



# Light-YOLOv3: fast method for detecting green mangoes in complex scenes using picking robots

Zhi-Feng Xu<sup>1</sup> · Rui-Sheng Jia<sup>1,2</sup> · Hong-Mei Sun<sup>1,2</sup> · Qing-Ming Liu<sup>1</sup> · Zhe Cui<sup>1</sup>

Published online: 30 July 2020

© Springer Science+Business Media, LLC, part of Springer Nature 2020

## Abstract

When a robot picks green fruit under natural light, the color of the fruit is similar to the background; uneven lighting and fruit and leaf occlusion often affect the performance of the detection method. We take green mangoes as an experimental object. A lightweight green mangoes detection method based on YOLOv3 is proposed here. To improve the detection speed of the method, we first combine the color, texture, and shape features of green mango to design a lightweight network unit to replace the residual units in YOLOv3. Second, the improved Multiscale context aggregation (MSCA) module is used to concatenate multilayer features and make predictions, solving the problem of insufficient position information and semantic information on the prediction feature map in YOLOv3; this approach effectively improves the detection effect for the green mangoes. To address the overlap of green mangoes, soft non-maximum suppression (Soft-NMS) is used to replace non-maximum suppression (NMS), thereby reducing the missing of predicted boxes due to green mango overlaps. Finally, an auxiliary inspection green mango image enhancement algorithm (CLAHE-Mango) is proposed, is suitable for low-brightness detection environments and improves the accuracy of the green mango detection method. The experimental results show that the F1% of Light-YOLOv3 in the test set is 97.7%. To verify the performance of Light-YOLOv3 under the embedded platform, we embed one-stage methods into the Adreno 640 and Mali-G76 platforms. Compared with YOLOv3, the F1% of Light-YOLOv3 is increased by 4.5%, and the running speed is increased by 5 times, which can meet the real-time running requirements for picking robots. Through three sets of comparative experiments, we could determine that our method has the best detection results in terms of dense, backlit, direct light, night, long distance, and special angle scenes under complex lighting.

**Keywords** Green mangoes · Picking robots · YOLOv3 · Convolutional neural networks (CNNs)

## 1 Introduction

The application of machine vision to the early yield estimation of fruit trees and object recognition performed by agricultural robots have become a popular research area in recent years [1]. Fruit-and-vegetable-picking robots realize picking mechanization and automation, reduces labor and time costs, and greatly improve the picking efficiency. The target of early

yield estimation is usually applied to immature green fruits. The picking objects of picking robots are also usually green mangoes, green apples, fragrant pears, and other green fruits. Accurate identification and positioning of the fruits are the key for robots to pick fruits. Therefore, it is very important to study efficient detection methods for green fruits.

The fruit images collected in the natural environment have many problems, such as uneven illumination, shadows, reflection of leaves, occlusion of branches and leaves and mutual occlusion between fruits [2]. The green fruit itself is green, which is similar to the color of leaves and weeds, and it cannot be distinguished by color characteristics alone [3]. The detection of green fruits based on machine vision is a major challenge. Even with visual inspection by humans, it is sometimes difficult to clearly distinguish the green fruits hidden in a green scene. In fact, the surface feature information of green fruits is always different from the green branches and leaves. Computers equipped with machine vision can often detect this nuance better human eyes. At present, domestic and foreign

✉ Rui-Sheng Jia  
jrs716@163.com

✉ Hong-Mei Sun  
shm0221@163.com

<sup>1</sup> College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China

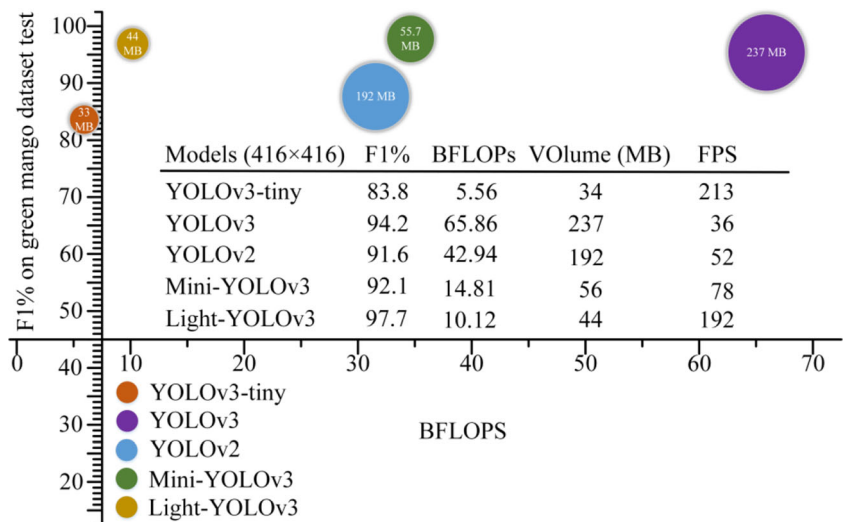
<sup>2</sup> Shandong Province Key Laboratory of Wisdom Mine Information Technology, Shandong University of Science and Technology, Qingdao 266590, China

