information

Article

Object Detection Based on YOLOv5 and GhostNet for Orchard Pests

Yitao Zhang [1,2], Weiming Cai [2,3,*], Shengli Fan [2,3], Ruiyin Song [3] and Jing Jin [2,3]

[1] School of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China
[2] Signal Intelligence Detection and Life Behavior Perception Institute, NingboTech University, Ningbo 315100, China
[3] Zhejiang Engineering Research Center for Intelligent Marine Ranch Equipment, Ningbo 315100, China
* Correspondence: caiwm@nit.zju.edu.cn

Abstract: Real-time detection and identification of orchard pests is related to the economy of the orchard industry. Using lab picture collections and pictures from web crawling, a dataset of common pests in orchards has been created. It contains 24,748 color images and covers seven types of orchard pests. Based on this dataset, this paper combines YOLOv5 and GhostNet and explains the benefits of this method using feature maps, heatmaps and loss curve. The results show that the mAP of the proposed method increases by 1.5% compared to the original YOLOv5, with 2× or 3× fewer parameters, less GFLOPs and the same or less detection time. Considering the fewer parameters of the Ghost convolution, our new method can reach a higher mAP with the same epochs. Smaller neural networks are more feasible to deploy on FPGAs and other embedding devices which have limited memory. This research provides a method to deploy the algorithm on embedding devices.

Keywords: orchard pests; GhostNet; YOLOv5; embedding devices

Citation: Zhang, Y.; Cai, W.; Fan, S.; Song, R.; Jin, J. Object Detection Based on YOLOv5 and GhostNet for Orchard Pests. *Information* **2022**, *13*, 548. https://doi.org/10.3390/info13110548

Academic Editor: Gianluca Valentino

Received: 20 July 2022
Accepted: 16 November 2022
Published: 20 November 2022

1. Introduction

During the growth of fruit trees, it causes huge economic losses if the pests cannot be well prevented. Therefore, it is necessary to identify the orchard pests in a timely and accurate manner, and take corresponding disinfecting measures according to different types of pests [1]. Traditional recognition methods mainly rely on human experience to perform repetitive mechanical human eye recognition. However, this method mainly depends on human experience, which is too subjective, and the effect of recognition by different people may be different [2]. Second, the planting area of orchards is quite wide. Not only a lot of labor cost is required, but it is also inefficient if only human eyes are used for recognition. In view of the problems of poor objectivity and low efficiency in the above methods, a new orchard pest identification technology is urgently needed. Computer vision based on deep learning is an important topic recently, it has a wide range of applications and is used in face recognition and autonomous driving [3]. Ten years ago, due to the limitation of GPU computing power, its development was stagnant. However, in recent years, with the rapid increase of GPU computing power, this limitation has been gradually broken. More and more scholars have begun to use computer vision to solve problems such as image recognition and detection in the field of agriculture [4]. Cheng et al. used a deep convolutional neural network to accomplish the recognition and classification of stored grain pest images and achieved an accuracy of 97.61% [5]. Ding and Taylor used convolutional neural networks to detect and count pests on the pest images from the sticky plate of sexual attractants [6]. In 2018, Shen et al. used the Faster-RCNN algorithm to detect stored grain pests with a magazine background, and the mAp of the method reached 88% [7]. In 2019, Li et al. integrated image enhancements of different scales into the detection and recognition model, thereby solving the problem to which the

traditional single image scale algorithm is not applicable, i.e., the detection of small target pests [8]. In 2020, some researchers used improved yolov3 and Faster RCNN to detect rice planthoppers [9]. Wang et al. applied context information to convolutional neural networks; using the context of pests as prior information, and fusing image visual features and prior-context information to improve the accuracy of pest detection and identification in complex environments [10]. Xie et al. proposed a Faster DR-IACNN model based on the self-built grape leaf disease dataset and Faster R-CNN detection algorithm. The Inception-v1 module, Inception-ResNet-v2 module and SE module were introduced and worked well. The proposed model achieved an outstanding ability for feature extraction, as the mAP was 81.1% [11]. In 2021, James and other researchers used SSD-MobileNet to detect rice pests, and the trained model achieved 78.4% accuracy compared to the 74% accuracy in previous research [12]. In 2022, Wang et al. designed a lightweight attention module to focus on the informative features, and the proposed network outperformed previous studies for pest localization [13]. In 2022, Pang et al. proposed a real-time object detection model for orchard pests based on the improved YOLOv4 algorithm, and the improved model achieved 88% mAP [14].

The YOLO detection network has been attracting attention since it was proposed [15–18], mainly because the network is fast and accurate. It has now been iterated to the fifth version. Our research is based on YOLOv5. In this paper, we propose a novel network that drastically reduces GFLOPs and the weight of the network with almost no loss of accuracy, and explain the feasibility of this approach from the perspective of feature map visualization. The experimental results show that compared with the original YOLOv5, our method can converge faster and maintain a higher mAP.

2. Materials and Methods

2.1. Dataset Construction

The success of computer vision depends on a large amount of data, especially when the neural network parameters are large; it could cause underfitting if the dataset is too small. The more sufficient the dataset is, the more effective features the model can extract, and the better the model fitting effect. Most of the current agricultural pest datasets are collected from laboratory specimens, and the number of pictures is small, which can't meet the conditions required for large-scale network training. Based on the above reasons, we constructed the original orchard pest dataset by means of a web crawler and laboratory specimen image collection. However, there are still many problems with the pictures from the web crawling: ① many pictures are dirty; ② the size of the pictures are not uniform; ③ the pictures have a high degree of overlap. On the basis of the original dataset, we have performed a series of data enhancements, mainly including noise, blurring, rotation, cropping, and flipping. We have enhanced the images that are from the web scraping section. The effects of various data enhancements are shown in Figure 1. Under the conditions of laboratory collection, we simulated the real scene as much as possible to ensure the quality of the picture [19].

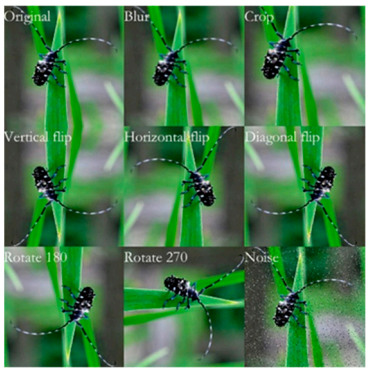


Figure 1. Image augmentation.

The image from web scraping before and after data augmentation are shown in Table 1, the image from laboratory collection is shown in Table 2 and the final dataset is shown in Table 3.

Table 1. Images from web crawler.

Type	Before Data Augmentation	After Data Augmentation
Cicadidae	433	3890
Gryllotalpa spps	403	3620
Scarabaeoidea	288	2592
Locusta migratoria manilensis	354	3186
Cerambycidae	343	3083
Buprestidae	410	3690
Hyphantria cunea	269	2420
sum	2500	22,481

Table 2. Images from laboratory specimen.

Type	Number
Cicadidae	346
Gryllotalpa spps	308
Scarabaeoidea	342
Locusta migratoria manilensis	315
Cerambycidae	298
Buprestidae	306
Hyphantria cunea	352
sum	2267

Table 3. Final dataset.

