2021 IEEE 5th Information Technology,Networking,Electronic and Automation Control Conference (ITNEC) | 978-1-6654-1599-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/ITNEC52019.2021.9586883

PFG-YOLO: A Safety Helmet Detection based on YOLOv4

Huixuan Wang[1], Huayong Ge[1], Muxian Li[1]
1. School of information science and technology, Donghua University, Shanghai,China
w18375357956@163.com,gehuayong@dhu.edu.cn, 974073492@qq.com
Corresponding Author: Huayong Ge Email: gehuayong@dhu.edu.cn

***Abstract*—Developing effective helmet detection is very useful in construction sites. To explore the applicability of YOLOv4 in helmet testing, a new helmet detection system PFG-YOLO based on the YOLv4 network is proposed. The up-sampling using interpolation method is replaced by PixelShuffle which solves the losing information on fusing low-resolution with high-resolution feature maps. The ghost module is introduced to reduce the FLOPs and parameters of the network. The fire module is introduced to reduce the model size. When comparing the performance of the two models, the PFG-YOLO we proposed, not only increases the accuracy but also decreases the complexity of the network model. The mAP of class helmet and class no-helmet increased about 5%, both parameters and FLOPs at the corresponding layers were reduced by about a half, and the model size reduced about 20%.**

Keywords—YOLOv4, PixelShuffle ,SqueezeNet , GhostNet, Helmet detection

I. INTRODUCTION

A helmet is extremely important for building worker on construction sites because head injuries are often fatal, the helmet can protect construction workers to a certain extent[1]. To ensure workers wear safety helmets, construction companies usually manually check the wearing of helmets by video monitoring or other way, which cannot make sure a timely warning to workers who have no helmet on their head. Object detection can solve this problem by accurately detecting the helmet in real-time. However, detecting small targets like hard hats is also a challenging computer vision problem, which attracts interest from both academia and industry.

Object detection can be divided into two categories, they are two-stage methods and one-stage methods respectively and most of them are anchors based[2]. The two-stage methods treat object detection as a classification problem. First, the region containing the object is generated, and then the candidate region is classified and calibrated to get the final detection result. The one-stage detection algorithm directly gives the final detection result without an explicit step of generating a candidate box. A two-stage algorithm like Fast R-CNN[3] and Faster R-CNN[4] both have good performance in precision, but the speed is not good. In response to this question, YOLO[5] was proposed. YOLO eliminates the operation of generating candidate boxes and divides the image into grids, then predicts object borders and categories directly in each grid. After YOLO, YOLOv9000[6] YOLOv3[7], and YOLOv4[8] were proposed, these algorithms make the YOLO series even better.

Recently, anchor free methods like CornerNet[9] and CenterNet[10] draw the attention of researchers. These algorithms have a great performance not only in precision but also in speed. In order to improve the performance of various algorithms, many useful "tricks" are proposed. Wenzhe Shi et al. proposed PixelShuffle[11], this method introduced an efficient sub-pixel convolution layer that can learn a set of filters instead of an up-sampling filter to get a high-resolution map from a low-resolution feature map. Han et al. proposed SqueezeNet[12] which is a lightweight CNN model and increases the detection speed. Kai Han et al. proposed GhostNet[13], they used a series of linear transformations to generate many ghost feature maps that can replace the intrinsic feature maps to reveal information, which is called cheap operation.

In this paper, a better helmet detection based on YOLOv4 is presented. Firstly, the substitution PixelShuffle for up-sampling to get high-quality feature maps. Secondly, the fire module is introduced to reduce the model size. Finally, the ghost module is applied to replace some normal convolution to reduce parameters and FLOPs. Some related works are reviewed in the next section, section Ⅲ introduces our network structure and loss function, experiments and results are arranged in section Ⅳ, the final is the conclusion of our works.

II. RELATED WORK

A. The Introduction of YOLOv4

Fig. 1 shows the network structure of YOLOv4. Firstly, CSPdarknet53 is the backbone for extracting feature maps. Secondly, SPP (Spatial Pyramid Pooling)[14] is an additional module, PANet[15] is a path-aggregation neck. Finally, YOLO head is the same as YOLOv3.

