Proceedings of the 2019 IEEE 16th International Conference on Networking, Sensing and Control, May 9-11, 2019, Banff, Alberta, Canada

Helmet Detection Based On Improved YOLO V3 Deep Model

Fan Wu[1,3], Guoqing Jin [2], Mingyu Gao [1,3], Zhiwei HE [1,3]*, Yuxiang Yang [1,3]
[1]college of Electronic Information,Hangzhou Dianzi University
[2]Hangzhou Xujian Technology Co., Ltd.
[3]Zhejiang Provincial Key Lab of Equipment Electronics, Hangzhou, China
zwhe@hdu.edu.cn

Abstract—**Helmet wearing is very important to the safety of workers at construction sites and factories. How to warn/identify/certify workers "whether or not the helmet is worn" is often a difficult point for enterprises to monitor. Based on the YOLO V3 full-regression deep neural network architecture, this paper utilizes the advantage of Densenet in model parameters and technical cost to replace the backbone of the YOLO V3 network for feature extraction, thus forming the so-called YOLO-Densebackbone convolutional neural network. The test results show that the improved model can effectively deal with situations that the helmet is stained, partially occluded, or there are many targets with a low image resolution. In the test set, compared with the traditional YOLO V3, the improved algorithm detection accuracy increased by 2.44% with the same detection rate. The establishment of this model has important practical significance for improving helmet detection and ensuring safe construction.**

Key word—YOLO V3; Densenet; CNN; Safety helmet detection; machine vision

I. INTRODUCTION

In recent years, intelligent monitoring has become one of the main directions of computer vision in engineering. The traditional methods of image recognition such as Gaussian mixture model [1] can only segment foreground from background. It can be used under the special recognition task, but this method is not suitable for mobile cameras, and it is even more difficult to distinguish which category the foreground belongs to. In the field of pedestrian recognition, researchers use HOG [2] (Histogram of Oriented Gradient), a hand-designed feature extractor to extract contour features. Then, based on the extracted features, a classifier such as SVM [3] (support vector machine) is used to detect pedestrians. Due to the imperfect characteristics and poor generalization ability of the hand-designed feature extractor, it is difficult to apply in practical engineering.

With the continuous development of computer hardware, the application of GPU for large-scale parallel computing has been widely used. The improvement of the computing speed makes it possible to train large deep neural networks. Servers with multiple GPUs provide the necessary guarantees for the application of neural network algorithms. In the field of target detection, the method of using deep learning has become mainstream. Among them are algorithms based on region proposals, such as R-CNN [4] (Regions with CNN features) and its improved versions of Fast R-CNN [5] and Faster R-CNN [6]. There are also regression networks such as SSD [7] (Single Shot Multibox Detector) and YOLO [8] (You Only Look Once) networks. R-CNN was proposed by Ross Girshick in 2014. The implementation of the algorithm is divided into several steps: region proposals extraction, CNN (Convolutional Neural Networks) feature computation and bounding-box regression. However, the region proposals need to be pre-fetched, which will take up a lot of disk space. At the same time, each region proposal needs to be calculated by the CNN network, and a large number of overlapping regions will bring waste of computing resources. Due to these shortcomings, Fast-RCNN proposed a solution for building ROI pooling layers and multi-task loss layers. Faster-RCNN used a method of adding additional RPN branch networks to integrate region proposals extraction into deep networks. These solutions have improved the speed of the R-CNN algorithms, but it is still difficult to meet the engineering requirements in real-time video. The regression-based target detection network is more advantageous in detection speed. The SSD network achieves a good balance of efficiency and effect due to the combination of regression and multi-scale features. After continuous iterative improvement of YOLO, at 320×320 YOLOv3 runs in 22 ms at 28.2 mAP, as accurate as SSD but three times faster [10][16]. In terms of speed and accuracy, it is more appropriate to use YOLO V3 as the target detection network in engineering.

Due to the lack of safety awareness of the construction team, there is no guarantee that every worker will wear a safety helmet as required, the safety of construction sites is difficult to secure. We propose an automatic detection method for safety helmet based on YOLO V3 deep neural network. In our project, we use Densenet [11] (Densely Connected Convolutional Networks) instead of the original backbone Darknet53. The experimental results show that we can get better detection results under the same detection time.