Type	Number
Cicadidae	4326
Gryllotalpa spps	3928
Scarabaeoidea	2934
Locusta migratoria manilensis	3501
Cerambycidae	3381
Buprestidae	3996
Hyphantria cunea	2772
sum	24,748

2.2. Detection Models

YOLOv5 is proposed by Ultralytics, which was improved based on YOLOv4. It is a detection model which takes on the advantages of previous versions and other networks, such as CSPNet and PANet [20,21], and gets a good tradeoff between accuracy and speed. The model structure is smaller than previous versions but more efficient. Its main structure diagram is shown in Figure 2. It introduces multi-scale prediction, and utilizes both FPN and PANet. FPN transfers deep semantic features to shallow layers, enhancing semantic expression on multiple scales. In contrast, PANet transmits the localization information of the shallow layer to the deep layer, and enhances the localization ability on multiple scales. YOLOv5 enhances multi-scale the semantic representation and localization capabilities through the combination of the two networks. It is mainly composed of C3, Conv and SPPF modules, as shown in Figure 2. YOLOv5 utilizes a depth multiple and width multiple to control the width and depth of the network. YOLOv5n is the network with the smallest depth and the smallest feature map width in this series. Other networks are continuously deepening and widening on this basis. In general terms, YOLOv5 has made improvements mainly by including the following four aspects: ① input (mosaic data augmentation,

adaptive anchor box calculation and adaptive image scaling); ② backbone (CSPNet and Focus module); ③ neck (FPN and PANet); ④ loss (replace IoU with CIoU).

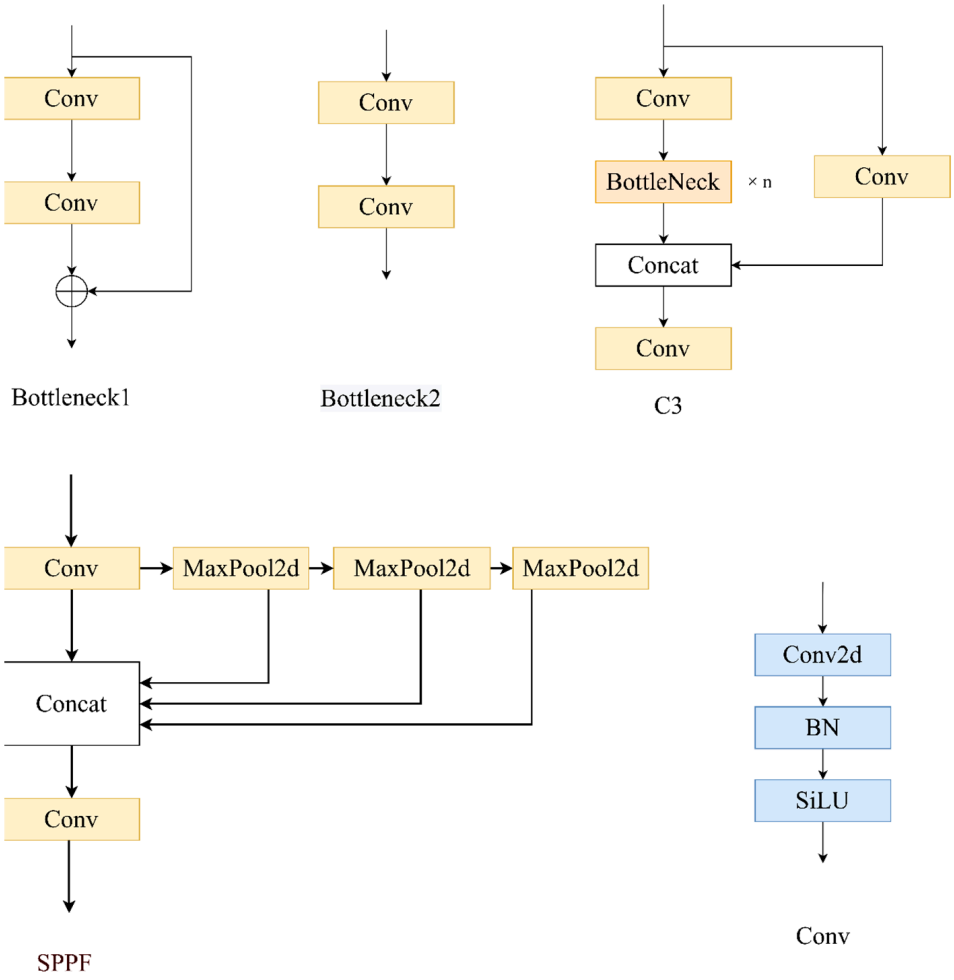


Figure 2. Main module of YOLOv5.

2.3. GhostNet Module

In order to fit the datasets better, neural networks often consist of a large number of parameters, especially in early fully connected layers. With the development of convolutional neural networks, we can use filters to reduce a lot of the parameters. Building a network which needs to finish a detect task, it often requires a lot of feature maps, which usually contain hundreds of channels; the model usually is big. Even the SOTA model has lots of layers, which means the model bloats. For a given neural network, model compression means that we can get it to easily deploy on embedding devices with fewer parameters. Researchers have designed some methods to compress the size of model. SequeezeNet [22] replaces a 3×3 with a 1×1 convolution kernel and reduces the channels of input. It achieves AlexNet-level accuracy on ImageNet with 50× fewer parameters, and the original model size of 240 Mb, is reduced to 4.8 Mb after model compression. Xception [23] uses split convolution operations and a more efficient feature fusion for the model parameters. Mobilenets [24] utilizes a series of depthwise separable convolutions which achieves a better performance with fewer parameters. Shufflenet [25] utilizes channel shuffle operations to improve the information flow exchange between channel groups. The group convolution can reduce parameters significantly, so it now gets widely used. Although these models achieve excellent performance with very few FLOPs, the correlation and redundancy between feature maps have never been well exploited. The remaining 1×1 convolution layers would still result in a lot of parameters. In order to avoid redundant parameters

which results in a large amount of consumption and makes deploying neural networks on embedding devices more convenient, Han et al. proposed a new method named GhostNet, which aimed to generate more feature maps with cheaper operations [26,27]. For the input image, $X \in \mathbb{R}^{c \times h \times w}$, where c, h,w are the number of input channels, the height of the input image and the width of the input image, respectively. In general, the operations of an arbitrary convolutional layer for producing n feature maps can be formulated as:

$$Y = X * f + b \tag{1}$$

where $*$ is the convolution operation, b the bias, $Y \in \mathbb{R}^{h' \times w' \times n}$, the output feature map with n channels and $f \in \mathbb{R}^{c \times k \times k \times n}$, the convolution filters in this layer. In addition, h' and w' are the height and width, respectively, of the output feature maps and $k \times k$ is the kernel size of the convolution filters f. During this convolution procedure [26], the required number of FLOPs can be calculated as $n \cdot h' \cdot w' \cdot c \cdot k \cdot k$, which is often as large as hundreds of thousands since the number of filters n and the channel number c are generally very larger(e.g., 256 or 512 or 1024).

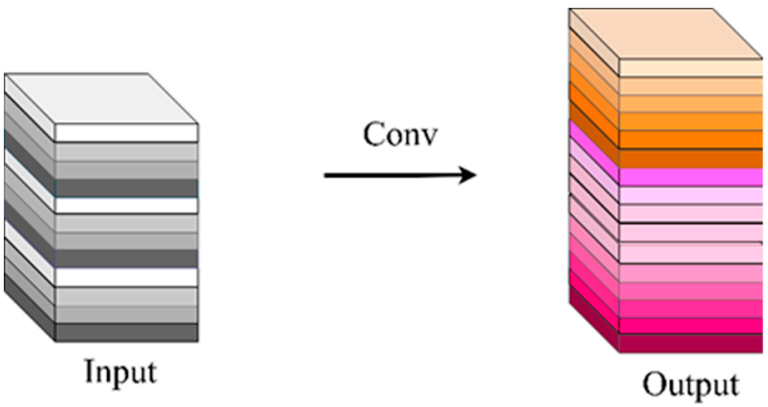
By visualizing some feature maps, we can discover that some features are similar, and we can get it by some cheap operations instead. Specifically, m intrinsic feature maps $Y' \in \mathbb{R}^{h' \times w' \times m}$ can be generated by Equation (2), where $f' \in \mathbb{R}^{c \times k \times k \times m}$ is the convolution filters, and it is similar with Equation (1). However, we only operate on partial convolutions, and the remaining feature maps are generated with a linear operation, as shown in Equation (3).

$$Y' = X * f' + b \tag{2}$$

$$y_{i,j} = \Phi_{i,j}(y_i'), \ \forall i = 1, \ldots, m, \ j = 1, \ldots, s, \tag{3}$$

where y_i' is the i-th intrinsic feature maps in Y', $\Phi_{i,j}$ the linear operation for y_i'generating the j-th ghost feature map $y_{i,j}$.