The prediction steps of YOLOv4 can be divided into three steps. The first step is feature extraction. The input image is reshaped to $416 \times 416 \times 3$, then the reshaped image is sent to CSPdarknet53 which is made up of five

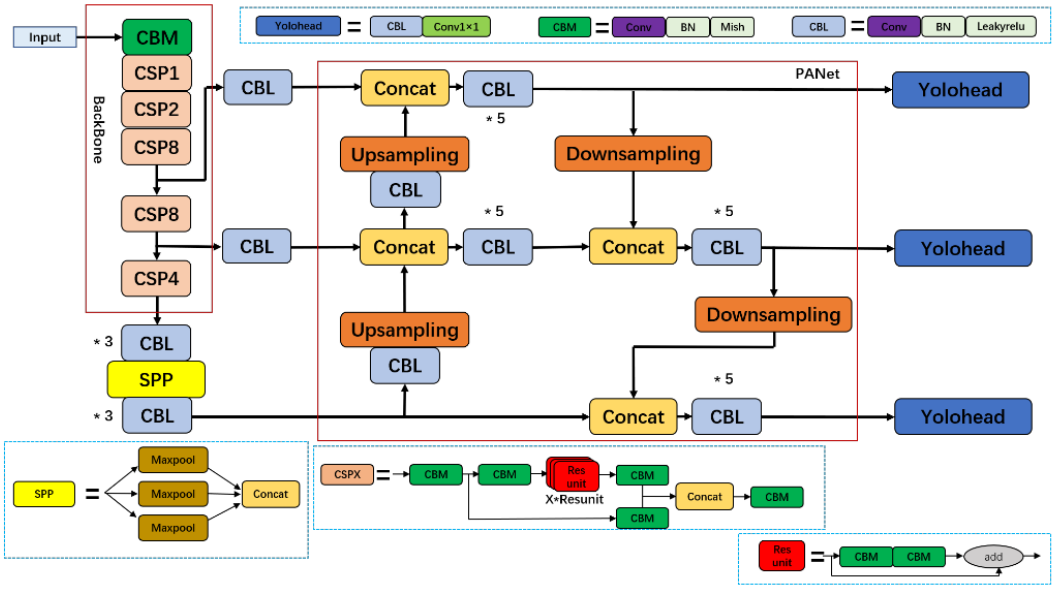


Fig. 1. The Network of YOLOv4

Resblock-body. After processing by CSPdarknet53, three scales of feature maps are extracted. The second step is dealing with feature maps, SPP as an additional module which is used for increasing receptive field just like ASPP[16] and RFB[17] do, then the dealt deep layer feature maps are sent to PANet with other two lower layer feature maps. In PANet, different scales of feature maps are concatenated by up-sampling and down-sampling. In PANet feature maps are extracted twice, after down-to-up feature maps extraction and concatenation, up-to-down feature maps extraction and concatenation are followed. The final step is prediction.

B. *ESPCN*

ESPCN (efficient sub-pixel convolutional neural network) consists of two parts, one is hidden layers, another is sub-pixel convolution layer which is also called PixelShuffle. ESPCN is a single image super-resolution(SISR) technique[18]. There are many ways you can upsample the feature map, like using a single filter, bicubic interpolation, or nearest interpolation. Other ways like Transposed Convolution and un-pooling, but these ways can't restore information like the SISR technique. Rather than simply double the size of the image, the purpose of this way is to recover missing high-resolution information on low-resolution images.

C. *SqueezeNet*

There are three strategies the SqueezeNet adopts: the first is to replace part of the filter of 3x3 with the filter of 1x1, the second is to reduce the input channels, the third is down-sampling is carried out at the late stage of the whole network so that the convolution layer has relatively large activation maps. A new module named "fire module" in this network is composed of two layers: "squeeze layer" and "expand layer". 1×1 convolution filters form the "squeeze layer", the "expand layer" is divided into two parts, one consists of 1×1 convolution filters and another is 3×3 convolution filters. The two parts are concatenated at the end of the fire module.

