Article

Using Generative Module and Pruning Inference for the Fast and Accurate Detection of Apple Flower in Natural Environments

Yan Zhang ^{1,†} , Shupeng He ^{2,†} , Shiyun Wa ^{1,†} , Zhiqi Zong ^{2,†} and Yunling Liu ^{1,*}

¹ College of Information and Electrical Engineering, China Agricultural University, Beijing 100083, China; 2019308250102@cau.edu.cn (Y.Z.); shiyunwa@gmail.com (S.W.)

² Yantai Research Institute, China Agricultural University, Yantai 264670, China; 2019505430320@cau.edu.cn (S.H.); 2021505030412@cau.edu.cn (Z.Z.)

* Correspondence: liuyunling@cau.edu.cn

† These authors contributed equally to this work.

Abstract: Apple flower detection is an important project in the apple planting stage. This paper proposes an optimized detection network model based on a generative module and pruning inference. Due to the problems of instability, non-convergence, and overfitting of convolutional neural networks in the case of insufficient samples, this paper uses a generative module and various image pre-processing methods including Cutout, CutMix, Mixup, SnapMix, and Mosaic algorithms for data augmentation. In order to solve the problem of slowing down the training and inference due to the increasing complexity of detection networks, the pruning inference proposed in this paper can automatically deactivate part of the network structure according to the different conditions, reduce the network parameters and operations, and significantly improve the network speed. The proposed model can achieve 90.01%, 98.79%, and 97.43% in precision, recall, and mAP, respectively, in detecting the apple flowers, and the inference speed can reach 29 FPS. On the YOLO-v5 model with slightly lower performance, the inference speed can reach 71 FPS by the pruning inference. These experimental results demonstrate that the model proposed in this paper can meet the needs of agricultural production.

Keywords: generative module; pruning inference; object detection; YOLO; EfficientDet; apple flower

Citation: Zhang, Y.; He, S.; Wa, S.; Zong, Z.; Liu, Y. Using Generative Module and Pruning Inference for the Fast and Accurate Detection of Apple Flower in Natural Environments. *Information* **2021**, *12*, 495. <https://doi.org/10.3390/info12120495>

Academic Editor: Gholamreza Anbarjafari

Received: 9 November 2021

Accepted: 26 November 2021

Published: 29 November 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Apple flowers have a high nutritional value and solid practical uses. During the apple planting stage, apple flower thinning can effectively regulate the nutrient supply of apple trees, which is closely related to fruit weight and quality [1]. Moreover, apple flowers' growing status, for instance, bloom intensity or flower numbers [2], must be supervised since the flower is a relatively visual indicator for estimating fruit's further growth states at an early stage. Therefore, apple flower detection plays a vital role in apple tree yield estimation and health status assessment of apple trees. However, nowadays, apple flower detection technology is inefficient, inaccurate, and labor-intensive. In addition, apple flower images are prone to have multiple flower objects in an individual image due to the natural shape characteristic of being dense; a bunch of apple flowers is liable to cause mutual shielding. Mutable illuminance causes diverse clarity of objects with varying resolutions in captured images. Thus, implementing an automatic apple flower detection to improve the efficiency of apple flower detection and increase farmers' economic efficiency is necessary and urgent.

In recent years, machine learning has been widely applied to several agricultural studies, and some results have been achieved [3]. Researchers have applied computer vision techniques in apple quality inspection [4–6], apple pesticide residues [7], apple size assessment [8,9], and apple detection [10]. Nahina Islam et al. explored the potential of

machine learning algorithms for weed and crop classification in UAV images. They discovered that conventional RF and SVM algorithms are efficient and practical to use [11]. Jie Xu et al. proposed a multi-class classification model based on SVM and ANNs to estimate the occurrence of frost disasters towards tea [12]. Some traditional machine learning methods used above have high speed for real-time reference due to their relatively straightforward computation, whereas, since being trained on chosen features, they typically share with low accuracy. Especially in this paper's research field, these techniques do not perform well if previously unseen datasets occur, consisting of different flower species acquired under different conditions.