related literature has proposed a variety of solutions to the above problems. Rajneesh B et al. [4] proposed a detection method based on fast Fourier transform (FFT) leakage to identify green citrus images collected under natural lighting conditions, with a recognition accuracy rate of 82.2%. Lu J et al. [5] proposed to use the contour features of the surface of the fruit with a circular light distribution, combined with the Hough transform for circle fitting. The recall rate of the algorithm in 20 citrus orchard scene images reached 81.2%. After using local binary pattern (LBP) texture features, the accuracy of the test on 25 images could reach 82.3%, but the image acquisition process requires manual configuration of the light source. Stein M et al. [6] used a multiview method to solve the mango occlusion problem, and they correlated and tracked the fruit through multiple image sequences from multiple viewpoints; however, this method required complex auxiliary equipment and had poor real-time performance. From the perspective of deep learning, in the images of green fruits, the pixel values of the fruit and the background are very similar. The boundary gradient between the fruit and the background is too small, which makes the network model insensitive to green fruits when extracting features. Thus, in the detection stage, the network could not distinguish the boundary between the green fruit and the background very well. This limitation has led to difficulties in detecting and positioning green mangoes. We take green mangoes as the experimental object here, and more importantly, we explore the detection method for green fruits. The color of mature green mangoes is highly similar to that of leaves. The shape is not a standard circle or ellipse, and even the fruits are covered by leaves and branches or are overlapping with each other. These factors have brought great difficulties to the detection of green mangoes. The convolutional neural network (CNN) is a deep feedforward neural network with characteristics of local connection and weight sharing. The CNN is the cornerstone of breakthrough achievements in the field of computer vision in recent years and is widely used in natural language processing, recommendation systems, and speech recognition. Compared with classical methods, deep convolutional neural networks (DCNN) [7] have shown great advantages in the field of target detection in recent years [8]. Because of its high-level extraction of high-dimensional features, the DCNN makes it possible to accurately identify green mangoes in complex situations. It is mainly divided into two methods; one is two-stage object detection, which is a detection method that is based on region generation; the core idea is to first generate region proposal, and then to classify them. Representative methods are RCNN [9], Fast RCNN [10], Faster RCNN [11], RFCN [12], and Mask R-CNN [13]. Sa et al. [14] used RGB and NIR image information to train the multimodal Faster RCNN model, and they detected mangoes using the early and late fusion methods, respectively. However, this method cannot detect the objects well in the case of large areas of occlusion.

Because the region suggestion step consumes a large amount of computing resources and the detection time is long, it cannot meet real-time requirements. Another method is one-stage object detection, which is based on regression. The core idea is to use a single CNN to process the entire image directly to achieve object detection and predict the object's category. Its speed is usually faster than the two-stage method, for which the representative methods are SSD [15], YOLO [16], YOLOv2 [17], YOLOv3 [18] and RetinaNet [19]. Due to the limited memory and computing resources of embedded devices, even the one-stage method is difficult to apply to embedded devices. At the beginning of the thesis, we studied the current more advanced object detection methods. Although the lightweight algorithm that can be applied to the embedded side is sufficiently fast, the detection accuracy is too poor to meet the actual picking of the robot in the orchard. In the current one-stage methods, the lightweight YOLOv3-tiny can be applied to embedded platforms. However, its shallow network layer results in poor feature extraction ability and low detection accuracy. YOLOv3 has shown excellent performance because of the design of darknet53 and the predictive network. However, the amount of computation and the model size of YOLOv3 exceed the affordability of the embedded side. Thus, we believe that improvement based on YOLOv3 is the best choice.

To realize the detection and positioning of green mangoes by picking robots in a complex environment, a method based on deep learning is proposed. First, we design a lightweight unit to replace the residual units in YOLOv3 based on the features of green mangoes. While ensuring high precision, this approach greatly reduces the amount of computation needed and achieves a good balance between green mango detection and real-time performance. Second, in order to overcome problems such as fruit and leaf occlusion, the MSCA [20] is used to perform feature fusion on feature maps of three scales to further correlate the semantic and detailed information of the object from shallow to deep. The above operation enables our method to detect green mangoes at different distances and is suitable for various types of picking equipment. To address the overlapping of green mangoes, Soft-NMS [21] is used to replace NMS [22], thereby reducing the missing of predicted boxes due to green mangoes overlapping. Finally, to help solve the problem of green mango detection under backlighting, darkness and other lighting conditions, the CLAHE-Mango for the auxiliary detection of green mangoes is proposed to improve the accuracy of the green mango detection algorithm. We have transplanted this method to two advanced embedded platforms, Adreno 640 and Mali-G76, and designed three sets of comparative experiments to prove that the method is suitable for green mango detection in various complex scenarios. Our method provides new ideas for green fruit detection based on machine vision (Fig. 1).