The kernel size for the linear operation can be choose as 3 or 5 or 7, there are no significant difference to improve the precision, but both of them have a significant reduction on weights and GFLOPs, as can be seen in Figure 3, which shows the GhostNet model.



(**a**)

Figure 3. *Cont.*

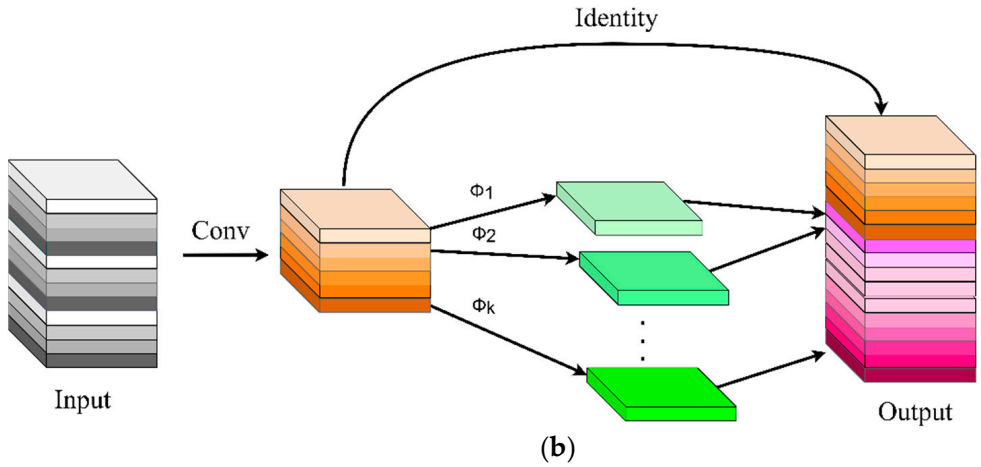


Figure 3. Ghost module, normal Conv is shown in (**a**) and Ghost Conv is shown in (**b**).

We have replaced the C3 module with C3Ghost and replaced the Conv module with GhostConv, as is shown in Figure 4. The linear operation is a group convolution [28], as is shown in Figure 5. In some cases, group convolution can indeed bring about better model results than standard 2D convolution, because group convolution can increase the diagonal correlation between adjacent layer filters and reduce the training parameters. It is not easy to overfit, which is similar to the effect of regularization. In general speaking, the parameters p utilizes standard 2D convolution as is shown in Equation (4), and the parameters P' is shown in Equation (5) with group convolution, where h_1, w_1, C_1, C_2, g are the filters height, the filters width, the input channels, the output channels and the number of groups, respectively. We can get similar feature maps with the $1/g$ times number of parameters.

$$P = h_1 \times w_1 \times C_1 \times C_2 \tag{4}$$

$$P' = h_1 \times w_1 \times C_1 \times C_2 \times \frac{1}{g} \tag{5}$$

The number of channels of input and the output of the Conv and the GhostConv module are the same. C_1 is the number of input channels and C_2 is the number of output channels. There are two Conv module in GhostConv, the number of hidden channels is half of the output. The details are shown in Figure 4.

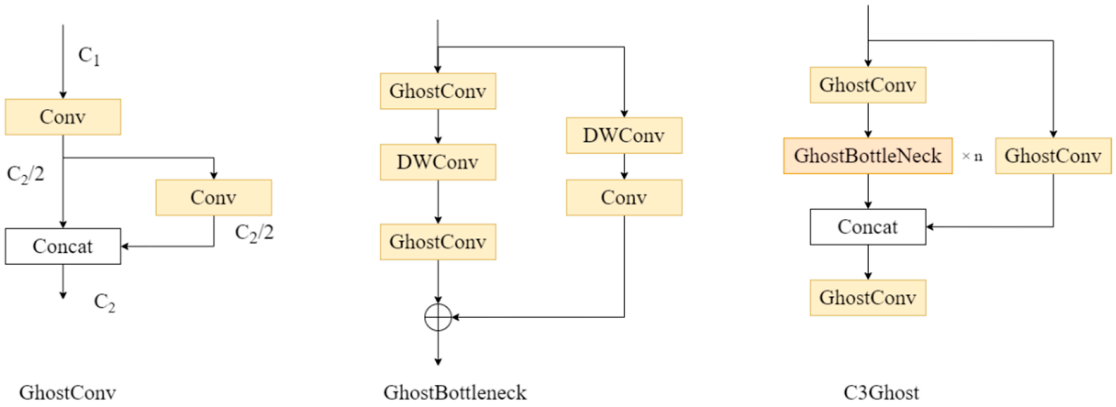


Figure 4. Main structure of YOLOv5 with GhostNet.

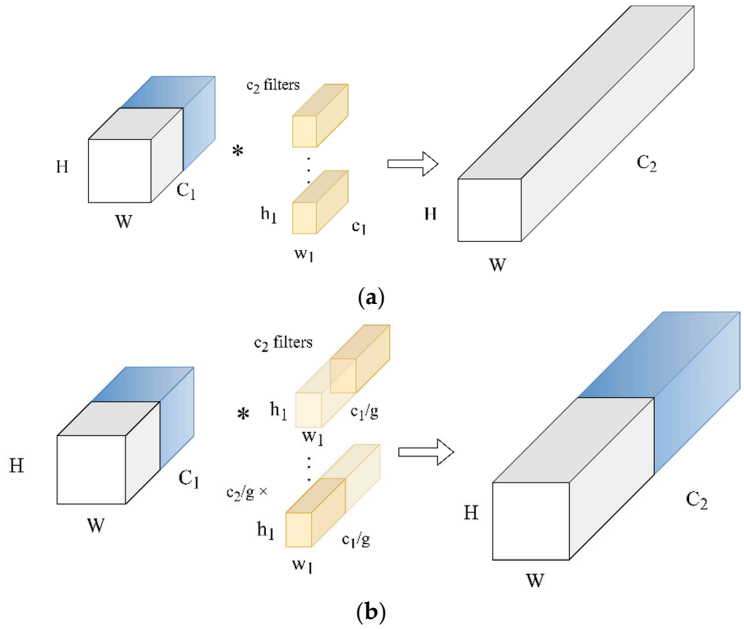


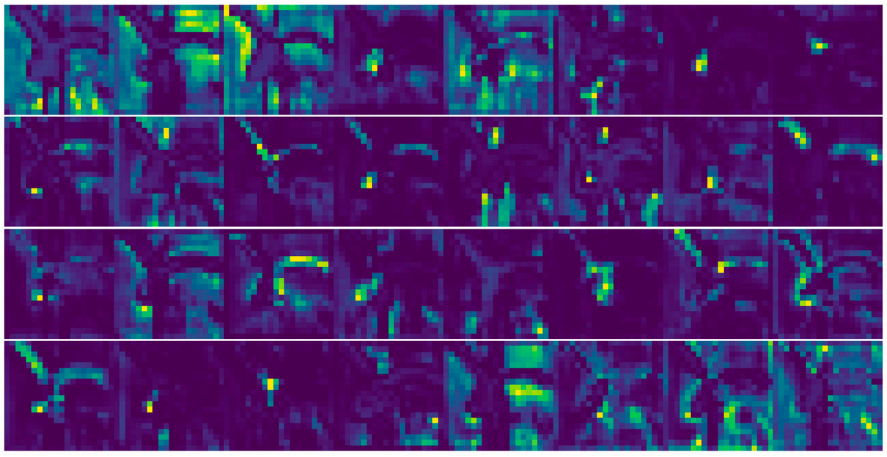
Figure 5. Standard 2D convolution is shown in (**a**) and group convolution is shown in (**b**).