Additionally, some scholars have proposed various methods and recently made progress in research concerning apple flower detection. To build a pollination robot system, Lim et al. [13] proposed a novel method based on Faster R-CNN and Single Shot Detector (SSD) Net and achieved an accuracy of 91.9%. To detect and count tomato flowers from images effectively, Afonso et al. [14] first applied a set of grayscale transformations and then threshold and combined them by a logical binary AND operation with a recall of 79% and precision of 77%. To recognize and detect flower images accurately, Tian et al. [15] introduced SSD deep learning techniques into this field and achieved an accuracy of 87.4% based on the evaluation standard of Pascal VOC2012. Balvant V. Biradar et al. [16] proposed a method that uses Gaussian low-pass filter and morphological operations for pre-processing the flower images and a global thresholding technique using OTSU's algorithm to segment flower regions. Results have shown that the accuracy is over 92% to detect and count the number of flowers from flower images. Dihua Wu et al. [1] proposed a channel pruning-based deep learning algorithm for apple flower detection. The mAP using the proposed method was 97.31%; the detection speed was 72.33 f/s. Kaiqiong Sun et al. [17] proposed an automated apple, peach, and pear flower detection method, achieving an F1 score at pixel-level up to 89.6% on one of the apple datasets and an average F1 score of 80.9% on the peach, pear, and another apple datasets. Guy Farjon et al. [18] presented a visual flower detector based on a deep convolutional neural network, followed by a blooming level estimator and a peak blooming day finding algorithm. The trained detector detected flowers on trees with an Average Precision (AP) score of 0.68.

Therefore, inspired by the previous scholars' research and the issues mentioned above in apple flower detection, we propose an optimization method based on the generative module and pruning inference and apply it to the mainstream detection network. The effectiveness of this method is verified experimentally. GM-EfficientDet-D5 can achieve 90.01% precision, 98.79% recall, and 97.43% mAP, compared to the not optimized EfficientDet-D5 with 3.98%, 7.46%, and 7.48% improvement in these three indexes. Moreover, considerable variations in apple flower images are considered based on maturity, light, genotype, and orientation. We also test the detection performance on various sizes of apple flowers, and experimental results are satisfactory and outperform those of other models.

The rest of this paper is divided into four parts: the Materials and Methods section introduces the dataset and design details of the generative module and pruning inference; the Results section shows the experimental process and results as well as their analysis; the Discussion section conducts numerous ablation experiments verifying the effectiveness of the optimized method; and the Conclusion section summarizes the whole paper.

2. Materials and Methods

2.1. Dataset Analysis

The data were collected in Taolin Village, Changping District, Beijing (40°13' E, 116°25' N). A total of 2158 apple flower images were collected from April to June 2021, including 200 images collected from the Internet. The pixel resolution of the images was 2250 × 2250 pixels, and three shooting distances were included, as shown in Figure 1. Before image pre-processing, we manually labeled the apple flowers using the Labelme library in Python.

Figure 1. Image dataset was collected in Taolin Village in three scales.

There are several difficulties in processing this dataset: 1. Apple flowers are too dense, causing mutual obscuration. 2. The colors of apple flowers are different. 3. The appearance of apple flowers varies depending on maturity, light conditions, genotype, and head orientation.

By further analyzing the data samples, we found that the distribution of apple flowers' number in each image of the dataset varies, as shown in Figure 2A,B. Most of them are in the range of 5–20. Among them, there are three images without apple flowers and one image with 103 flowers, as shown in Figure 2C. These cases are too sparse or too dense, which hinders the model training.

Figure 2. Dataset overall. (A) is the distribution of apple flowers' number in each image; (B) is the distribution of different scales of apple flowers in each image; (C) shows four samples with different numbers of apple flowers.

2.2. Data Augmentation

2.2.1. Simple Augmentation

In this paper, we referred to the method proposed by Alex et al. [19]. We used image flipping, image translation, and image scaling for simple data augmentation, as shown in Figure 3. Image flipping and image translation mainly improve the model's accuracy by

increasing the amount of data. Image scaling is to achieve the learning of high-dimensional features by low-frequency networks. The above data augmentation methods are achieved by using affine transformation.

Figure 3. Simple data augmentation.

2.2.2. Advanced Augmentation

1. Mixup [20] is designed to solve the problem of colossal memory loss and the unsatisfactory sensitivity of the network to adversarial samples, as shown in Figure 4A. Since the model we used includes the generative module, enhancing the sensitivity of adversarial samples can improve the accuracy of the generative module, thus improving the regularization of final generated images. Mixup will encourage the model to have a linear understanding, which means that judgments on a sample are not so absolute, thus reducing overfitting.
2. Cutout [21] randomly cuts out part of the sample and fills it with a particular pixel, and the classification result remains unchanged. The Cutout is done by masking the image with a fixed size rectangle, and all values are set to 0 or other solid color values within the rectangle, as shown in Figure 4B. Cutout enables the convolutional neural network to use the global information of the whole image instead of the local information of some minor features.
3. CutMix [22]. The CutMix method is to cut off part of the region. Instead of filling 0 pixels, the region pixel values of other data in the training set are stochastically filled, as shown in Figure 4C. CutMix enables the model to identify two targets from a local view of an image, improving the training efficiency. Moreover, it enables the model to focus on the areas where the target is difficult to distinguish. However, there is no information in some areas, which will affect the training efficiency.
4. SnapMix [23]. The SnapMix method randomly cuts out some areas in the sample. It fills them with a particular patch from other images stochastically, and the classification label remains unchanged, as shown in Figure 4D.
5. Mosaic [24] can utilize multiple images at once. The most crucial advantage of Mosaic is that it can enrich the background of the detected objects. The data of multiple images will be counted in the BatchNorm calculation, which can effectively improve the model's generalization. In this paper, we used multiple images containing apple flowers between 5 and 10 to generate a single image containing at least 20 apple