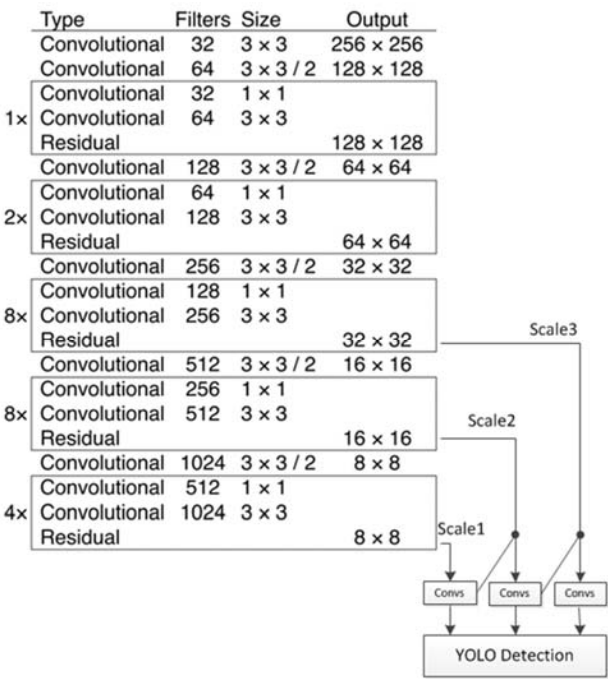
II. ALGORITHM ANALYSIS

A. YOLO V3

YOLO frame detection as a regression problem. We simply run neural network on a new image at test time to predict detections [8]. The algorithm divides the input image into $S \times S$ grids. If the center point of the object's ground truth fall within a certain grid, the grid is responsible for detecting the object. Each grid outputs B prediction bounding boxes, including position information of the bounding box (center point coordinates x, y, width w, height h), and prediction confidence.

YOLO V3 draws on the anchor box [6] idea of Faster R-CNN. YOLO V3 abandon the manually selected anchor box and run k-means clustering on the dimensions of bounding boxes to get good priors [9]. YOLO V3 uses this method to obtain 9 cluster centers, which can better cover the characteristics of the ground truth of the train set. YOLO also adopted a multi-scale prediction method similar to the FPN [12] (Feature Pyramid Networks) network. Because of predictions on multiple scale feature maps, YOLO V3 has acquired image features at different scales and greatly improved the detection of small targets. Combining the anchor box and multi-scale prediction idea, YOLO V3 first assigns several anchor boxes to each scale feature map according to the length and width of the anchor boxes. Calculate the IoU (Intersection-over-Union) of anchor boxes to each ground truth and assign the ground truth to the feature map of the anchor box closest to its shape. When performing bounding box regression training, back propagation will cause the predicted bounding box to approach the ground truth.

YOLO V3 uses Darknet53 network as the backbone, As shown in Figure 1.



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8

Fig. 1. YOLO V3 network structure [10]

Its structure is similar to ResNet [13], both use the residual to make the network deeper. In addition, due to the use of a fully convolutional structure, the detection speed is greatly improved. At each scale, the output is followed by a number of 3×3 and 1×1 convolutional layers to facilitate the calculate loss function.

The loss function of the YOLO V3 consists of coordinate error, IOU error, and classification error. We perform error calculations on S^2 grids.

$$\text{loss} = \sum_{i=0}^{S^2} coordErr + iouErr + clsErr \tag{1}$$

On the assumption that:

$$\text{BC}(a, \hat{a}) = -[aloga + (1-a)\log(1-\hat{a})] \tag{2}$$

$$\text{ST}(w, h) = 2 - w * h \tag{3}$$

The BC function is the cross entropy loss function. The value calculated by the ST function is the scaling factor. The YOLO V3 loss function is as follows:

$$\begin{aligned}\text{coordErr} = &\sum_{i=0}^{S^2}\sum_{j=0}^{B}\{I_{ij}^{obj} * ST(w_i, h_i) \\ &* [BC(x_i, \hat{x}_i) + BC(y_i, \hat{y}_i)]\} \\ &+ \lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\{I_{ij}^{obj} * ST(w_i, h_i) \\ &* [(w_i - \widehat{w}_i)^2 + (h_i - \hat{h}_i)^2]\}\end{aligned} \tag{4}$$

$$\begin{aligned}\text{iouErr} = &\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{obj}[BC(c_i, \hat{c}_i)] \\ &+ \lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{noobj}[BC(c_i, \hat{c}_i)]\end{aligned} \tag{5}$$

$$\text{clsErr} = \sum_{i=0}^{S^2} I_i^{obj} \sum_{c\in classes} BC\big(p_i(c), \hat{p}_i(c)\big) \tag{6}$$