3. Results

3.1. Feature Maps Visualization

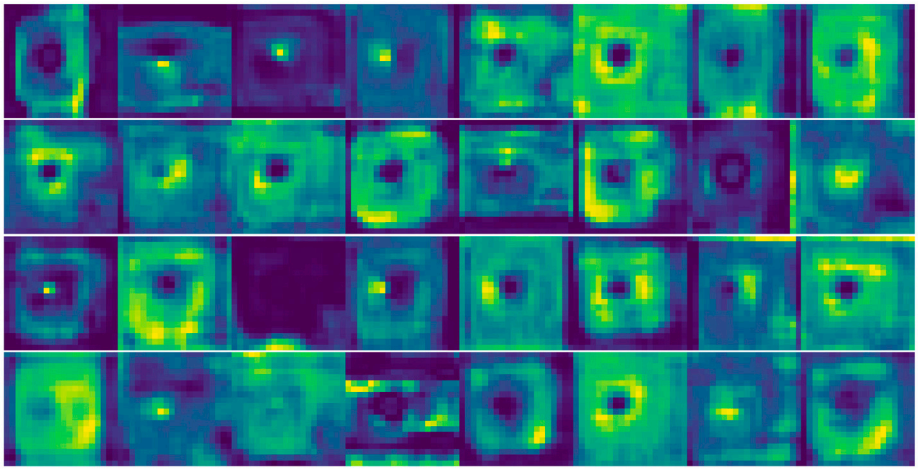
The visualization of feature maps is shown in Figure 6. By saving the convolution output of a specific layer, we get different feature maps by adding GhostNet in different places. We add GhostNet module in the head, in the backbone and add both respectively; the corresponding model is named HEAD-GHOST, BACKBONE-GHOST and ALL-GHOST. We find that the output images are similar to output images without GhostNet; it illustrates that we can replace C3 module with C3Ghost, and Conv with GhostConv to get analogous feature maps.

(**a**)

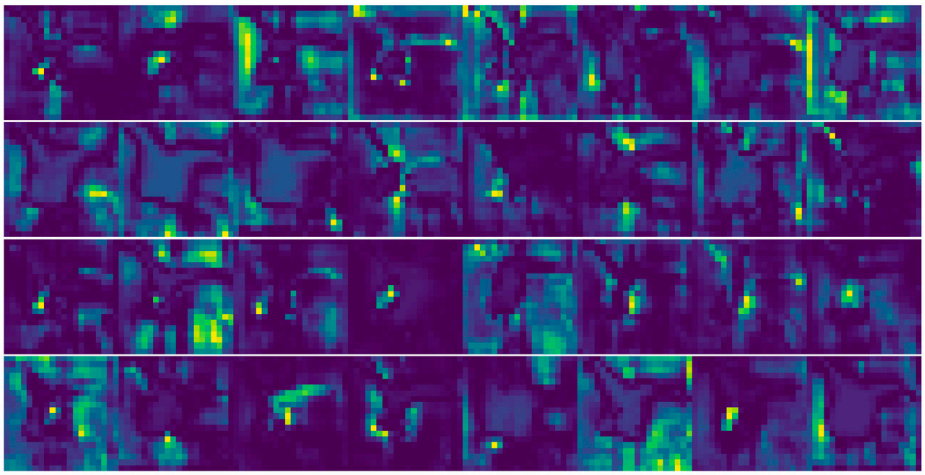


(**b**)

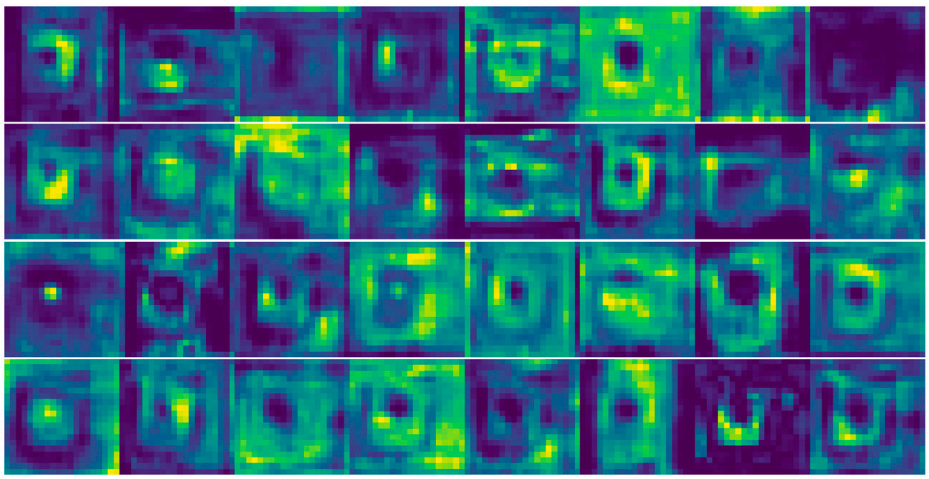
Figure 6. *Cont.*



(**c**)

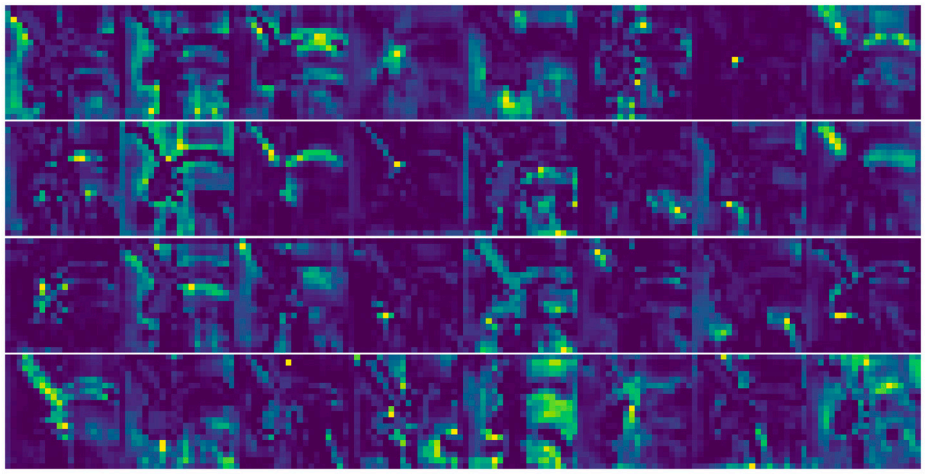


(**d**)

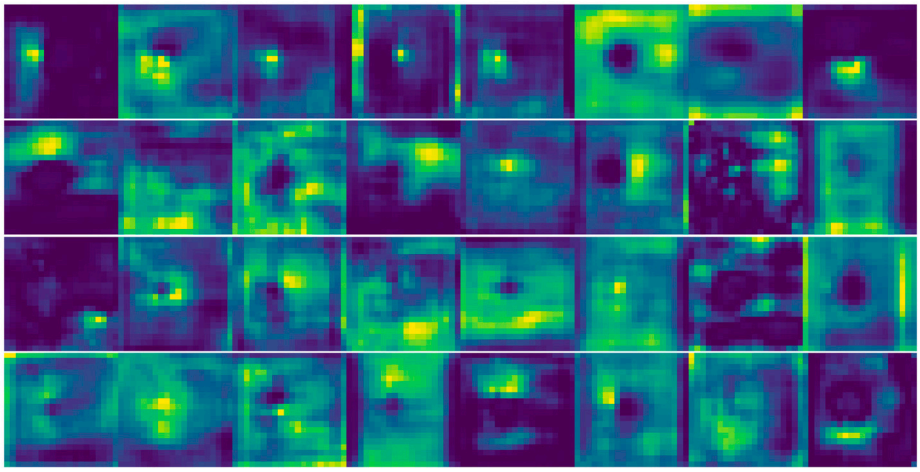


(**e**)