D. *GhostNet*

The core part of GhostNet[13] is the Ghost module. The Ghost module divides the convolutional layer into two parts. The first part of it is to get intrinsic feature maps by convolution layer with half of the convolution kernels, then to get the Ghost feature maps through cheap operation on intrinsic feature maps. The cheap operations in the second part are linear operations like 3×3 or 5×5 convolution. The number of feature maps gets by taking ghost module is the same as a convolutional layer but only half of parameters and FLOPs. GhostNet[13] is also a lightweight network.

III. THE PFG-YOLO DETECTION MODEL

Based on YOLOv4 structure, a new model PFG-YOLO which has better performance is proposed. The improvements base on YOLOv4 are local at the PANet, the network structure is showing in Fig. 2.

The PANet in YOLOv4 is an important improvement of YOLOv4 compared to YOLOv3-SPP which is another version of YOLOv3. As Fig.2 shows there are two steps in this part: down-to- up feature maps extraction and concatenation, up-to-down feature maps extraction and concatenation. There are 3 kinds of module at these two steps, they are convolution upsampling and downsampling. And the function of upsampling and downsampling is to change the scale of higher layer feature maps and lower feature maps for the purpose to integrate the information of these two different layers. Firstly, look at how YOLOv4

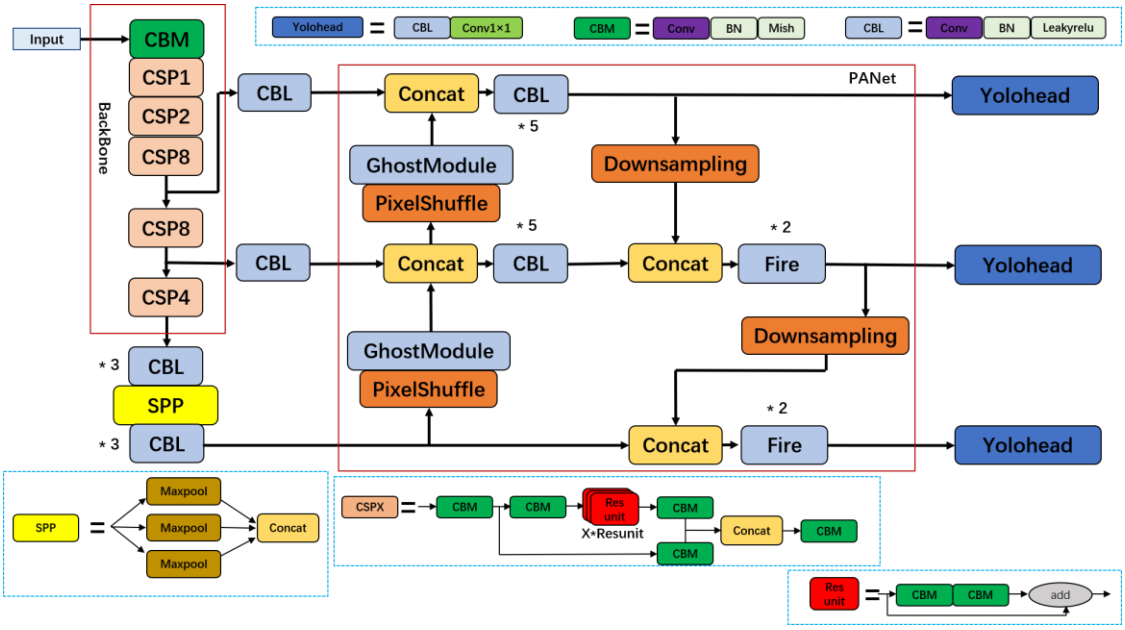


Fig. 2. The Network of PFG-YOLO

implements this function. As Fig. 3 shows, the first convolutional layer reduce half of the channels, the second up-sampling layer which is using the nearest interpolation double the scale of input feature maps. For better accuracy, the following series of operations are taken on YOLOv4.