flowers by Mosaic, as shown in Figure 4E. In this way, we improve the recognition performance of the model for high-density images.

Figure 4. Illustration of five augmentation methods. (A) Mixup; (B) Cutout; (C) CutMix; (D) SnapMix; (E) Mosaic.

As a result, the data set will be expanded from 2158 image samples to 37,890 data samples, and the result of data augmentation is shown in Table 1.

Table 1. Distribution of the dataset.

	Large	Medium	Small	Total
Original dataset	1044	1022	46	2158
After data augmentation	15,660	15,330	6900	37,890
Training set	14,094	13,797	6210	34,101
Validation set	1566	1533	690	3789

2.3. Generative Module

As mentioned in the analysis of the dataset characteristics, there are high-density small object detection scenarios in practical applications. The general approaches to solve the small target detection problem include: increasing the resolution of the input image, which increases the computational complexity, and multi-scale feature representation, which makes the results uncontrollable.

At present, the mainstream detection network incorporates the Feature Pyramid Network (FPN) [25]. After the backbone extracts the features, it contains the neck network with the fusion of deep feature maps and shallow feature maps. This structure improves the detection ability of the network for different scales of objects. However, it also makes the network complex and has the possibility of overfitting. Therefore, we proposed generative module (GM) in this paper, which aims to mitigate possible overfitting due to network complexity. This module enhances the detection network robustness by adding asymmetric generative sub-network branches to regularize the results. Take the structure of YOLO as an example, and its structure is shown in Figure 5.

In this paper, we use CGAN, CVAE, and CVAE-GAN for generative module implementation.

1. VAE [26] is suitable for generating unseen data but cannot control the generated content. CVAE (Conditional VAE) [27] can generate the data you want by specifying its label during data generation. Therefore, CVAE can be used as an implementation of generative module. When generating data, we first sampled the data from a normal distribution, spliced in the label of the generated data, and passed the spliced vector into the decoder; thus, we can generate the data corresponding to the label as shown in Figure 6A.
2. GAN [28] generator can only generate images based on random noise, and it has no control over the specifics of which labeled image is generated. Moreover, the discriminator can only receive the image input to discriminate whether the image is from the generator. The CGAN [29] adds additional information to the generator's input and discriminator of the GAN. If this additional information is the image's label, the generator can be controlled to generate the image with a specific label, as shown

in Figure 6B. Therefore, CGAN can be used as an implementation of generative module.

3. CVAE-GAN. The network structure of CVAE-GAN is shown in Figure 6C, which combines the features of CVAE and CGAN. Although it helps to improve the quality of generated images, the units make the network more complex and may reduce the network speed during inference.

Figure 5. Illustration of generative module on YOLO-v5 structure.

Figure 6. Illustration of three implementations of generative module.

2.4. Pruning Inference

In mainstream target detection networks, to improve object detection capability in different scales, the backbone network is always followed by a neck sub-network and a detection head sub-network, which is often based on FPN. However, this also causes

the problem that the network is too complex and not easy to train. For the dataset in this paper, apple flowers that are too large in size and scale hardly appear in the same image. Nevertheless, based on the existing network training method, each image will still be extracted several times and then up-sampled for feature fusion to output the result. Therefore, we proposed pruning inference (PI). The core idea of pruning inference is that during the training process, if the lower branch network has no higher loss than the upper networks, then the input of upper networks will be set to 0 to achieve structural pruning and improve the training speed. For example, in the training process of the FPN and UNet [30] structures, pruning is performed, and the pruning result is shown in Figure 7.

Figure 7. Illustration of pruning process of feature pyramid network and UNet.

In the network inference stage, to obtain the detection results in real-time despite the lack of computing power on the mobile, the generative module can also be deactivated by setting the input to zero to improve the inference speed.