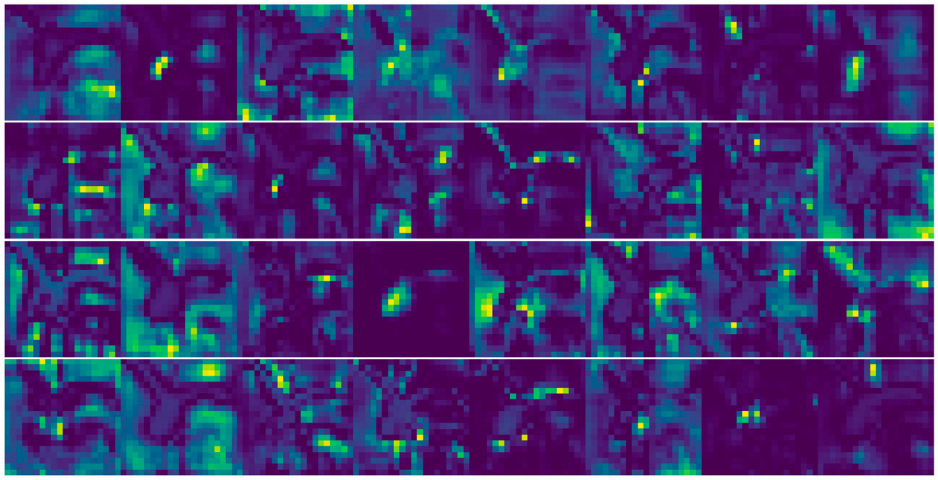
Figure 6. *Cont*.



(**f**)

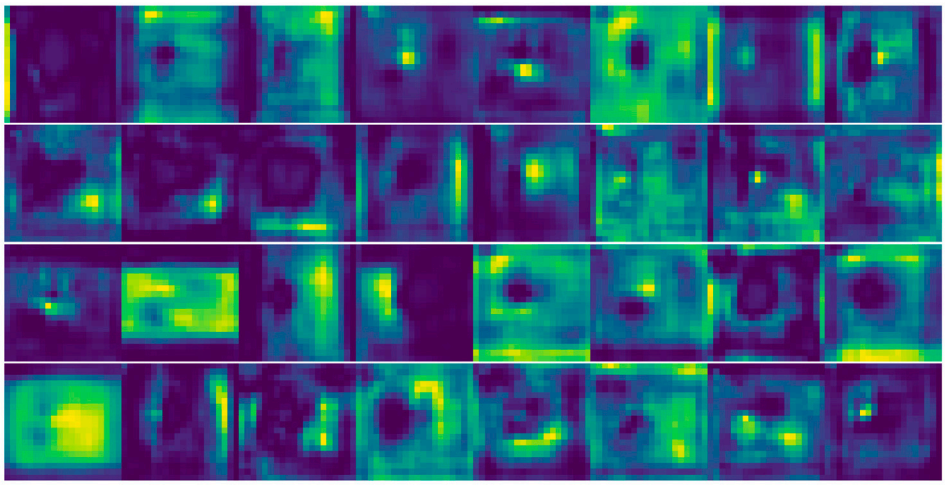


(**g**)



(**h**)

Figure 6. *Cont*.



(**i**)

Figure 6. Visualization of Feature maps: (**a**) is the input image; (**b**,**c**) are the eighth and the twenty-third layer of YOLOv5s, respectively; (**d**,**e**) are the eighth and the twenty-third layer of YOLOv5s-BACKBONE-GHOST, respectively; (**f**,**g**) are the eighth and the twenty-third layer of YOLOv5s-HEAD-GHOST, respectively; (**h**,**i**) are the eighth and the twenty-third layer of YOLOv5s-ALL-GHOST, respectively.

We use the structural similarity metric (SSIM) to evaluate the similarity of two images. As can be seen in Equation (6), where x and y are two images, μ_x, μ_y, σ_x, σ_y, σ_{xy} are the average of x, the average of y, the variance of x, the variance of y and the covariance of x, y, respectively, c_1, c_2 are two hyperparameters.

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{\left(\mu_x^2 + \mu_y^2 + c_1\right)\left(\sigma_x^2 + \sigma_y^2 + c_2\right)} \tag{6}$$

We get 32 images for every feature map from Figure 6. For brevity, we select the first ten images and calculate SSIM with the output of the original feature maps respectively, the heatmap is shown in Figure 7. The average score of SSIM is shown in Table 4. The SSIM is close to 0.8 by adding the Ghost module.

Since the detection network passes through the backbone part first and then the head part, we choose the last layer for head and backbone, which correspond to 8 and 23 respectively. Referring to Table 4 it can be seen that the mean SSIM with added Ghost in only one part (head or backbone) is higher than when adding both.

Table 4. Mean SSIM.

Model	Mean SSIM
ALL-GHOST-8	0.819
ALL-GHOST-23	0.817
BACKBONE-GHOST-8	0.804
BACKBONE-GHOST-23	0.847
HEAD-GHOST-8	0.830
HEAD-GHOST-23	0.827

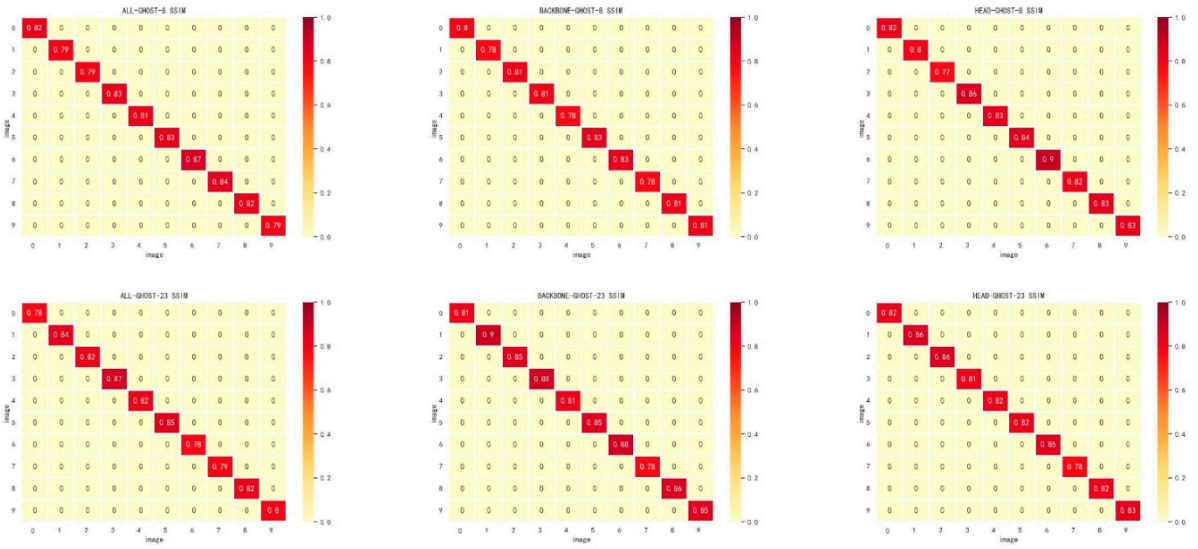


Figure 7. Heatmaps of SSIM compared to original images. ALL-GHOST-8 and ALL-GHOST-23 means the output of the eighth and the twenty-third layer in which the Ghost module is added both in the head and backbone. BACKBONE-GHOST-8 and BACKBONE-GHOST-23 means the output of the eighth and the twenty-third layer in which the Ghost module is added in backbone only. HEAD-GHOST-8 and HEAD-GHOST-23 means the output of the eighth and the twenty-third layer which Ghost module is added in head only.