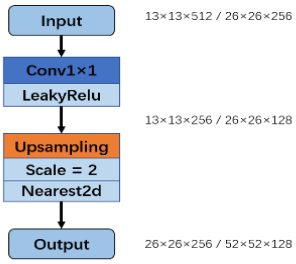
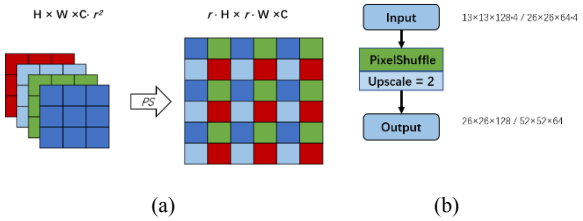


Fig. 3. The up-sampling of YOLOv4

shuffling operator that can turn $\mathbf{I}^{LR}$ (H ×W ×C·r^2) to $\mathbf{I}^{HR}$ (r H ×r W × C) which can be expressed by equation (2), we choose the upscale factor *"r"* here is 2, where x, y are the coordinates of high-resolution at H and W dimension, as Fig. 4. (a) shows one color represents a C channel, the output is resized by these four different colors. The size of tensor changes as Fig. 4. (b) shows.



(a) (b)

Fig. 4. The PixelShuffle.

A. *The substitution PixelShuffle for upsampling:*

Firstly, PixelShuffle is introduced to replace the up-sampling module. In order to double the scale of lower feature maps, r set here is 2 which is the upscale factor. After a layer of PixelShuffle, the channels change to 1/4 of input channels, next, a convolutional layer is taken which changes the channels to 1/2 of input channels. This substitution can alleviate the loss of information during the up-sampling of lower-resolution feature maps to higher-resolution. Fig. 4 shows the process of these operations.

$$\mathbf{I}^{HR} = \mathcal{PS}(\mathbf{I}^{LR}) \tag{1}$$

$$\mathcal{PS}(T)_{x,y,c} = T_{\lfloor x/r \rfloor, \lfloor y/r \rfloor, C \cdot r \cdot \mathrm{mod}(y,r) + C \cdot \mathrm{mod}(x,r) + c} \tag{2}$$

Mathematically, equation (1) describe this operation: $\mathbf{I}^{LR}$ and $\mathbf{I}^{HR}$ here represent the low-resolution image and high-resolution image respectively. $\mathcal{PS}$ is a periodic

B. *The introduction of Ghost module:*

For the sake of better performance, the Ghost module is introduced to replace the convolution layer. The number of convolution kernels is half of the convolution layers through the primary convolution in the Ghost module. After primary convolution, only can get a half of the channels that the convolution layer gets. The following cheap operation gets another half ghost feature maps. Both these two kinds of feature maps concatenate at the end of the Ghost module. Fig. 5 shows the tensor changes of using the Ghost module replacing the convolution layer.

$$FLOPs = n \times h' \times w' \times c \times k \times k \tag{3}$$

According to equation (3), h' and w' are the height and width of output. $k \times k$ is the kernel size of convolution filters. By utilizing the Ghost module, almost a half of the FLOPs is reduced here compare to the convolution layer

needs, because the primary convolution only needs only half the convolution filters of the convolution layer.

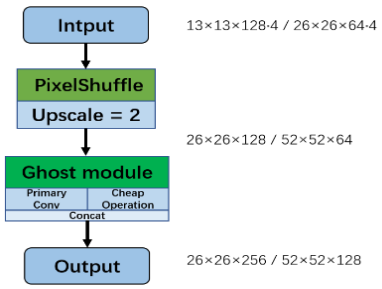


Fig. 5. The PixelShuffle and Ghost module.