2.5. Loss Function

The loss function of our model consists of three parts: box coordinate error, *CIoU* error, and classification error (see Formulas (1)–(4)). Box coordinate error (x_i, y_i) is the center position coordinate of the box predicted, (w_i, h_i) is the width and height of the predicted box. Correspondingly, $(\hat{x}_i, \hat{y}_i)$ and $(\hat{w}_i, \hat{h}_i)$ are the labeled ground truth box coordinates and size. λ_{coord} and λ_{noobj} is constant; $K \times K$ is the number of grids; M is the total number of predicted boxes; I_{ij}^{obj} is 1 when the i th grid contains a detection target and 0 in other cases.

$$Loss = Loss_{bounding_box} + Loss_{ciou} + Loss_{classification} \quad (1)$$

$$Loss_{bounding_box} = \lambda_{coord} \sum_{i=0}^{K \times K} \sum_{j=0}^M I_{ij}^{obj} (2 - w_i \times h_i) [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \lambda_{coord} \sum_{i=0}^{K \times K} \sum_{j=0}^M I_{ij}^{obj} (2 - w_i \times h_i) [(w_i - \hat{w}_i)^2 + (h_i - \hat{h}_i)^2] \quad (2)$$

$$Loss_{ciou} = \sum_{i=0}^{K \times K} \sum_{j=0}^M I_{ij}^{obj} [\hat{C}_i \log(C_i) + (1 - \hat{C}_i) \log(1 - C_i)] + \lambda_{noobj} \sum_{i=0}^{K \times K} \sum_{j=0}^M I_{ij}^{noobj} [\hat{C}_i \log(C_i) + (1 - \hat{C}_i) \log(1 - C_i)] \quad (3)$$

$$Loss_{classification} = \sum_{i=0}^{K \times K} I_{ij}^{obj} \sum_{c \in classes} [\hat{p}_i(c) \log(p_i(c)) + (1 - \hat{p}_i(c)) \log(1 - p_i(c))] \quad (4)$$

Zheng [31] proposed a more effective *IoU* calculation method, *CIoU*, whose formula is Formula (5). $C_i = Pr(Object) \times CIoU$.

$$CIoU = 1 - IoU + \frac{\rho^2(A, B)}{c^2} + \alpha v \quad (5)$$

The categories of classification are defined in the model as two categories, namely positive and negative. For each ground truth box, the prediction box and its *IoU* are calculated. The largest *IoU* is a positive class, while the others are negative classes.

2.6. Warm-Up

Warm-up [32] is a training strategy. In the pre-training stage of the model, some epochs or steps are trained with a small learning rate, and then the weights are modified to the preset learning rate for training. At the beginning of training, the model's weights are stochastically initialized, and the model's understanding level of data is 0. Suppose a large learning rate is adopted at the beginning. In that case, the model may shock. Subsequently, we adopt a Warm-up method to solve such a problem. The Warm-up method first trains with a lower learning rate, making the model data have certain prior knowledge, and then uses the preset vector of the training, making the model convergence faster and improving the effect. Finally, a low study rate is used to continue to explore to prevent missing local optimal points. For example, train the model with a 0.1 learning rate until the error is lower than 80%, then use a 0.1 learning rate to train it.

The above Warm-up is a constant Warm-up, whose deficiency is that changing from a minimal learning rate to a relatively large one may cause a sudden increase in training error. Therefore, Facebook proposed a gradual Warm-up in 2018 to solve this problem. The Warm-up starts from the initial small learning rate; it slightly speeds up each step until we reach the relatively large learning rate. The ultimately reached rate is initially set and used for the following training. This paper tries *exp* Warm-up, firstly increases linearly from a minimal value to the preset learning rate, then decays according to *exp* function law. *cos* Warm-up is tried as well. According to the *cos* function law, its learning rate increases linearly from a minimal value to a preset value and then decays according to the *cos* function law. The learning rate changing curves of the two Warm-up strategies are shown in Figure 8.

Figure 8. The learning rate of two warm-up schemes.

The principle of cosine decay is shown in Formula (6). Among it, i represents the number of iterations, η_{max}^i and η_{min}^i represent the maximum and minimum values of the learning rate, respectively, and T_{cur} represents the number of epochs currently executed. In contrast, T_i represents the total number of epochs in the number i step.

$$\eta_t = \eta_{min}^i + \frac{1}{2}(\eta_{max}^i - \eta_{min}^i)(1 + \cos(\frac{T_{cur}}{T_i}\pi)) \quad (6)$$