3.2. Results on Pest Dataset

The results from comparing the performance of different models by adding GhostNet to different locations in terms of parameters, weights, GFLOPs, mAp and inference time, are shown in Table 5. The hardware environment of our experiments was an AMD Ryzen 5 3600 CPU, NVIDIA RTX2070 SUPER(8G), and the software environment was Ubuntu18.04.6 LTS, Pytorch1.9.0, Python 3.7. The epoch was set to 100 and other hyperparameters were the same as the original YOLOv5. Some of the main parameters such as momentum were set to 0.937, weight decay to 0.0005, warmup epochs to 3 and the augmentation was set to true. The results illustrated that compared to the network without GhostNet the model still worked well when GhostNet was added. A possible explanation is that the feature maps have no significant differences. This can be seen from the previous heatmaps in Figure 7. The corresponding semantic information loss was very small with the GhostNet module. However, for the detection network, there were slight differences when adding the GhostNet module in different places. The detection time was not significantly improved, but the weights of model was reduced almost by half, and mAp increased by nearly 1%. It illustrates that the GhostNet module can prune deep neural networks while maintaining a comparable performance for orchard pests.

Table 5. Parameters of each model.

Models	Params (Millions)	Weights (Mb)	GFLOPs
YOLOv5n	1.77	3.75	4.2
YOLOv5n-ALL-GHOST	**0.95**	**2.25**	**2.3**
YOLOv5n-HEAD-GHOST	1.42	3.12	3.6
YOLOv5n-BACKBONE-GHOST	1.29	2.88	2.9
YOLOv5s	7.03	13.8	15.9
YOLOv5s-ALL-GHOST	**3.7**	**7.52**	**8.2**
YOLOv5s-HEAD-GHOST	5.6	11.1	13.4
YOLOv5s-BACKBONE-GHOST	5.1	10.1	10.7

Table 5. *Cont.*

Models	Params (Millions)	Weights (Mb)	GFLOPs
YOLOv5m	20.89	40.3	48.1
YOLOv5m-ALL-GHOST	**8.55**	**16.8**	**18.4**
YOLOv5m-HEAD-GHOST	15.5	30.1	38
YOLOv5m-BACKBONE-GHOST	13.89	27	28.5
YOLOv5l	46.17	88.6	108
YOLOv5l-ALL-GHOST	**15.62**	**30.5**	**33.3**
YOLOv5l-HEAD-GHOST	32.7	63.1	82
YOLOv5l-BACKBONE-GHOST	29.02	56	59.3
YOLOv5x	86.25	165	204.3
YOLOv5x-ALL-GHOST	**25.09**	**48.7**	**53.3**
YOLOv5x-HEAD-GHOST	59.2	113	151.3
YOLOv5x-BACKBONE-GHOST	52.1	100	106.4

3.3. Results on Loss Curve

For the object detection task, in addition to the classification accuracy of the detected target, the prediction accuracy of its location was also important. Intersection over union (IoU) is one of the important metrics to measure the location accuracy, which is shown in Equation (7), where A is the ground truth, B the prediction box. The IoU loss is zero when the two rectangles are disjoint. In order to measure the regression positioning loss better, some researchers have considered the distance of two central points and have proposed DIoU, which is shown in Equation (8), where b, b^{gt}, d, c are the central point of the prediction box, the central point of ground truth, the distance of two central points of two boxes and the diagonal length of the smallest enclosing box covering two boxes, respectively. An outstanding regression loss should consider overlapping areas, center point distance and aspect ratio simultaneously. It is called CIoU, as is shown in Equations (9)–(11) and Figure 8, where v, w^{gt}, h^{gt}, w, h, α are the aspect ratio, the width and height of ground truth, the width and height of the prediction box and a hyperparameter, respectively. Training loss is shown in Figure 8, where classes loss and objectness loss utilizes cross-entropy loss.

$$IoU = \frac{A \cap B}{A \cup B} \tag{7}$$

$$DIoU = IoU - \frac{fulllength\rho^2\left(b, b^{gt}\right)}{c^2} = IoU - \frac{d^2}{c^2} \tag{8}$$

$$CIoU = IoU - \left(\frac{fulllength\rho^2(b, b^{gt})}{c^2} + \alpha v\right) \tag{9}$$

$$v = \frac{4}{\pi^2}\left(arctan\frac{w^{gt}}{h^{gt}} - arctan\frac{w}{h}\right)^2 \tag{10}$$

$$\alpha = \frac{v}{(1 - IoU) + v} \tag{11}$$

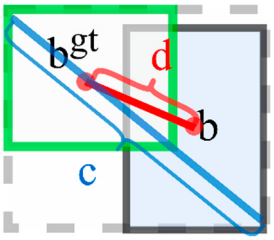


Figure 8. CIoU (Complete-IOU).

The rate of convergence with GhostNet in the head part is faster according to the train loss curve. It is probably due to the model compact, fewer parameters lead to faster convergence. There are only slight differences in the loss curve as is shown in Figures 9 and 10. Since the model performance in the loss curve is very similar after adding the ghost module, there is a higher mAp as is shown in Table 6. It indicates that GhostNet is effective.

