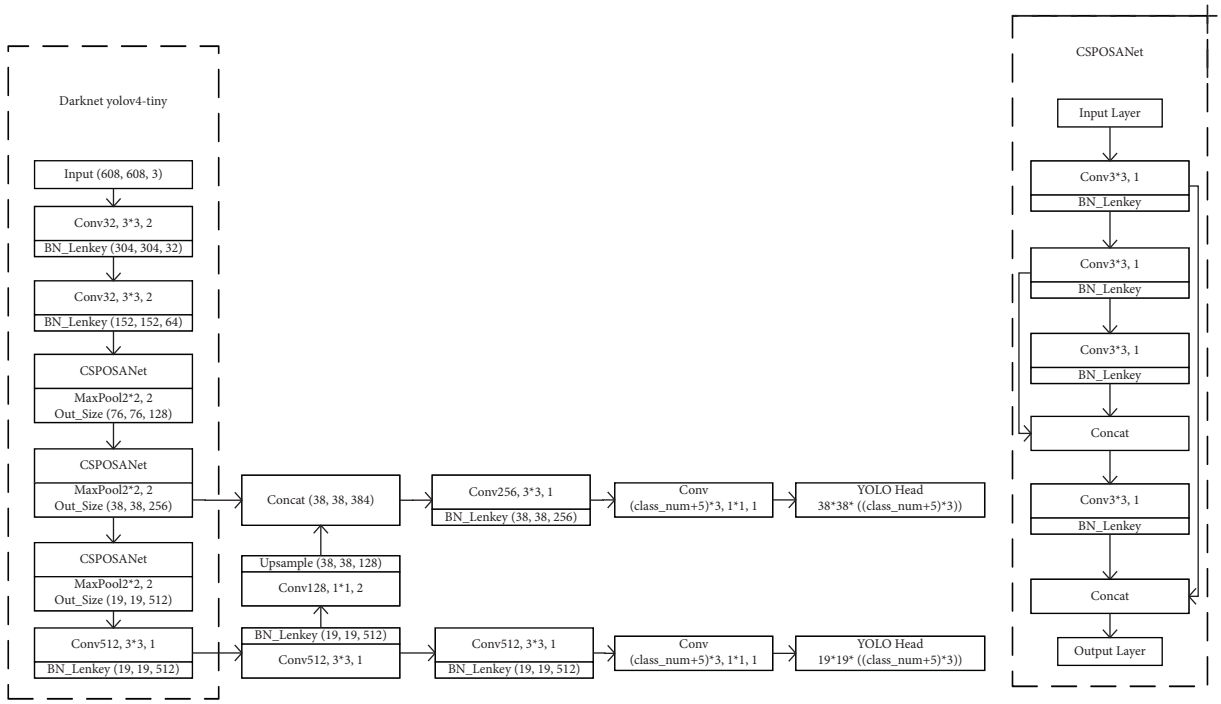

Figure 3: The structure of YOLOv4-tiny.