2.7. Evaluation Metrics

In order to verify the performance of the model, four indicators, including the Precision (P), Recall (R), mAP , and FPS, were adopted for the evaluation in this paper. When the Intersection over Union (IoU) ≥ 0.5 , it is a true case. When the $IoU \leq 0.5$, it is a false positive case. When the $IoU = 0$, it is a false negative case. The mAP is the average of Average Precision (AP) value when an apple flower is detected. The higher the value is, the better the detection result of an apple flower. The calculations of the P , R , and mAP are shown in the following Equations (7)–(9).

$$P = \frac{TP}{TP + FP} \quad (7)$$

$$R = \frac{TP}{TP + FN} \quad (8)$$

$$mAP = \frac{\sum_{i=1}^k (AP_i)}{k} \quad (9)$$

3. Results

3.1. Experiment

3.1.1. Equipment

The complete model training and validation process was implemented by a personal computer (processor: Intel(R) i9-10900KF@3.7 GHz; operating system: Ubuntu 18.04, 64 bits; memory: 16 GB). The training speed was optimized in Graphics Processing Unit (GPU) mode (NVIDIA RTX 3080 10 GB). Relevant model parameters (such as the base learning rate) adopted in this study are presented in Table 2.

Table 2. Distribution of the dataset.

Model Parameters	Values
Initial learning rate	0.02
Image input batch size	2
Gamma	0.1
Maximum iterations	200,000

3.1.2. Baseline Experiment

Since each model of the YOLO series [24,33,34], SSD series [35–37], and EfficientDet series [38] contained many sub-models, benchmarks were performed on all sub-models of these three network series. The experimental results are shown in Figures 9–12. It could be concluded that YOLO-v5, RefineDet, and EfficientDet-D5 performed best among the three network series in Table 3. Thus, subsequent experiments would adopt these three sub-models.

Figure 9. Training curves of accuracy and loss against number of iterations on YOLO series.

Figure 10. Training curves of accuracy and loss against number of iterations on SSD series.

Figure 11. Training curves of accuracy and loss against number of iterations on EfficientDet series, part I.

Figure 12. Training curves of accuracy and loss against number of iterations on EfficientDet series, part II.

Table 3. Comparisons of different detection network series' performance (in %).

Model	Precision	Recall	mAP	FPS
YOLO-v3	84.77	94.19	90.97	39
YOLO-v4	85.12	89.27	89.13	36
YOLO-v5	87.13	92.75	91.82	42
SSD	71.03	82.49	80.34	17
FSSD	81.61	93.37	91.47	21
RefineDet	84.95	93.39	91.77	23
EfficientDet-D2	84.57	88.19	86.39	47
EfficientDet-D3	86.22	89.81	87.22	41
EfficientDet-D4	85.71	91.49	88.69	42
EfficientDet-D5	86.03	91.33	89.91	35
EfficientDet-D6	85.71	91.49	83.47	33
EfficientDet-D7	85.24	90.98	84.14	29

3.2. Results and Analysis

In order to verify the validity of the proposed model, we used an enhanced validation set containing 3789 images tested on multiple detection networks, including Faster RCNN [39] and Mask RCNN [40]. The experimental results are shown in Table 4. The network after adding the generative module achieves precision, recall, and mAP up to 90.01%, 98.79%, and 97.43%. Compared with the original model before optimization, the generative module can improve the mAP of the model by as much as 7.65% on Mask RCNN. The above experimental results show that the generative module proposed in this paper can effectively improve the model performance. The effectiveness of GM-EfficientDet-D5 is demonstrated in Figure 13.

Figure 13. Demonstration of GM-EfficientDet-D5's effectiveness. (A) is large scale; (B) is medium scale; (C) is large scale.

Table 4. Performance of different models (in %).

Model	Precision	Recall	mAP	FPS
Faster RCNN	79.87	87.93	84.18	37
Mask RCNN	81.99	91.03	87.26	39
GM-Mask RCNN	85.39	95.60	94.91	33
YOLO-v5	87.13	92.75	91.82	42
GM-YOLO-v5	89.77	96.48	93.90	38
RefineDet	84.95	93.39	91.77	23
GM-RefineDet	87.41	97.11	93.38	17
EfficientDet-D5	86.03	91.33	89.91	35
GM-EfficientDet-D5	90.01	98.79	97.43	29

From the above experimental results, we selected YOLO-v5, the detection network with the best performance without generative module optimization, and GM-EfficientDet-D5, the detection network with the best performance with generative module optimization. We further analyzed the performance of these two networks at three scales, and the results are shown in Table 5. It is found that the generative module can effectively improve the performance of detection networks on small and medium scales.

Table 5. Comparisons of detection performance for different sizes of apple flowers. (P): Precision, (R): Recall (in %).