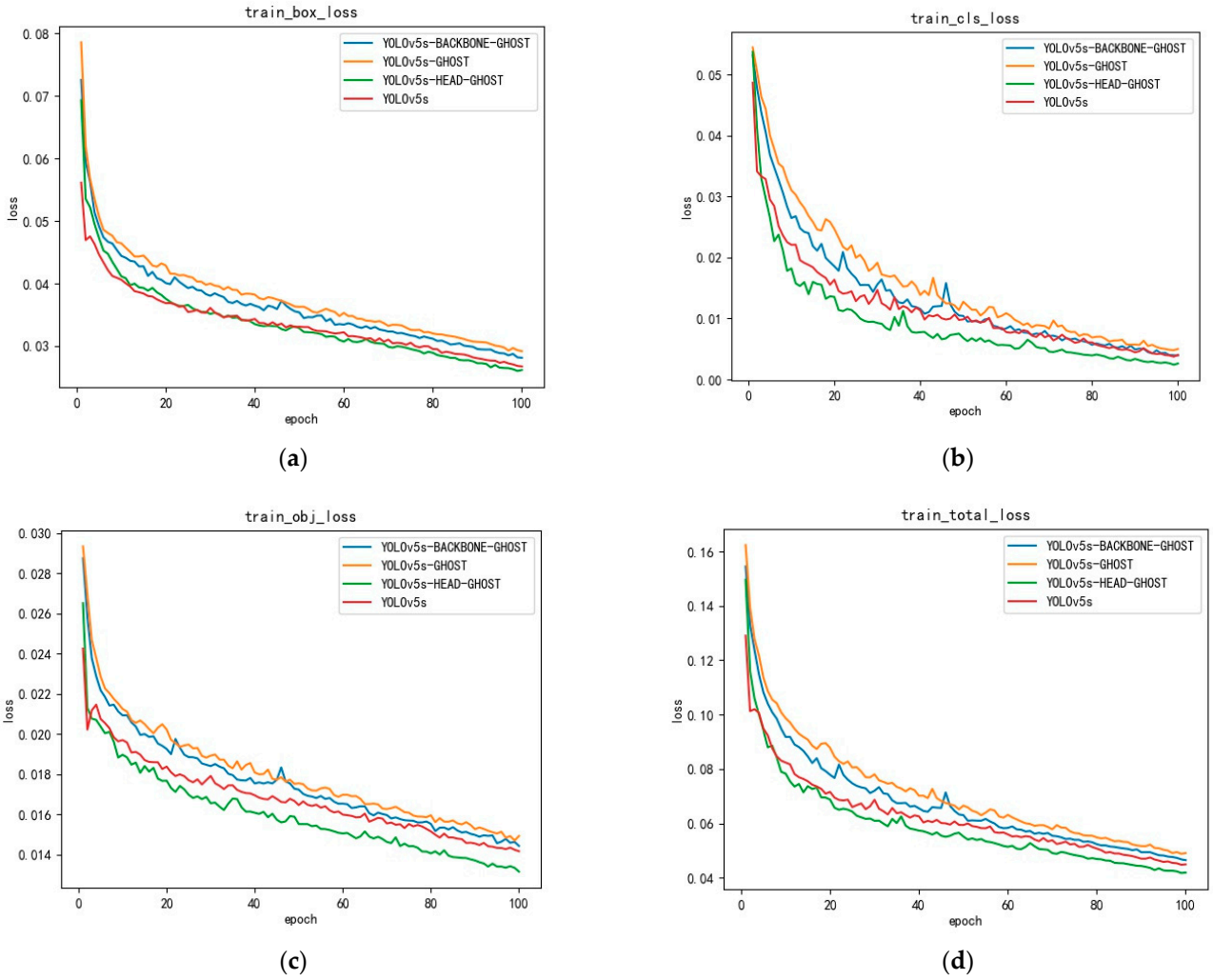


Figure 9. Train loss curve: (**a**) is the box loss, (**b**) classify loss, (**c**) object loss, (**d**) total loss.

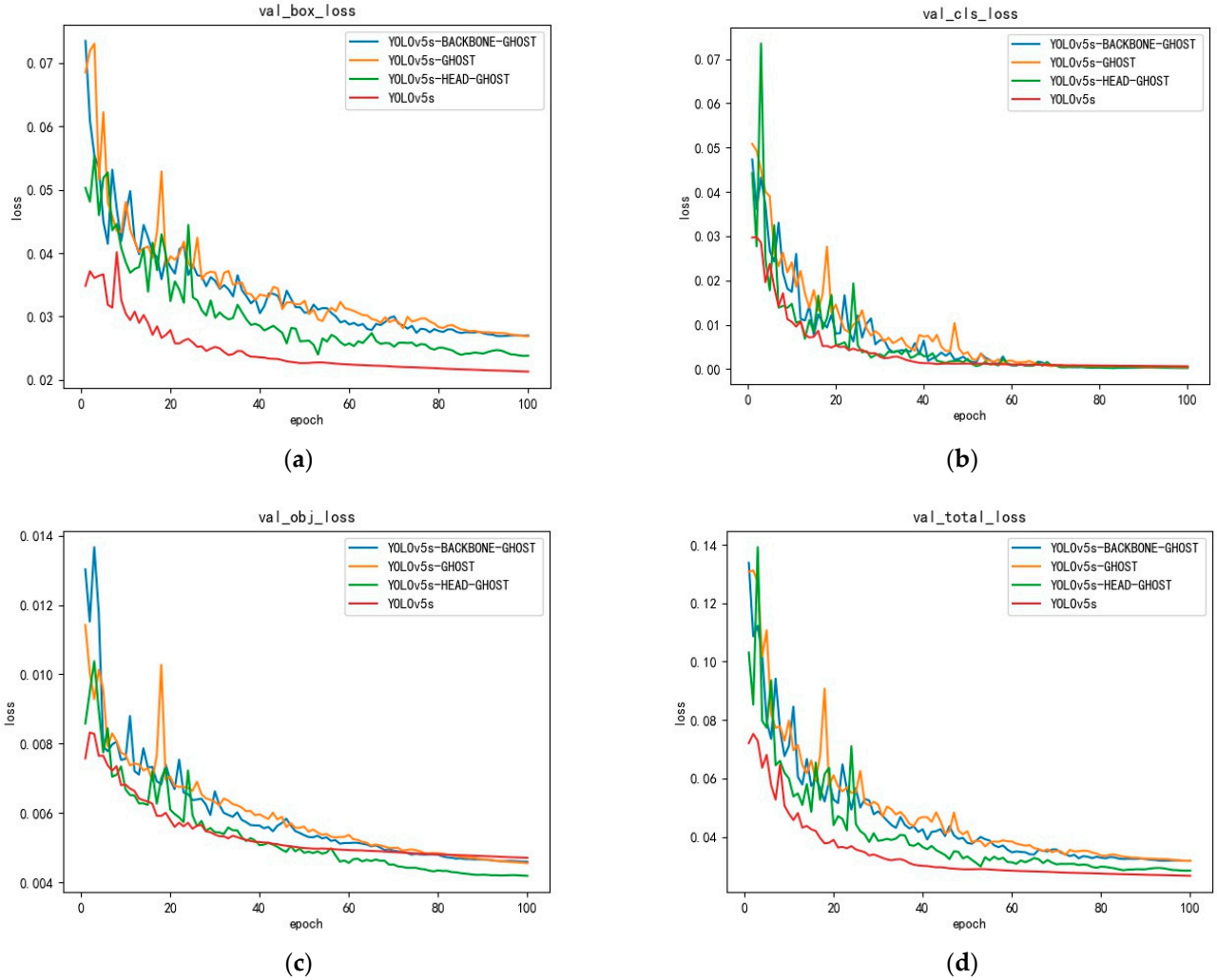


Figure 10. Val loss curve: (**a**) is the box loss, (**b**) classify loss, (**c**) object loss, (**d**) total loss.

Table 6. mAP and detection time of each model.

Models	mAP@0.5	Detection Time (ms)
Faster RCNN(ResNet50)	0.8134	117
Tiny-YOLOv3	**0.875**	**3.33**
YOLOv3	0.914	11.11
Tiny-YOLOv4	0.887	**2.22**
YOLOv4	**0.929**	12.22
YOLOv5n	0.95	**9**
YOLOv5n-ALL-GHOST	0.9513	10
YOLOv5n-HEAD-GHOST	**0.9649**	10
YOLOv5n-BACKBONE-GHOST	0.9499	11
YOLOv5s	0.9702	**11**
YOLOv5s-ALL-GHOST	0.9791	**11**
YOLOv5s-HEAD-GHOST	**0.9866**	**11**
YOLOv5s-BACKBONE-GHOST	0.9833	13
YOLOv5m	0.9863	**14.3**
YOLOv5m-ALL-GHOST	0.9704	16.3
YOLOv5m-HEAD-GHOST	**0.9898**	15.3
YOLOv5m-BACKBONE-GHOST	0.9843	15.3

Table 6. *Cont.*

Models	mAP@0.5	Detection Time (ms)
YOLOv5l	**0.9933**	28
YOLOv5l-ALL-GHOST	0.9666	**22**
YOLOv5l-HEAD-GHOST	0.9889	22.7
YOLOv5l-BACKBONE-GHOST	0.9778	24
YOLOv5x	**0.9945**	48
YOLOv5x-ALL-GHOST	0.9775	**32**
YOLOv5x-HEAD-GHOST	0.9919	39
YOLOv5x-BACKBONE-GHOST	0.9827	34

4. Discussion

For the original YOLOv5 model, we have compared Ghost module inserts in different positions. For a given input image, the results illustrate that there is no significant difference between analogous feature maps. The reason for these results may be caused by many repeated calculations during the convolution process. There are too many feature maps with huge channels, and some of them may only have subtle differences. We can utilize GhostNet to achieve similar feature maps while reducing the parameters, and this makes it more convenient to deploy on embedding devices. Moreover, it can save a lot of computation compared to traditional methods. Comparing to the other three models, the model which adds GhostNet only in the head section performs most efficiently. It is probable that there is still some feature information missed with GhostNet added, and the backbone network plays an important role in extracting features information. So, we recommend that GhostNet only need be added in the head section, and the network can achieve the same effect as the original model with 75% weights and lower GFLOPs, or even better mAp in some cases. The specific pruning effect depends on the depth of your network; the deeper the network, the better the pruning effect. If you pursue the ultimate pruning effect, you can add GhostNet to both the head part and the backbone part with $2\times$ fewer parameters, at least compared to the original model. The GhostNet is a plug-and-play module which is easily transferred to others classical models, and it can dramatically reduce computation. However, it is still necessary to consider the tradeoff between accuracy and model size at the same time for different memory situations.