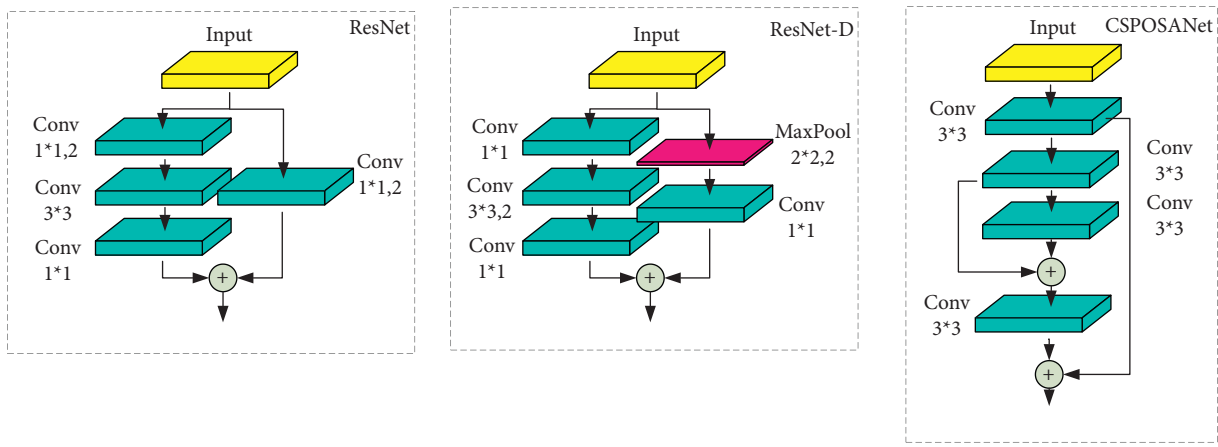

Figure 4: The structure of ResNet module, ResNet-D module, and CSPOSANet module.

Table 1: FLOPS.

Model	FLOPS
CSPOSANet	1.58 ∗ 109
ResNet-D	1.375 ∗ 108

object category and scale it to reduce the size of the input image.

$$w_{\text{new}} = w * \min\left(\frac{l}{w}, \frac{l}{h}\right),$$
$$h_{\text{new}} = h * \min\left(\frac{l}{w}, \frac{l}{h}\right), \tag{3}$$

where the maximum side length size is set to l, the width and height of the zoomed image are w and h, and the width and height of the zoomed image are w_{new} and h_{new}.

2.4.2. Processing of Outputs. Although the output of the model has been processed by nonmaximum suppression (NMS) to reduce the redundant result, in the actual scene test, the information about multiple boxes was output with some detection boxes overlapped which interferes with the normal inspection work. Therefore, we set the rule that for any prediction boxes ($b\text{box}_{\langle \text{cls}\, i,p\rangle}$ and $b\text{box}_{\langle \text{cls}\, i,p\rangle}$) in the category $cls\, i$, when the overlap (IOU) between them is greater than or equal to 0.5, it can be considered that the overlap of the two detection boxes is too large, so the two boxes are merged into a larger one (3).

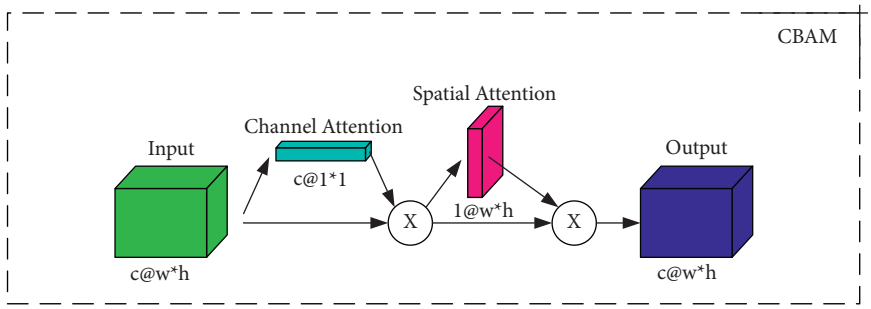

Figure 5: The structure of CBAM.