Where $\left(\hat{x}, \hat{y}, \widehat{w}, \hat{h}, \hat{C}, \hat{p}\right)$ are respectively represented as the center coordinates, width, height, confidence, and category probability of the predicted bounding box, and those symbols without the cusp are true labels. B indicates that each grid predicts B bounding boxes, I_{ij}^{obj} indicates that the object falls within the j-th bounding box of the i-th grid. I_{ij}^{noobj} indicates that there are no targets in this bounding box.

CoordErr is the coordinate error, the cross entropy loss is used for the coordinates of the center point, and the variance loss is used for the width and height. Set the $\lambda coord$ to 0.5, indicating that the width and height errors are less effective in the calculation. For the coordinate error, calculations are only made when the gird predicts an object [14].

IouErr is the IOU error. The weight of the grid containing the object and the grid containing no object is different. Therefore, $\lambda noobj = 0.5$ is introduced to weaken the influence of a large number of girds without objects on the loss value.

ClsErr is the classification error. It uses cross-entropy to calculation loss and only works on the grid with a target.

In addition, YOLO V3 uses the sigmoid function as the activation function for class prediction. Compared with the softmax function, sigmoid function effectively solves the problem that the same target has two labels.

B. Densenet network

Densenet connects each layer to every other layer in a feed-forward fashion [11]. It delivers better performance than ResNet with fewer parameters. The network makes better use of features, reducing the impact of vanishing-gradient to some extent. The basic network structure of Densenet is shown in Figure (2).

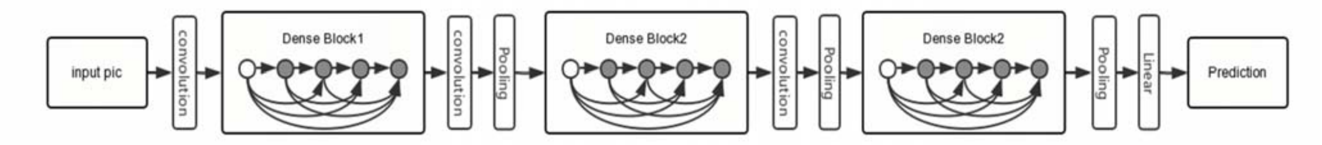


Fig. 2. DenseNet structure

Let each dense block have L layers, then the block has a total of L $(L + 1)$ / 2 connections, that is, the input of each layer comes from the output of all the previous layers. Thanks to this design, we can make the transfer of features and gradients more efficient while reducing the number of feature maps in the convolutional layer. The reduction of parameters also makes training easier.

The core expressions of ResNet and Densenet are as follows, we can see the difference between the two [11].