5. Conclusions

To reduce the computational costs of deep neural networks and employ a model which is more convenient for embedding devices, this paper presents GhostNet on the original YOLOv5 model for building efficient neural architectures. By replacing the convolution kernel with a linear operation, these cheap operations will save a lot of computing resources. We find that the feature maps have high similarity through feature visualization, and can calculate the SSIM with the original image to confirm this conclusion. Comparing mAP, detection time and loss curve respectively, we can conclude that the GhostNet module is a plug-and-play module which is easily transferred to others classical models while still retaining a comparable performance. The experimental results prove that under normal conditions, adding a GhostNet module only in the head section is enough for a detection task on embedding devices to produce a higher mAp and a lower loss. If there is enough memory for embedding devices, there still needs to be consideration of the accuracy and parameters. In some cases, the accuracy may be reduced. Since GhostNet can obtain the effect of ordinary convolution through a series of cheap operations, we will consider the possibility of GhostNet replacing ordinary convolution in the future.

Author Contributions: Conceptualization, Y.Z. and W.C.; methodology, W.C.; validation, S.F. and J.J. investigation, R.S.; resources, Y.Z. and W.C.; writing-original draft preparation. Y.Z.; writing-review and editing, Y.Z. and W.C.; funding acquisition, W.C. and J.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported in part by the National Natural Science Foundation of China under Grant 32073028 and Grant 31702393; and in part by the Ningbo Public Welfare Key Project under Grant 2019C10098 and Grant 2019B10079; and in part by the Ningbo Natural Science Foundation 2019A610075.

Data Availability Statement: Not applicable.

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

References

1. King, A. Technology: The Future of Agriculture. *Nature* **2017**, *544*, S21–S23. [CrossRef] [PubMed]
2. Ma, B.; Jin, Z.M.; Jiang, X.C.; Wan, X.J.; Xiao, X.X.; Chen, L.X.; Lu, Y.J. Research progress on online monitoring technology of stored grain pests. *Grain Storage* **2018**, *47*, 27–31.
3. Voulodimos, A.; Doulamis, N.; Doulamis, A.; Protopapadakis, E. Deep learning for computer vision: A brief review. *Comput. Intell. Neurosci.* **2018**, *2018*, 7068349. [CrossRef] [PubMed]
4. Saxena, L.; Armstrong, L. A survey of image processing techniques for agriculture. In Proceedings of the Asian Federation for Information Technology in Agriculture; Australian Society of Information and Communication Technologies in Agriculture: Perth, Australia, 2014; pp. 401–413.
5. Chen, X.; Wu, Y.Z.; Zhang, Y.H.; Le, Y. Image recognition of stored grain pests based on deep convolutional neural network. *Chin. Agric. Sci. Bull.* **2018**, *34*, 154–158.
6. Ding, W.; Taylor, G. Automatic moth detection from trap images for pest management. *Comput. Electron. Agric.* **2016**, *123*, 17–28. [CrossRef]
7. Shen, Y.; Zhou, H.; Li, J.; Jian, F.; Jayas, D.S. Detection of stored-grain insects using deep learning. *Comput. Electron. Agric.* **2018**, *145*, 319–325. [CrossRef]
8. Li, R.; Wang, R.; Zhang, J.; Xie, C.; Liu, L.; Wang, F.Y.; Chen, H.B.; Chen, T.J.; Hu, H.Y.; Jia, X.F.; et al. An effective data augmentation strategy for CNN-based pest localization and recognition in the field. *IEEE Access* **2019**, *7*, 160274–160283. [CrossRef]
9. He, Y.; Zhou, Z.; Tian, L.; Liu, Y.; Luo, X. Brown rice planthopper (*Nilaparvata lugens* Stal) detection based on deep learning. *Precis. Agric.* **2020**, *21*, 1385–1402. [CrossRef]
10. Wang, F.; Wang, R.; Xie, C.; Yang, P.; Liu, L. Fusing multi-scale context-aware information representation for automatic in-field pest detection and recognition. *Comput. Electron. Agric.* **2020**, *169*, 105222. [CrossRef]
11. Xie, X.; Ma, Y.; Liu, B.; He, J.; Wang, H. A deep-learning-based real-time detector for grape leaf diseases using improved convolutional neural networks. *Front Plant Sci.* **2020**, *11*, 751. [CrossRef] [PubMed]
12. James, E.C.; Francisdomson, Z.; Rizalyn, P.; Jaymer, J.; Rolyn, D. Design and development of a stationary pest infestation monitoring device for rice insect pests using convolutional neural network and raspberry pi. *Jcr* **2020**, *7*, 635–638. [CrossRef]
13. Wang, H.X.; Li, Y.F.; Dang, L.M.; Hyeonjoon, M. An efficient attention module for instance segmentation network in pest monitoring. *Comput. Electron. Agric.* **2022**, *195*, 106853. [CrossRef]
14. Pang, H.T.; Zhang, Y.T.; Cai, W.M.; Li, B.; Song, R.Y. A real-time object detection model for orchard pests based on improved YOLOv4 algorithm. *Sci. Rep.* **2022**, *12*, 13557. [CrossRef] [PubMed]
15. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
16. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7263–7271.
17. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
18. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
19. Pang, H.T. Research on Intelligent Recognition Technology of Orchard Pests Based on Deep Learning. Master's Thesis, Zhejiang University, Hangzhou, China, 2021.
20. Wang, C.Y.; Liao, H.Y.M.; Wu, Y.H.; Chen, P.Y.; Hsieh, J.W.; Yeh, I.H. CSPNet: A new backbone that can enhance learning capability of CNN. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, 14–19 June 2020; pp. 390–391.
21. Wang, K.; Liew, J.H.; Zou, Y.; Zhou, D.; Feng, J. Panet: Few-shot image semantic segmentation with prototype alignment. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 9197–9206.
22. Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-level accuracy with 50× fewer parameters and <0.5 MB model size. *arXiv* **2016**, arXiv:1602.07360.
23. Chollet, F. Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1251–1258.
24. Howard, A.G.; Zhu, M.L.; Chen, B.; Kalenichenko, D.; Wang, W.J.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861. [CrossRef]

25. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6848–6856.
26. Han, K.; Wang, Y.H.; Tian, Q.; Guo, J.Y.; Xu, C.J.; Xu, C. Ghostnet: More features from cheap operations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 1580–1589.
27. Han, K.; Wang, Y.H.; Xu, C.; Guo, J.Y.; Xu, C.J.; Wu, E.H.; Tian, Q. GhostNets on Heterogeneous Devices via Cheap Operations. *Int. J. Comput. Vis.* **2022**, *130*, 1050–1069. [CrossRef]
28. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems—Volume 1 (NIPS'12), Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.