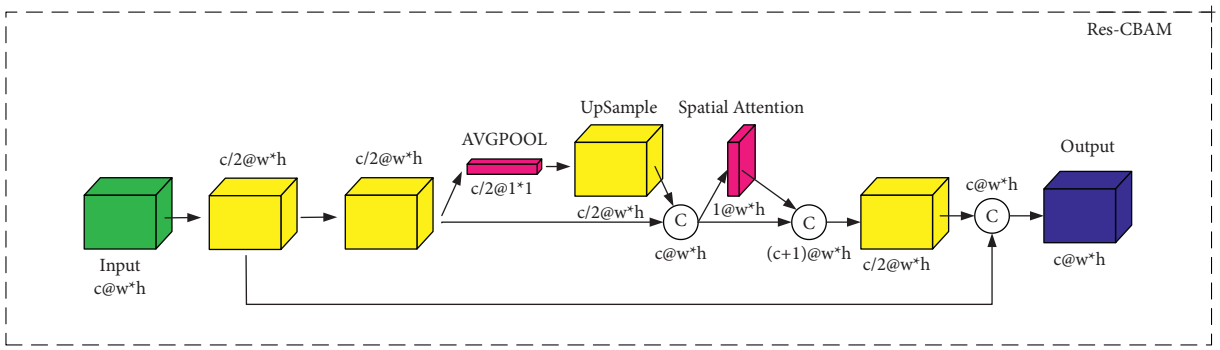

Figure 6: The Res-CBAM.

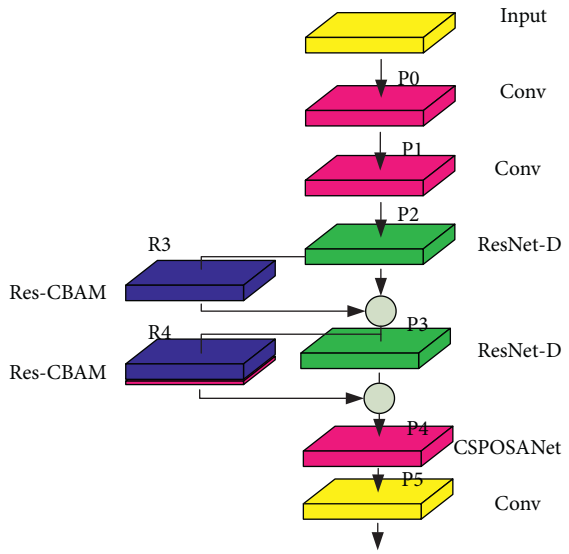

Figure 7: The proposed backbone.