$$\mathrm{x}_l = \mathrm{H}_l(x_{l-1}) + x_{l-1} \tag{7}$$

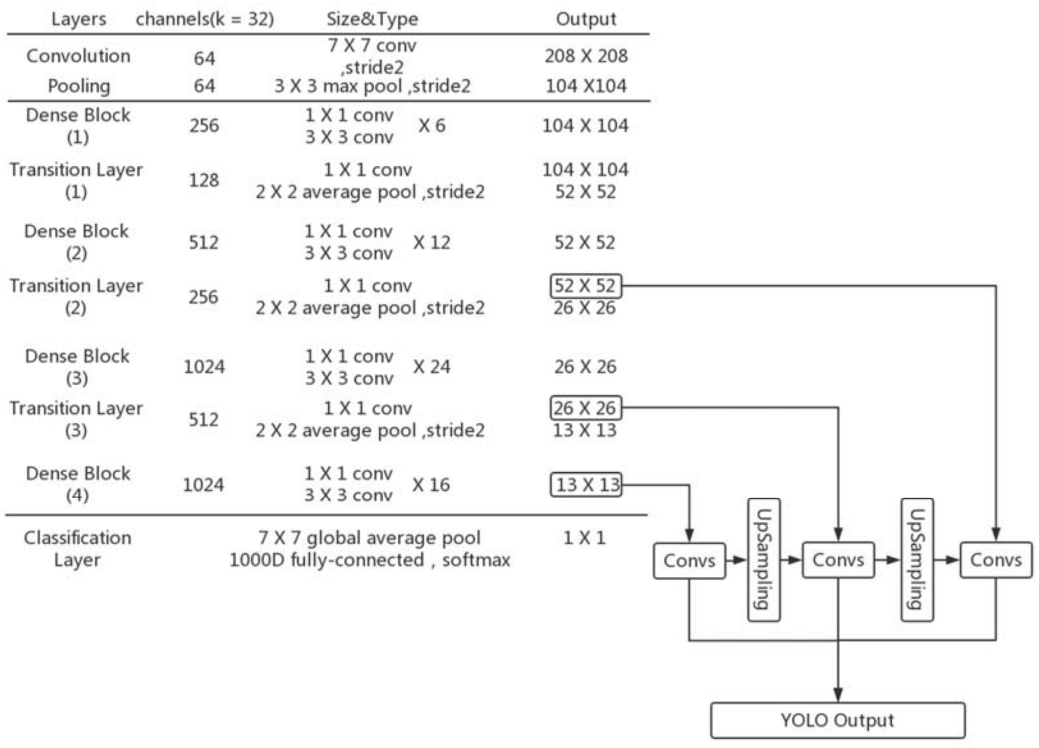
$$\mathrm{x}_l = \mathrm{H}_l(x_0, x_1, \dots, x_{l-1}) \tag{8}$$

l denotes a layer, x_l denotes the output of layer l, and H_l denotes one convolution calculation. For ResNet, the output of layer l is the output of layer l-1 plus the nonlinear transformation of the output of layer l-1. Densenet is an optimization based on this operation. $(x_0, x_1, \dots, x_{l-1})$ denotes that the feature map of the output from layer 0 to layer l-1 is channel-connected in the depth dimension. The output of the layer l is the sum of all the front layer outputs of this dense block.

In applications, We can achieve better results with limited data sets if we use a network that is easier to train and converges faster. These performance improvements make it easier to detect safety helmet in images.

III. Algorithm Improvement

The backbone of YOLO V3 is built on Darknet53, which draws heavily on the structure of ResNet. Due to DenseNet is easier to train and has higher accuracy, we borrow Densnet to build the YOLO-Densebackbone network structure.



Layers	channels(k = 32)	Size&Type	Output
Convolution	64	7 X 7 conv ,stride2	208 X 208
Pooling	64	3 X 3 max pool ,stride2	104 X104
Dense Block (1)	256	1 X 1 conv 3 X 3 conv X 6	104 X 104
Transition Layer (1)	128	1 X 1 conv 2 X 2 average pool ,stride2	104 X 104 52 X 52
Dense Block (2)	512	1 X 1 conv 3 X 3 conv X 12	52 X 52
Transition Layer (2)	256	1 X 1 conv 2 X 2 average pool ,stride2	52 X 52 26 X 26
Dense Block (3)	1024	1 X 1 conv 3 X 3 conv X 24	26 X 26
Transition Layer (3)	512	1 X 1 conv 2 X 2 average pool ,stride2	26 X 26 13 X 13
Dense Block (4)	1024	1 X 1 conv 3 X 3 conv X 16	13 X 13
Classification Layer		7 X 7 global average pool 1000D fully-connected , softmax	1 X 1

Fig. 3. YOLO-Densebackbone architecture. A 3-layer dense block with a growth rate of k = 32

Figure 3 shows the improved YOLO network structure. We use 3 dense blocks to form the backbone and set the growth rate (*k*) to 32. Growth rate (*k*) indicates the number of channels output by a dense block. In such a network, we do not need the growth rate to be too large to get a good result. Because Densenet needs multi-channel integration for feature maps, we need to make each dense block the same size. At the same time drawing on the idea of DenseNet-BC [11], we add a 1×1 convolution kernel before the 3×3 convolution kernel in the dense block, which is called "bottleneck layer". The number of the 1×1 filters is fixed at 4 times the growth rate. This operation reduces the number of feature maps and incorporates the characteristics of each channel. At the same time, the existence of transition layers further reduces the number of channels output by each dense block, greatly reducing the calculative quantity. Here we set the output of the transition layer to half the number of the last dense block channel. We call a layer in the dense block a convolution block which includes batch normalization (BN) layer, rectified linear units (ReLU) layer and convolution layer. In order to avoid over-fitting, we added drop-out operation to each convolution block.