Object Size	Small	Medium	Large
YOLO-v5 (P)	67.11	87.01	87.29
YOLO-v5 (R)	71.98	91.99	92.94
YOLO-v5 (mAP)	63.87	91.82	91.83
GM-EfficientDet-D5 (P)	78.18	89.93	90.25
GM-EfficientDet-D5 (R)	85.21	98.83	98.79
GM-EfficientDet-D5 (mAP)	83.94	97.42	97.45

4. Discussion

4.1. Ablation Experiment about Generative Module

In this paper, we proposed three possible implementations of the generative module. Furthermore, we have compared them on GM-EfficientDet-D5 and GM-YOLO-v5-PI, which are the best performers in the above experiments. The results are shown in Table 6.

Table 6. Performance of different generative module implementation on different models.

Model	GM	Precision	Recall	mAP	FPS
GM-EfficientDet-D5	CGAN	90.01	98.79	97.43	29
	CVAE	89.17	96.33	97.41	30
	CVAE-GAN	90.03	98.50	97.61	25
GM-YOLO-v5-PI	CGAN	85.28	89.20	88.47	71
	CVAE	84.71	89.31	89.02	76
	CVAE-GAN	91.27	94.12	93.18	47

The experimental results show that using CVAE-GAN to implement the generative module has the best network performance. However, this implementation can seriously reduce the inference speed of the model. Taking GM-YOLO-v5-PI as an example, CVAE-GAN reduces the inference speed of YOLO-v5 to 61.8% compared with CVAE. In comparison, CGAN and CVAE are approximately the same in terms of efficiency and performance. Specifically, the generative module inference using CVAE is the fastest, with slightly lower performance than the CGAN implementation.

4.2. Ablation Experiment about Pruning Inference

In this ablation experiment, we used the best-performed GM-EfficientDet-D5 model to verify the effectiveness of the proposed pruning inference. The experimental results are shown in Table 7.

Table 7. Performance of different pruning strategy on different models.

Model	Strategy	Precision	Recall	mAP	FPS
GM-EfficientDet-D5	baseline	90.01	98.79	97.43	29
	PI	89.13	98.10	96.18	51
EfficientDet-D5	baseline	86.03	91.33	89.91	35
	PI	85.91	89.18	88.33	53
GM-YOLO-v5	baseline	89.77	96.48	93.90	38
	PI	89.14	96.27	93.15	63
YOLO-v5	baseline	87.13	92.75	91.82	42
	PI	85.28	89.20	88.47	71

The experimental results show that the application of pruning inference affected Precision, Recall, and mAP indexes limitedly, but the FPS index was significantly improved. In particular, the GM-YOLO-v5 model has almost no loss in performance after applying pruning inference, and the FPS improves to 63, which makes it possible to integrate the model proposed in this paper into mobile terminals and perform real-time operations locally.

4.3. Module Analysis

The main innovation of the network model proposed in this paper can be summarized in the following two points:

1. Branches added for the overfitting phenomenon of complex network structures. As the neural network is compounded after the improvement hybridization, the overfitting ability of the network will increase. In order to reduce the possibility of overfitting, the model incorporates the generative module. Through this module, a result from the confrontation is obtained. Since features of the highest dimension are extracted and simulated, this result is combined with the upper part of the determination model. The results generated are computed together with the loss calculation, thus improving the robustness of the whole detection network.

2. Add the pruning inference to the network. In general, the higher the detection network performance, the better. However, the performance improvement is often accompanied by a considerable time cost. Moreover, the depth of the neural network deeper does not necessarily lead to better results. Owing to overfitting, the partitioning results of deeper networks may even be inferior to those of lower layers. Therefore, the pruning of the model is judged at the time of training in terms of the given conditions. We also zero the input of the generative module branch to achieve structural deactivation, which significantly improves the training speed and even reduces the overfitting of the neural network to achieve the “one model polymorphism”.

4.4. Smart Apple Flower Detection System

In order to make the model proposed in this paper available for practical application, we will implement the model packaging and build a user-visualizable interface. We developed it in C#, and the main functional modules of the software are: 1. Batch import images and label apple flowers. 2. Count imported images. 3. Generate CSV format record files for data backup.

5. Conclusions

This paper proposed a generative module to optimize the mainstream detection network and gained excellent results in detecting apple flowers with the precision reaching 90.01%, recall reaching 98.79%, and mAP reaching 97.43% on GM-EfficientDet-D5. Compared with the original network, these three indexes are improved by 3.98%, 7.46%, and 7.52%. Since the backbone of these network models was unstable, non-convergent, and overfitting when the dataset was insufficient, multiple image pre-processing methods were applied to augment the dataset, such as Mixup, Cutout, CutMix, SnapMix, and Mosaic.