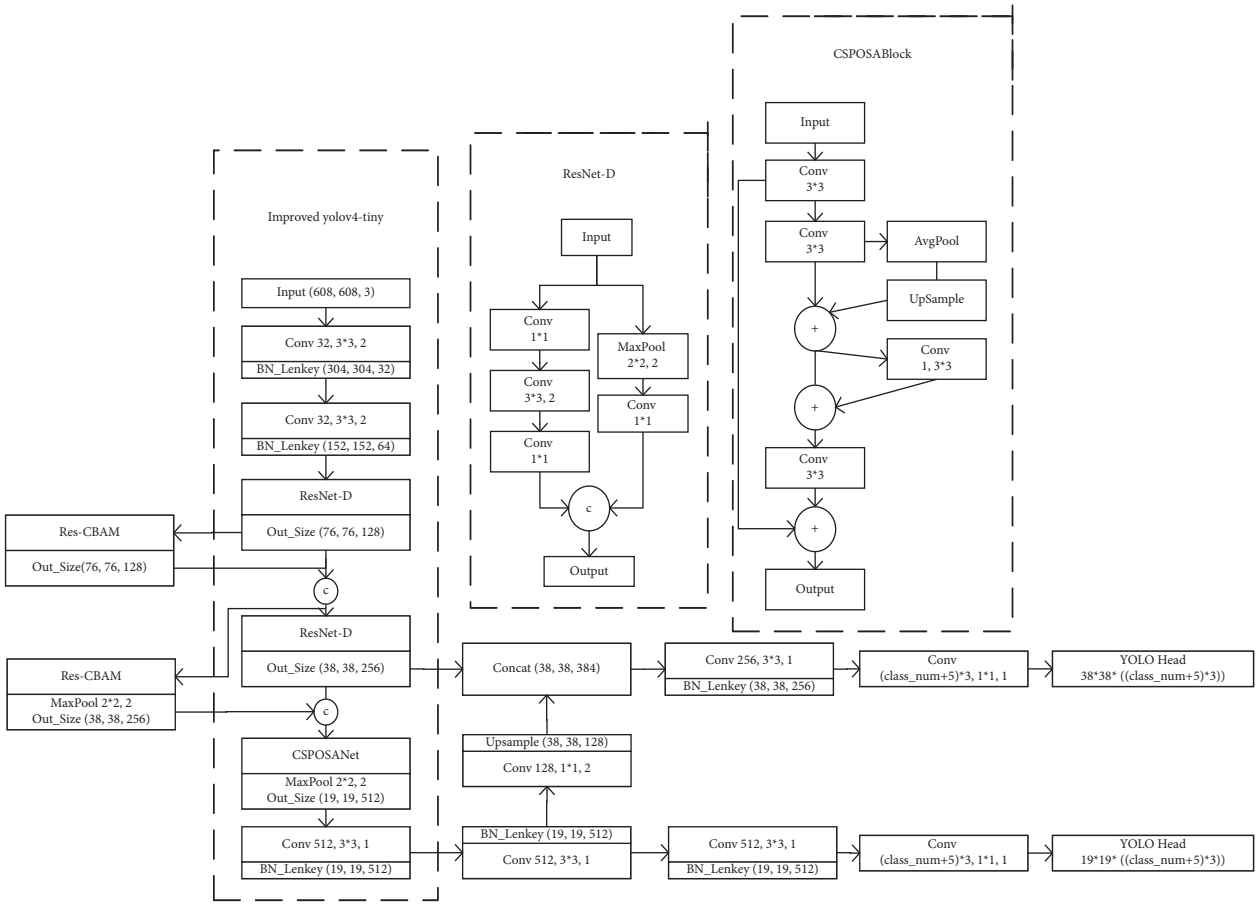

Figure 8: The proposed model.

$$b\text{box}_{\langle \text{cls}\, i,p\rangle}, b\text{box}_{\langle \text{cls}\, i,p\rangle} = \begin{cases} \text{if } iou\left(b\text{box}_{\langle \text{cls}\, i,p\rangle}, b\text{box}_{\langle \text{cls}\, i,p\rangle}\right) \geqq 0.5, & A, \\ \text{then}\ , \quad b\text{box}_{\langle \text{cls}\, i,p\rangle}, & b\text{box}_{\langle \text{cls}\, i,p\rangle}, \end{cases} \tag{4}$$

where

$$A = \begin{bmatrix} \text{cls}\, i \\ \min\left(x \min\left(b\text{box}_{\langle \text{cls}\, i,p\rangle}\right), x \min\left(b\text{box}_{\langle \text{cls}\, i,p\rangle}\right)\right) \\ \min\left(y \min\left(b\text{box}_{\langle \text{cls}\, i,p\rangle}\right), y \min\left(b\text{box}_{\langle \text{cls}\, i,p\rangle}\right)\right) \\ \max\left(x \max\left(b\text{box}_{\langle \text{cls}\, i,p\rangle}\right), x \max\left(b\text{box}_{\langle \text{cls}\, i,p\rangle}\right)\right) \\ \max\left(y \max\left(b\text{box}_{\langle \text{cls}\, i,p\rangle}\right), y \max\left(b\text{box}_{\langle \text{cls}\, i,p\rangle}\right)\right) \end{bmatrix}. \tag{5}$$

3. Experiment

3.1. k-Means Clustering. YOLO's default anchor box is an empirical value obtained through k-means clustering on the COCO dataset of 80 categories. However, applying this default value to datasets collected in the projects may affect the convergence speed and accuracy of the model. Therefore, it is necessary to perform a re-clustering analysis on the object boxes of the dataset. The features of $19 * 19$ and $38 * 38$ are obtained when the input size is $608 * 608$, and each scale corresponds to three anchor box values, a total of 6 anchors (Table 2). The feature of $19 * 19$ corresponds to a larger receptive field, i.e., the large-scale anchor frame, while the feature of $38 * 38$ corresponds to a smaller receptive field, i.e., the small-scale anchor frame. In this paper, the feature of $19 * 19$ corresponds to the anchor frames 3, 4, and 5, and the feature of $38 * 38$ corresponds to the anchor frames 1, 2, and 3.