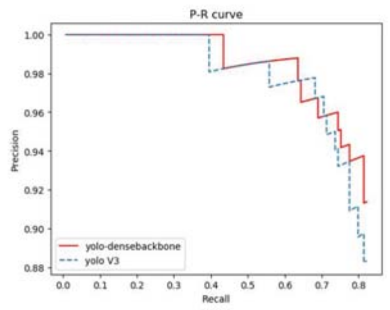
The size of the final three-scale feature map is set at 13×13, 26×26, and 52×52. Large objects are predicted by smaller feature maps, while large feature maps with more details are used to improve the prediction of small objects. The feature map of 13×13 is the output of the backbone network. The feature maps of 26×26 and 52×52 are upsampled by the previous output. The three scales respectively preserve the prediction results.

Since there is no safety helmet in the ImageNet-1000, we do not load the pre-trained model. We chose 9 anchor boxes as YOLO V3 did, and re-executed K-means clustering through our own train set. On our own data set the 9 clusters were: (17×21), (28×33), (42×51), (58×69), (83×93), (106×124), (152×177), (212×244), (326×393).

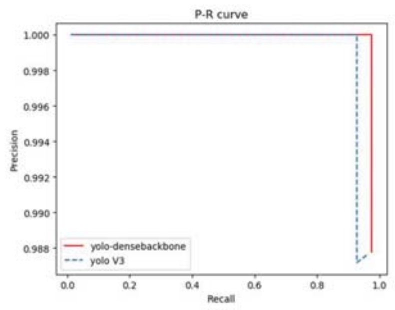
IV. Experimental Result

The data set used in this experiment is composed of pictures taken by the Internet and a small number of pictures taken by ourselves. The images are divided into train set, validation set, and test set. The number of pictures in the train set is 402. The ratio of the three data set is approximately 8:1:1. We also analyzed the video I shot. Simulated under a computer with an NVidia 1080 Ti graphics card.

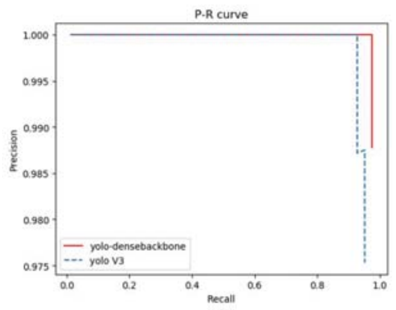
Due to the small number of images and the uneven distribution of the training set, we made corresponding data augmentation. All images are manually labeled. We first convert the training image to 416×416. We use the regional interpolation with constant aspect ratio to convert the image resolution and fill the pixel value (128,128,128) in the missing part. Random horizontal translation, image flipping, and image distorting are used for data augmentation. At the same time, the same transformation is performed on the bounding box. In order to make the data in the same distribution, we manually designed the train set and validation set to make the data distribution more similar.



(a) P-R curve of yolov3-epoch769 and yolo-densebackbone-epoch 721 in validation set



(b) P-R curve of yolov3-epoch769 and yolo-densebackbone-epoch 721 in test set



(c) P-R curve of yolov3-epoch1212 and yolo-densebackbone-epoch 721 in test set

Fig. 4. Precision-Recall curve

We used the same way to train the YOLO V3 network and YOLO-Densebackbone network respectively for 1500 epochs. Val loss is used to monitor the training process. If the validation set loss of this iteration is better than the previous one, the weight file will be saved.

Under the same hardware conditions, we completed training on two networks almost simultaneously. Due to the small number of training samples, we ended the training after training for 1500 epochs, and the loss value was hardly reduced.

Figure 4(a) and Figure 4(b) are the Precision-Recall curves of the two networks in the validation and test sets and the number of iterations is 800. Figure 4(c) is the P-R curves of the two networks for 1500 epochs. It can be seen that the recall and precision of YOLO-Densebackbone are greater than YOLO V3 most of the time. We also found that YOLO-Densebackbone found the best weight in the 721 epochs, and the loss value decreased faster than YOLO V3. However, YOLO V3 found the best weight in the 1212 epochs.