In order to verify the effectiveness of the proposed method, this paper applied the generative module to multiple mainstream detection networks; the results indicated that the performances of networks adding the generative module have all been improved. Afterward, we discussed the effect of different implementations of the generative module on the results. We found that the model using the CVAE-GAN implementation has the best performance but the lowest inference speed. Therefore, we applied the pruning inference algorithm proposed in this paper to the model and found that pruning inference can improve the inference speed of YOLO-v5 to 76 PFS with almost no impact on the model performance, which can meet the demand of real-time display.

Finally, the proposed model is encapsulated and visualized for application development so that the proposed model and algorithm can be applied in practical scenarios.

Author Contributions: Conceptualization, Y.Z.; methodology, Y.Z.; validation, Y.Z.; writing—original draft preparation, Y.Z.; writing—review and editing, Y.Z., S.W., S.H. and Z.Z.; visualization, Z.Z. and S.H.; supervision, Y.Z.; project administration, Y.Z.; funding acquisition, Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Modern Agricultural Industrial Technology System of China (No. CARS-28-20).

Acknowledgments: We are grateful to the Edison Coding Club of CIEE in China Agricultural University for their strong support during our thesis writing.

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

References

1. Wu, D.; Lv, S.; Jiang, M.; Song, H. Using channel pruning-based YOLO v4 deep learning algorithm for the real-time and accurate detection of apple flowers in natural environments. *Comput. Electron. Agric.* **2020**, *178*, 105742. [[CrossRef](#)]
2. Dias, P.A.; Tabb, A.; Medeiros, H. Apple flower detection using deep convolutional networks. *Comput. Ind.* **2018**, *99*, 17–28. [[CrossRef](#)]
3. Pathan, M.; Patel, N.; Yagnik, H.; Shah, M. Artificial cognition for applications in smart agriculture: A comprehensive review. *Artif. Intell. Agric.* **2020**, *4*, 81–95. [[CrossRef](#)]