3.2. Network Training. The training environment information in this article is shown in Table 3. The learning rate setting strategy is shown in Figure 9, and the learning rate l of x iteration can be expressed as follows:

Table 2: The anchors.

No.	1	2	3	4	5	6
Default anchors	10, 14,	23, 27,	37, 58,	81, 82,	135, 169	344, 319
Resetting anchors	19, 25,	45, 57,	68, 150,	120, 88,	159, 242,	326, 380

Table 3: Experimental environment.

Item	Version
OS	Ubuntu 16.04.12
Kernel	Linux version 4.4.0-203-generic
CPU	Intel(R) i9-9900K CPU @ 3.60 GHz
GPU	2 ∗ GTX1080Ti
GCC	Version 5.4.0
Nvidia driver	460.32.03
CUDA	V10.2.89
OpenCV	3.2.0
Framework	Darknet

$$l = \begin{cases} l_r * \left(\frac{x}{b_n}\right)^n, & x < b_n, \\ l_r, & b_n \le x < s1, \\ 0.1 * l_r, & s1 \le x < s2, \\ 0.01 * l_r, & s2 \le x < \max\_\text{step}, \end{cases} \tag{6}$$

where the basic learning rate is l_r, the initial learning rate change node is b_n, the initial learning rate change coefficient is n ($n = 4$), the maximum number of iterations is max _step, and the learning rate change nodes are $s1$, $s2$ ($s1, s2 = 0.8 * \max\text{step}, 0.9 * \max\_\text{step}$).

Online data enhancement is uniformly used in training, and the strategies adopted are mosaic and Gaussian_noise. The mosaic strategy means combining multiple images to enrich the object information and detect the background of the object, and Gaussian_noise means adding Gaussian noise.

4. Results and Discussion

We used mAP (mean value of average precision), AvgFPS (average frames per second), and GPU (graphic processing unit) memory occupancy as indicators to evaluate the model, where mAP refers to the average accuracy of all categories detected by the model, and AvgFPS refer to the average number of frames that can be detected per second, and GPU memory usage refers to the GPU memory size occupied by the model when it is running. In this article, the YOLOv4-tiny model was compared with our method, and we tested the performance of the models trained with the default anchor and the anchors re-clustered (Table 4).

Table 4 shows the following:

(1) Whether it is the model we proposed or the YOLOv4-tiny, the accuracy of the model has decreased by about 10% with the re-clustered anchor.

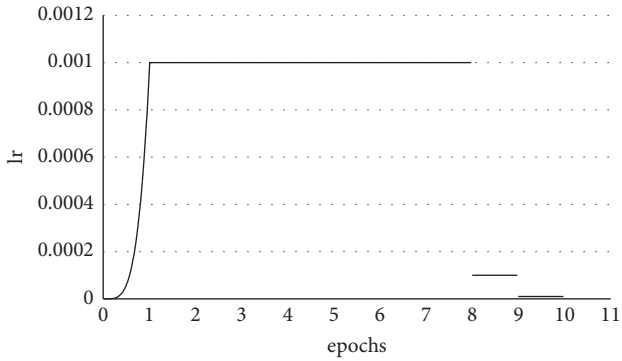

Figure 9: The change of learning rate during training.

Table 4: The performance of the trained model with the default anchor and the re-clustered anchors.

Method	AvgFPS	mAP
YOLOv4-tiny (default anchors)	218	0.6511
YOLOv4-tiny (resetting anchors)	212	0.5532
Proposed method (default anchors)	**248**	0.6401
Proposed method (resetting anchors)	241	0.5394

Bold value indicates our model is 13% faster than the YOLOv4-tiny.

(2) The accuracy of YOLOv4-tiny is higher, and the AvgFPS of our method are higher. The accuracy of our method is about 1% lower than that of YOLOv4-tiny, but in terms of recognition speed, the model we proposed is 13% faster than the YOLOv4-tiny, and the increase is much greater than the decrease in model accuracy.