C. *The substitution double-fire for 5×Conv:*

The 5×Conv in PANet is to reduce and increase dimension, 5×Conv consists of three 1×1 convolutional layers and two 3×3 convolutional layers. As Fig.6 shows, a fire module consists of two 1×1 convolutional layers and a 3×3 convolutional layer. For the purpose of reducing the parameters of the model, the two double-fire is taken here. Firstly, PFG-YOLO replaced the four 5×Conv with four double-fire but the performance of it is not good, because the model parameters reduced about 23% and the accuracy also dropped. Considering maintaining accuracy, PFG-YOLO choose the two 5×Conv at the up-to-down step in the PANet, which is the tail part of the network. The replacement reduced about 20% of model parameters without declining the accuracy of the model. The process of the fire module is shown in Fig. 6, o represents the number of output channels, x1 is channels of convolution 1×1, and x3 is the channels of convolution 3×3.

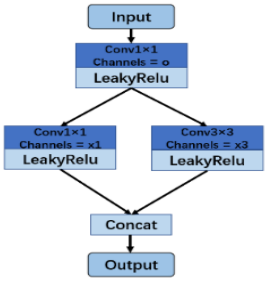


Fig. 6. The Fire module.

Equation (4) can calculate the parameters that 5× Conv needs, $k \times k$ means the 1×1 or 3×3 convolution. Equation (5) is the function to give the parameters of fire module, the c_{in} and c_{out} are input channels and output channels respectively, o and x_1, x_3 are also out channels here. The o set here is 1/8 c_{in} and x_1, x_3 here both are $1/2c_{\text{out}}$.

$$k \times k \times c_{in} \times c_{\text{out}} \tag{4}$$

$$1 \times 1 \times c_{\text{in}} \times \text{o} + 1 \times 1 \times o \times x_1 + 3 \times 3 \times o \times x_3 \tag{5}$$

IV. EXPERIMENTAL RESULTS AND ANALYSIS

A. *DataSet*

Thank njvisionpower for providing Safety-Helmet-Wearing-Dataset. After dealing with the dataset, 3220 images left in the dataset, there are 8949 people wearing helmets whose class name is helmet, and 1488 people without helmets on their heads whose class name is no-helmet. The ratio of the training set to the test set is 8: 2.

B. *Experiment Settings*

The experiment settings of YOLOv4 and PFG-YOLO are both in the same condition. An 11GB 2080Ti card is used to train the YOLOv4 network and PFG-YOLO network model. The framework we choose here is Pytorch. In the training set, the proportion of verification is 20%. The pretrained YOLOv4 model with VOC2012 dataset is used to learn on our helmet dataset for the purpose of reducing training time. The size of the input size is 416×416×3. PFG-YOLO also choose Adam optimizer for optimization and the weight decay take here is 0.0005. The number of anchor boxes is 9, after executing K-means clustering on our dataset, the anchor size changes to (4,7), (8,13), (13,24), (20,35), (28,51), (41,68), (57,105), (89,147), (154,235).

C. *Results*

As Table 1 shows, compared with YOLOv4, the mAP of PFG-YOLO increased 4.8%, the improvements of AP helmet and no-helmet are 1.99%, 7.6% respectively.

TABLE I. DETECTION RESULTS

Algorithm	AP(helmet)	AP(no_helmet)	mAP
YOLOv4	93.39%	63.48%	78.43%
PFG-YOLO	95.38%	71.08%	83.23%

When detecting the helmet through the YOLOv4 model, the detection result can easily be disturbed by some external conditions, these external conditions may be headphones, lamb, or other things which have the same color or shape as a helmet. To reduce the negative effect of these external conditions, PFG-YOLO is proposed. Some test results are shown in Fig. 7, (a) and (c) are the results of PFG-YOLO, (b) and (d) are the results of YOLOv4. The red box means the objects which have no ground truth box but is detected by the model, the green box and blue are ground truth and detection result respectively. According to Fig. 7, PFG-YOLO has better robustness compared to YOLOv4.

(a) (b)

(c) (d)

Fig. 7. Detection result of PFG-YOLO and YOLOv4.

D. Ablation Experiments

To display the more detail of PFG-YOLO improvements based on YOLOv4, the ablation experiments is carried out. Module added in the order PixelShuffle, fire module and ghost module. The changes of model size and AP of helmet are also showing in the Table 2.