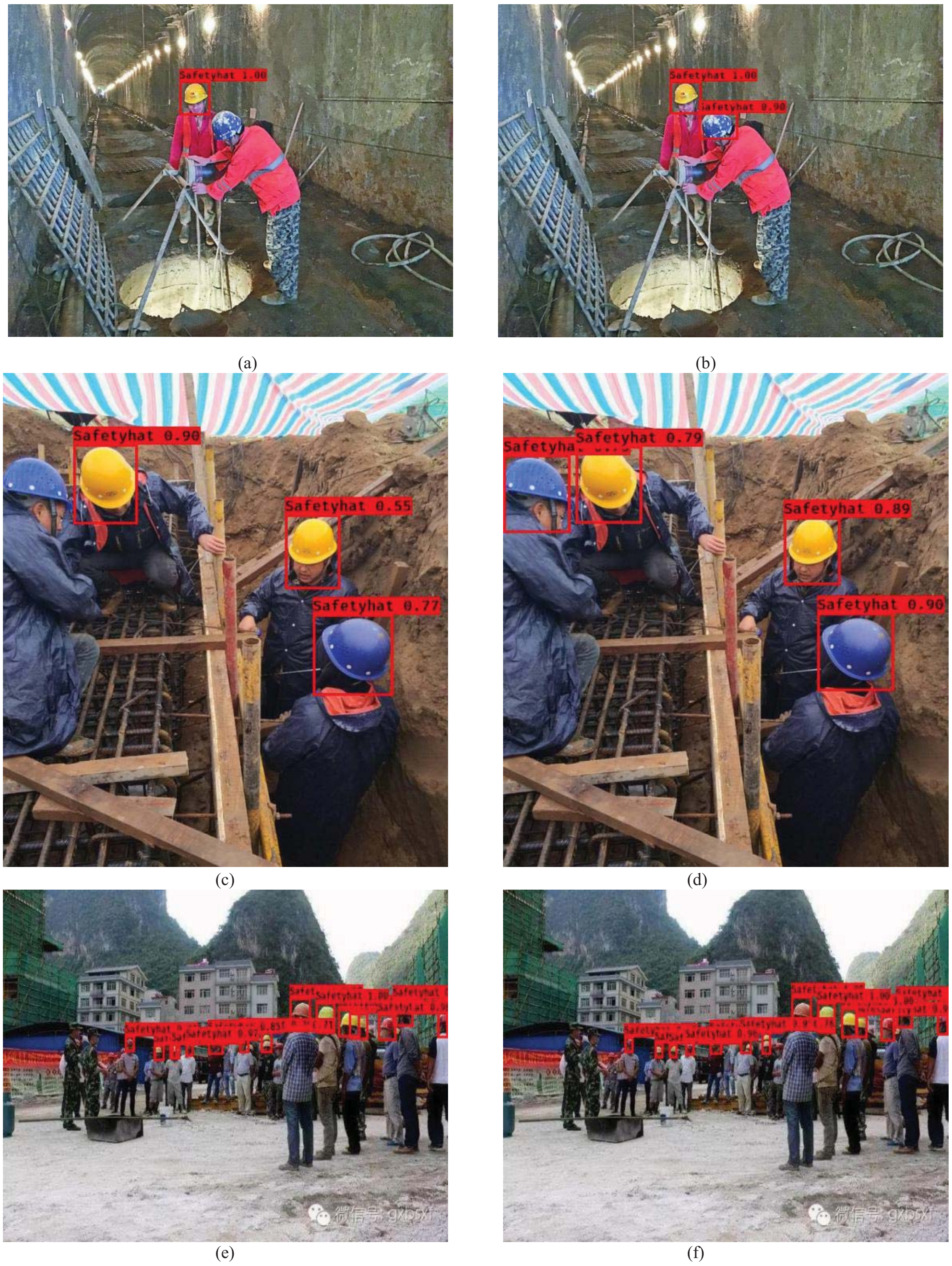


Fig. 5. (a), (c), (e) are the result detected by YOLO, (b), (d), (f) are the result detected by YOLO-Densebackbone

The test results of the two networks are shown in Figure 5. It can be seen that the resolution of the first set of pictures is poor and there are more paint spots on the safety helmet. Figure 5(b) shows that YOLO-Densebackbone successfully detected two stained safety helmets. The second set of pictures shows that YOLO-Densebackbone can also detect occluded targets well. The leftmost safety helmet is successfully detected by YOLO-Densebackbone network, but the YOLO V3 output does not have that bounding box. The third set of pictures has more safety helmets, and neither algorithm can completely detect all the safety helmets (even people can hardly fully recognize). However, YOLO-Densebackbone did not detect more overlapping bounding box and negative target like YOLO V3. The improved algorithm reduces false positives.