(3) We test the models on the GTX1080Ti with the 1080P inspection video, and the results showed that both can reach a speed of more than 210 AvgFPS, which is beyond the real-time detection standard.

Figures 10 and 11 show the results of six pictures of the power line, and some conclusions can be drawn as follows:

(1) As shown in Figures 10(a), 10(b), 11(a), and 11(b), both methods can better recognize the object on the image, even if the object is located at the boundary of the image, and the method we proposed has higher confidence in predicting the frame than that of YOLOv4-tiny. The confidence levels of the box are larger, but in Figures 10(a) and 11(a), YOLOv4-tiny predicts a correct box more than the method we proposed. As shown in Figures 10(a) and 11(a), the confidence levels of the boxes predicted by our method are 83.29, 48.53, 53.46, and 64.65, respectively, and the confidence levels of the boxes predicted by YOLOv4-tiny are 36.69, 56.98, 37.19, and 30.56, respectively; as shown in Figures 10(b) and 11(b), the confidence levels of the boxes predicted by our method are 98.72, 61.49, and 45.51, and the confidence levels of the boxes predicted by YOLOv4-tiny are 97.80, 73.83, and 37.20, respectively.

(2) As shown in Figures 10(c), 10(d), 11(c), and 11(d), the model we proposed can better position the

Figure 10: The results of the model we proposed. (a) dx_gt: 83.29, 48.53, jyz_dx: 53.46, 64.65. (b) jyz_dx: 98.72, gt_jyz: 61.49, 45.51. (c) gt_jyz: 47.85, jyz_dx: 55.63. (d) gt_jyz: 45.16, jyz_dx: 31.54. (e) gt_jyz: 37.55, jyz_dx: 74.90. (f) gt_jyz: 42.52, 37.81, jyz_dx: 43.36.

partially occluded jyz_dx object, but YOLOv4-tiny does not detect the occluded object, indicating that the model we proposed is better in the ability of recognizing occlusion object.

(3) As shown in Figures 10(e), 10(f), 11(e), and 11(f), when the shooting conditions are not good (backlighting and underexposure), both methods can position the object within a certain limit. It can be proven that the performance of our method is generally similar to YOLOv4-tiny. In terms of the ability to recognize occluded targets, the model we proposed is better.

When the model input size is 608 ∗ 608, the weight file sizes of the YOLOv4-tiny model and the model we proposed are 22.4 MB and 25.1 MB, respectively. In actual deployment (Table 5), the memory occupied by YOLOv4-tiny in a single GPU when training and testing is 1163 MB and 501 MB, respectively. The memory occupied in a single GPU when training and testing the model we proposed is 1150 MB and 485 MB, respectively. When the model input size is 608 ∗ 608, the weight file sizes of the YOLOv4-tiny model and the model we proposed are 22.4 MB and 25.1 MB, respectively. In actual employment (Table 6), the memory occupied by YOLOv4-tiny in a single GPU when training

(a) (b) (c) (d) (e) (f)

Figure 11: The results of YOLOv4-tiny. (a) dx_gt: 36.69, 56.58, jyz_dx: 37.19, 20.56, gt_jyz: 65.45. (b) jyz_dx: 97.80, gt_jyz: 73.83, 37.20. (c) gt_jyz: 38.30. (d) gt_jyz: 46.42, jyz_dx: 77.99. (e) gt_jyz: 37.02, jyz_dx: 89.62. (f) gt_jyz: 77.56.

Table 5: The GPU occupancy.

Process	Method	GPU occupancy (MB)
Train	YOLOv4-tiny	1163
	Proposed method	**1150**
Predict	YOLOv4-tiny	501
	Proposed method	**485**

Bold indicates that our model takes about 2% less on the GPU occupancy.

and testing is 1163 MB and 501 MB, respectively. The memory occupied in single GPU when training and testing of the model we proposed is 1150 MB and 485 MB, respectively. Based on the above analysis, the weight file of the model we proposed is about 12% larger than that of YOLOv4-tiny, but it takes about 2% less on the GPU

Table 6: The AvgFPS in different devices.