TABLE II. ABLATION EXPERIMENTAL RESULTS

Algorithm	**YOLOv4**	**-**	**-**	**PFG-YOLO**
PixelShuffle		√	√	√
Fire module			√	√
Ghost module				√
Model size (MB)	244.3	243.8	195.0	194.9
AP(%)helmet	93.39	94.99	94.78	95.38

As Table 2 shows, before the introduction of the Fire module, the AP helmet increased 1.6%, but the model size also went up to 243.8 MB from 244.3 MB. After this replacement, the model size decreased by about 20%. The final step is to use the Ghost module to replace the convolution layers after two PixelShuffle. The FLOPs can reduce almost half through this replacement, the change of FLOPs and parameters at that layer can be found on the Table 3. On GTX 2080Ti, the FPS of detecting one same image by YOLOv4 and PFG-YOLO are 3.346 and 3.390 respectively.

TABLE III. FLOPS AND PARAMETERS RESULTS

Conv to Ghost	**Paras**	**FLOPs**
Before change	41728	45859840
After change	22956	25958400

According to the statistics above, we can conclude that PFG-YOLO has better performance than YOLOv4, not only in accuracy but also in model size. A different up-sample way: PixelShuffle which fuses the higher and lower feature maps without losing information like using interpolation. two kinds of modules introduction for reducing parameters and FLOPs. The cooperation among these three kinds of modules made PFG-YOLO get better performance.

V. CONCLUSION

In this paper, a new helmet detection base PFG-YOLO which is based on the YOLOv4 network structure is proposed. The accuracy and model size of it are both improved compared to YOLOv4. We firstly introduce the PixelShuffle to replace the up-sampling using interpolation method, which increases the accuracy of network model through the way reducing the alleviate losing information of concatenating the lower resolution feature maps with higher ones. The introductions of ghost module and double fire not only reduce the parameters of the model but also reduce the FLOPs. But there are still some defects in our network: the no-helmet object cannot be detected, some small objects also cannot be detected. Even though the speed of our network is better than YOLOv4, the improvement of speed is not enough. The solutions to these problems are the purposes of our further study next.

REFERENCES

[1] Y. Li, H. Wei, Z. Han, J. Huang, and W. Wang, "Deep Learning-Based Safety Helmet Detection in Engineering Management Based on Convolutional Neural Networks," Advances in Civil Engineering, vol. 2020, no. 6, pp. 1-10, 2020.

[2] X. Wu, D. Sahoo, and S. C. H. Hoi, "Recent advances in deep learning for object detection," Neurocomputing, vol. 396, pp. 39-64, 2020, doi: 10.1016/j.neucom.2020.01.085.

[3] R. Girshick, "Fast R-CNN," Computer Science, 2015.

[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," IEEE Transactions on Pattern Analysis & Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 2017.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," IEEE, 2016.

[6] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," IEEE, pp. 6517-6525, 2017.

[7] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv e-prints, 2018.

[8] A. Bochkovskiy, C. Y. Wang, and H. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," 2020.

[9] H. Law and J. Deng, "CornerNet: Detecting Objects as Paired Keypoints," International Journal of Computer Vision, vol. 128, no. 3, pp. 642-656, 2020.

[10] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "CenterNet: Keypoint Triplets for Object Detection," 2019.

[11] W. Shi, J. Caballero, F. Huszár, J. Totz, and Z. Wang, "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network," IEEE, 2016.

[12] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," 2016.

[13] K. Han, Y. Wang, Q. Tian, J. Guo, and C. Xu, "GhostNet: More Features From Cheap Operations," in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," IEEE Transactions on Pattern Analysis & Machine Intelligence, vol. 37, no. 9, pp. 1904-16, 2014.

[15] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path Aggregation Network for Instance Segmentation," in IEEE, 2018.

[16] L. C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking Atrous Convolution for Semantic Image Segmentation," 2017.

[17] S. Liu, H. Di, and Y. Wang, "Receptive Field Block Net for Accurate and Fast Object Detection," 2017.

[18] C. Y. Yang, C. Ma, and M. H. Yang, "Single-Image Super-Resolution: A Benchmark," in European Conference on Computer Vision, 2014.