The mAP for the two algorithms is calculated as shown in the following table. For the test set, our network's mAP increased by 2.44%, and both recall and precision were slightly improved.

TABLE 1. Comparison between YOLO network and YOLO-Densebackbone network

	mAP	Recall	Precision
YOLO V3	95.15%	97.53%	95.18%
YOLO-Dense backbone	97.59%	97.59%	98.78%

In the video test, both networks run at 21 fps. It can be said that the detection speed was not affected when the backbone was modified to make the network deeper.

V. Conclusion

This paper refers to the Densenet network and modifies the backbone based on YOLO V3. The problem of inaccurate detection and overlapping bounding boxes in the original network is alleviated. We extract features through the new network structure and connect the YOLO V3 network's original output methods for prediction. The experimental results show that the optimized target detection network improves the effectiveness of detecting safety helmets and can provide better reference results for security monitoring equipment. In the next work, we will collect a large number of positive and negative samples of real scenes to improve the generalization ability of the network. It will also try to further expand the data set through the GAN network [15] to achieve better training results. We can form a better safety helmet detection system by using neural network algorithms and front-end video capture modules.

Acknowledgment

This work was supported in part by the Nation Natural Science Foundation of China (61873077 and U1609216). This work was supported by Zhejiang Provincial Key Lab of Equipment Electronics.

References

[1] Chen Z , Ellis T . Self-adaptive Gaussian mixture model for urban traffic monitoring system[C]// IEEE International Conference on Computer Vision Workshops. IEEE, 2011:1771-1772.

[2] Dalal N , Triggs B . Histograms of Oriented Gradients for Human Detection[C]// 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005, IEEE, 2005:886-893.

[3] Burges C J C . A Tutorial on Support Vector Machines for Pattern Recognition[J]. Data Mining and Knowledge Discovery, 1998, 2(2):121-167.

[4] Girshick R , Donahue J , Darrell T , et al. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation[C]// IEEE Conference on Computer Vision & Pattern Recognition. IEEE Computer Society, 2014:580-587.

[5] Girshick R. Fast R-CNN [C]// Proc of IEEE International Conference on Computer Vision. 2015:1440-1448.

[6] Ren S, He K, Girshick R, et al. Faster R-CNN: towards real-time object detection with region proposal networks [C]// Proceedings of the 2015 advances in Neural Information Processing Systems. Palais des Congrès de Montréal, Montréal CANADA. 2015:91-99.

[7] Liu W, Anguelov D, Erhan D, et al. SSD: single shot multibox detector [C]// Proc of European Conference on Computer Vision. Springer, 2016:21-37.

[8] Redmon J, Divvala S, Girshick R, et al. You only look once: unified, real time object detection [C]// Computer Vision and Pattern Recognition. 2016:779-786.

[9] Redmon J,Farhadi A.YOLO9000:Better,Faster, Stronger [C]//IEEE Conference on Computer Vision and Pattern Recognition. 2017:6517-6525

[10] Redmon J,Farhadi A. YOLOv3: An Incremental Improvement [C]//IEEE Conference on Computer Vision and Pattern Recognition. 2018.

[11] Huang G,Liu Z,Weinberger K Q,et al. Densely connected convolutional networks[C]//IEEE Conference on Computer Vision and Pattern Recognition. 2016:4700-4708.

[12] Lin T Y , Dollar P , Girshick R , et al. Feature Pyramid Networks for Object Detection[C]// 2017 IEEE Conference on Computer Vision and Pattern Recognition （CVPR）. IEEE Computer Society, 2017:2117-2124.

[13] He K , Zhang X , Ren S , et al. Deep Residual Learning for Image Recognition[C]// 2016 IEEE Conference on Computer Vision and Pattern Recognition(CVPR). IEEE Computer Society,2016:770-778.

[14] Ren S , He K , Girshick R , et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2015, 39(6):1137-1149.

[15] Goodfellow I J, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets[C]// International Conference on Neural Information Processing Systems. MIT Press, 2014:2672-2680.

[16] Lin T Y , Goyal P , Girshick R , et al. Focal Loss for Dense Object Detection[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2017, PP(99):2999-3007.