Method	Jetson TX2	Jetson Xavier NX
YOLOv4-tiny	33.8	45.2
Proposed method	**38.9**	**52.1**

Bold indicates that the speed of the model we proposed is increased by about 15% than YOLOv4-tiny in the computing device of Jetson TX2 or Jetson Xavier NX.

occupancy. At the same time, the detection speed of the model we proposed is about 13% faster, suitable for deployment on some edge computing devices with weak GPU performance.

Jetson TX2 and Jetson Xavier NX are both lightweight AI development boards with CUDA computing units provided by NVIDIA. They are edge computing modules for embedded devices that can be applied to real-time detection of actual power inspection. We use the same 1080P video collected in the actual power line to test the AvgFPS of YOLOv4-tiny and the model we proposed on Jetson TX2 and Jetson Xavier NX devices (Table 6). The AvgFPS of YOLOv4-tiny and the model we proposed are 33.8 and 38.9 in Jetson TX2 and 45.2 and 52.1 in Jetson Xavier NX, both of which can achieve real-time detection. At the same time, the speed of the model we proposed is increased by about 15% than YOLOv4-tiny, which is slightly higher than that of 1080Ti.

5. Conclusion

In this paper, an improved YOLOv4-tiny network for power inspection that combines the ResNet-D block and the adjusted Res-CBAM block is proposed to solve the problem of inefficient power inspection target identification. The result shows that the accuracy was ensured, and the speed is increased by 13%. The model we proposed achieves the 0.6401 mAP and the 248 FPS on 1080Ti. The object can be identified with a high degree of confidence in the occlusion and bad lighting conditions; even real-time detection (above 35FPS) can be achieved with a smaller GPU occupancy rate on weaker GPU devices (Jetson TX2 and Jetson Xavier NX).

Data Availability

The data presented in this study are available on request from the corresponding author.

Disclosure

The funders had no role in the design of the study; in the collection, analyses, or interpretation of the data; in the writing of the manuscript; or in the decision to publish the results.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was funded by the Science and Technology Project of SGCC (Research and application of audio-visual active perception and collaborative cognitive technology for smart grid operation and maintenance scenarios) (5600-202046347A-0-0-00). The authors would like to thank Electric Power Research Institute of State Grid Henan Electric Power Company and Beijing Imperial Image Intelligent Technology Co. for support in the work in this paper.

References

[1] Z. Xiao, H. Liu, V. Havyarimana, T. Li, and D. Wang, "Analytical study on multi-tier 5G heterogeneous small cell networks: coverage performance and energy efficiency," *Sensors*, vol. 16, no. 11, 2016.

[2] F. Zeng, L. Qiao, X. Zhu, H. Vincent, and J. Bai, "A price-based optimization strategy of power control and resource allocation in full-duplex heterogeneous macrocell-femtocell networks," *IEEE Access*, vol. 6, Article ID 42004, 2018.

[3] C. He, "Based on the improvement of SIFT algorithm, transmission line hang-up fault detection," *International Electronic Elements*, vol. 29, no. 2, pp. 99–102+107, 2021.

[4] H. Zhang, "An insulator image recognition method based on SIFT- BOW model," *Laser Journal*, vol. 41, no. 11, pp. 96–99, 2020.

[5] S. U. I. Yu, "Review on Mounted UAV for Transmission Line Inspection. Power System Technology," pp. 1–15, 2020.

[6] Y. Jiyu, C. Cifa, and G. Guoqiang, "Aerial insulator detection of transmission line based on improved Faster," *Computer Engineering*, vol. 47, pp. 1–8, 2021.

[7] Z. Zhenbing, J. Zhigang, L. Yanxu et al., "Overview of visual defect detection of transmission line components," *Journal of Image and Graphics*, vol. 11, pp. 2545–2560, 2021.