4. Weng, S.; Zhu, W.; Zhang, X.; Yuan, H.; Zheng, L.; Zhao, J.; Huang, L.; Han, P. Recent advances in Raman technology with applications in agriculture, food and biosystems: A review. *Artif. Intell. Agric.* **2019**, *3*, 1–10. [\[CrossRef\]](#)
5. Zhang, W.; Hu, J.; Zhou, G.; He, M. Detection of Apple Defects Based on the FCM-NPGA and a Multivariate Image Analysis. *IEEE Access* **2020**, *8*, 38833–38845. [\[CrossRef\]](#)
6. Guo, Z.; Wang, M.; Agyekum, A.A.; Wu, J.; Chen, Q.; Zuo, M.; El-Seedi, H.R.; Tao, F.; Shi, J.; Ouyang, Q.; et al. Quantitative detection of apple watercore and soluble solids content by near infrared transmittance spectroscopy. *J. Food Eng.* **2020**, *279*, 109955. [\[CrossRef\]](#)
7. Jiang, B.; He, J.; Yang, S.; Fu, H.; Li, T.; Song, H.; He, D. Fusion of machine vision technology and AlexNet-CNNs deep learning network for the detection of postharvest apple pesticide residues. *Artif. Intell. Agric.* **2019**, *1*, 1–8. [\[CrossRef\]](#)
8. Abbas, H.M.T.; Shakoar, U.; Khan, M.J.; Ahmed, M.; Khurshid, K. Automated Sorting and Grading of Agricultural Products based on Image Processing. In Proceedings of the 2019 8th International Conference on Information and Communication Technologies (ICICT), Karachi, Pakistan, 16–17 November 2019; pp. 78–81. [\[CrossRef\]](#)
9. Sun, S.; Jiang, M.; He, D.; Long, Y.; Song, H. Recognition of green apples in an orchard environment by combining the GrabCut model and Ncut algorithm. *Biosyst. Eng.* **2019**, *187*, 201–213. [\[CrossRef\]](#)
10. Mazzia, V.; Khaliq, A.; Salvetti, F.; Chiaberge, M. Real-Time Apple Detection System Using Embedded Systems With Hardware Accelerators: An Edge AI Application. *IEEE Access* **2020**, *8*, 9102–9114. [\[CrossRef\]](#)
11. Islam, N.; Rashid, M.M.; Wibowo, S.; Xu, C.Y.; Morshed, A.; Wasimi, S.A.; Moore, S.; Rahman, S.M. Early Weed Detection Using Image Processing and Machine Learning Techniques in an Australian Chilli Farm. *Agriculture* **2021**, *11*, 387. [\[CrossRef\]](#)
12. Xu, J.; Guga, S.; Rong, G.; Riao, D.; Liu, X.; Li, K.; Zhang, J. Estimation of Frost Hazard for Tea Tree in Zhejiang Province Based on Machine Learning. *Agriculture* **2021**, *11*, 607. [\[CrossRef\]](#)
13. Lim, J.; Ahn, H.S.; Nejati, M.; Bell, J.; Williams, H.; MacDonald, B.A. Deep Neural Network Based Real-time Kiwi Fruit Flower Detection in an Orchard Environment. *arXiv* **2020**, arXiv:2006.04343.
14. Afonso, M.; Mencarelli, A.; Polder, G.; Wehrens, R.; Lensink, D.; Faber, N. Detection of Tomato Flowers from Greenhouse Images Using Colorspace Transformations. In *Progress in Artificial Intelligence*; Moura Oliveira, P., Novais, P., Reis, L.P., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 146–155.
15. Tian, M.; Chen, H.; Wang, Q. Detection and Recognition of Flower Image Based on SSD network in Video Stream. *J. Phys. Conf. Ser.* **2019**, *1237*, 032045. [\[CrossRef\]](#)
16. Biradar, B.V.; Shrikhande, S.P. Flower detection and counting using morphological and segmentation technique. *Int. J. Comput. Sci. Inform. Technol.* **2015**, *6*, 2498–2501.
17. Sun, K.; Wang, X.; Liu, S.; Liu, C. Apple, peach, and pear flower detection using semantic segmentation network and shape constraint level set. *Comput. Electron. Agric.* **2021**, *185*, 106150. [\[CrossRef\]](#)
18. Farjon, G.; Krikeb, O.; Hillel, A.B.; Alchanatis, V. Detection and counting of flowers on apple trees for better chemical thinning decisions. *Precis. Agric.* **2020**, *21*, 503–521. [\[CrossRef\]](#)
19. Krizhevsky, A.; Sutskever, I.; Hinton, G. ImageNet Classification with Deep Convolutional Neural Networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105. [\[CrossRef\]](#)
20. Zhang, H.; Cisse, M.; Dauphin, Y.N.; Lopez-Paz, D. mixup: Beyond empirical risk minimization. *arXiv* **2017**, arXiv:1710.09412.
21. DeVries, T.; Taylor, G.W. Improved regularization of convolutional neural networks with cutout. *arXiv* **2017**, arXiv:1708.04552.
22. Yun, S.; Han, D.; Oh, S.J.; Chun, S.; Choe, J.; Yoo, Y. Cutmix: Regularization strategy to train strong classifiers with localizable features. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27–28 October 2019; pp. 6023–6032.
23. Huang, S.; Wang, X.; Tao, D. SnapMix: Semantically Proportional Mixing for Augmenting Fine-grained Data. *arXiv* **2020**, arXiv:2012.04846.
24. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolo4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
25. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125.
26. Kingma, D.P.; Welling, M. Auto-encoding variational bayes. *arXiv* **2013**, arXiv:1312.6114.
27. Du, L.; Ding, X.; Liu, T.; Li, Z. Modeling event background for if-then commonsense reasoning using context-aware variational autoencoder. *arXiv* **2019**, arXiv:1909.08824.
28. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Networks. *Adv. Neural Inf. Process. Syst.* **2014**, *3*, 2672–2680. [\[CrossRef\]](#)
29. Mirza, M.; Osindero, S. Conditional generative adversarial nets. *arXiv* **2014**, arXiv:1411.1784.
30. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Munich, Germany, 5–9 October 2015; Springer: Berlin/Heidelberg, Germany, 2015; pp. 234–241.
31. Zheng, Z.; Wang, P.; Liu, W.; Li, J.; Ye, R.; Ren, D. Distance-IoU loss: Faster and better learning for bounding box regression. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 12993–13000.
32. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.

-
33. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
 34. Jocher, G. yolov5. 2020. Available online: <https://github.com/ultralytics/yolov5> (accessed on 26 October 2020).
 35. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 21–37.
 36. Li, Z.; Zhou, F. FSSD: Feature fusion single shot multibox detector. *arXiv* **2017**, arXiv:1712.00960.
 37. Zhang, S.; Wen, L.; Bian, X.; Lei, Z.; Li, S.Z. Single-shot refinement neural network for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4203–4212.
 38. Tan, M.; Pang, R.; Le, Q.V. Efficientdet: Scalable and efficient object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 10781–10790.
 39. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 91–99. [[CrossRef](#)]
 40. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2961–2969.