[8] W. Bai, X. Zhang, X. Zhu et al., "Electric power inspection image enhancement based on E-FCNN," *Electric power*, vol. 54, pp. 1–8, 2020.

[9] H. Zheng, H. Wang, H. Zhou, X. Zhang, and H. Zhao, "Real-time track planning for power pole inspection based on hybrid algorithm," *Electric power*, pp. 1–8, 2020.

[10] H. Wu, Y. Cao, H. Wei, and T. Zhuang, "Face recognition based on haar like and euclidean distance," *Journal of Physics: Conference Series*, vol. 1813, pp. 12–36, 2021.

[11] Y. Cheng, "A face detection based on Haar and skin color segmentation algorithms," *Journal of Ordnance Equipment Engineering*, vol. 42, no. 1, pp. 254–258, 2021.

[12] H. Xu, "Pedestrian detection algorithm based on cascading feature classifier," *Research and Exploration in Laboratory*, vol. 42, no. 2, pp. 127–132, 2021.

[13] T. Lindeberg, "Scale invariant feature transform," *Scholarpedia*, vol. 7, no. 5, Article ID 10491, 2012.

[14] Y. Zhu and X. He, "Face recognition algorithm based on SIFT sparse representation," *Journal of Xi'an Polytechnic University*, vol. 34, no. 6, pp. 106–112, 2020.

[15] D. Zhu and Y. Yi, "Research on license plate positioning and parallelization based on SIFT algorithm," *Network Security Technology & Application*, vol. 11, pp. 67–70, 2020.

[16] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 886–893, San Diego, CA, USA, June 2005.

[17] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8, Anchorage, AK, USA, June 2008.

[18] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[19] Y. He, "Face mask detection algorithm based on HSV+HOG features and SVM," *Journal of Measurement Science and Instrumentation*, vol. 1-11, pp. 1674–8042, 2021.

[20] X. Ren and H. Chen, "Pedestrian detection based on HOG and SVM," *Science and technology innovation*, vol. 6, no. 3, pp. 73-74, 2021.

[21] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 10, pp. 580–587, Columbus, OH, USA, June 2014.

[22] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, Santiago, Chile, December 2015.

[23] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: towards real-time object detection with region proposal networks," 2015, https://arxiv.org/abs/1506.01497.

[24] W. Liu, D. Anguelov, D. Erhan et al., "Ssd: single shot multibox detector," in *Proceedings of the Computer Vision - ECCV 2016*, pp. 21–37, Amsterdam, The Netherlands, October 2016.

[25] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, Venice, Italy, October 2017.

[26] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li, "Single-shot refinement neural network for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4203–4212, Salt Lake City, UT, USA, June 2018.

[27] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, Las Vegas, NV, USA, June 2016.

[28] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "Yolov4: optimal speed and accuracy of object detection," 2020, https://arxiv.org/abs/2004.10934.

[29] J. Redmon and A. Farhadi, "Yolov3: an incremental improvement," 2018, https://arxiv.org/abs/1804.02767.

[30] C. Y. Wang, A. Bochkovskiy, and H. Y. M. Liao, "Scaled-YOLOv4: scaling cross stage partial network," 2020, https://arxiv.org/abs/2011.08036.

[31] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, "Bag of tricks for image classification with convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition CVPR*, pp. 558–567, Long Beach, CA, USA, June 2019.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the Conference on Computer Vision and Pattern Recognition CVPR*, vol. 10, pp. 770–778, Las Vegas, NV, USA, June 2016.

[33] J. Kollár, "Flops," *Nagoya Mathematical Journal*, vol. 113, pp. 15–36, 1989.

[34] S. Woo, J. Park, J. Y. Lee, and I. S. Kweon, "Cbam: convolutional block attention module," in *Proceedings of the European Conference on Computer Vision ECCV*, pp. 3–19, Munich, Germany, September 2018.