**Fig. 1** The performance of YOLO series detection methods on the green mango dataset



## 2 Related work

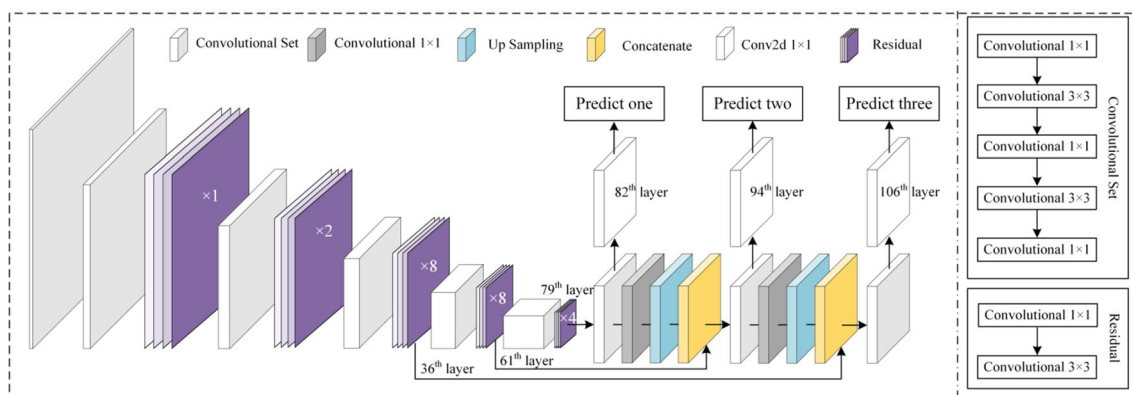
### 2.1 YOLOv3

YOLO is an end-to-end object detection method that is based on deep learning. It has the advantages of fast running speed and real-time effects in the GPU environment. YOLOv3 evolved from YOLO and YOLOv2. It uses darknet-53 for feature extraction and multiscale training. Compared with YOLO and YOLOv2, YOLOv3 improves the detection accuracy and increases the model complexity. The parameters of YOLOv3-416 reach 65.86 Bn, its model size is approximately 237 MB, and its detection speed is 36 FPS on GTX1080 graphics. The YOLOv3 network is shown in Fig. 2.

As shown in Fig. 2, YOLOv3 is a fully convolutional network whose backbone is darknet-53. Darknet-53 contains 4 types of residual units, each of which consists of a series of  $1 \times 1$  and  $3 \times 3$  convolutional layers (each convolutional layer contains a BN layer and a Leaky ReLU layer). There is no pooling layer in the YOLOv3 network. During its forward propagation, the tensor size is changed by changing the step

size of the convolution kernel (stride = 2), which will be up reduced at most 5 times (feature maps reduced to 1/25 of the original input size, or 1/32). YOLOv3 uses three feature maps of different scales for object detection. It can be seen from Fig. 2 that after the 79th layer of the convolutional network, after a Convolutional Set, standard  $3 \times 3$  convolution and Conv2d  $1 \times 1$ , the detection result of the first scale ( $13 \times 13$ ) is obtained. To achieve detection of fine-grained features, the feature maps of the 79st layer are upsampled, and then, they are concatenated with the 61st layer feature maps to obtain the 91st finer-grained feature maps, and next, after a few convolutional layers, obtain the second scale feature map ( $26 \times 26$ ). It has a medium-scale receptive field and is suitable for detecting medium-scale objects; The 91th layer feature map is upsampled and concatenated with the 36th layer feature map to obtain the third-scale feature map ( $52 \times 52$ ). It has the smallest receptive field and is suitable for detecting small-sized objects.

The original intention of designing a residual unit is to solve the problem of gradient disappearance and gradient explosion [23]. Therefore, many excellent networks currently use residual units to solve the gradient problem, which further



**Fig. 2** YOLOv3 network

improves the performance of the network. However, the standard  $3 \times 3$  convolution contained in the residual unit still has a relatively large amount of computation. YOLOv3 uses 21 sets of residual units (a total of 46 layers) and the prediction network uses many Convolutional Set combinations, which contain a large number of  $3 \times 3$  standard convolutions, resulting in larger FLOPs and a larger model for the entire network. For this reason, why YOLOv3 has better accuracy but cannot be applied to embedded platforms.

### 3 Green mangoes detection framework

#### 3.1 Green mangoes detection network

At present, the high computing resources of advanced object detection methods exceed the capabilities of embedded or mobile devices [8]. For this reason, we have designed Light-YOLOv3 with the premise resource constraints, thus ensuring higher accuracy and speed. Based on the YOLOv3 network, Light-YOLOv3 improves the backbone and prediction network. At the same time, Light-YOLOv3 retains the advantages of YOLOv3, such as multiscale prediction, anchor mechanism and so on. The network of Light-YOLOv3 is shown in Fig. 3.

In Fig. 3, Light-YOLOv3’s network contains two parts: the backbone and the prediction network. The backbone uses two convolutions on the input image to obtain high-dimensional features, and then, it uses 5 groups (a total of 14) of LightNet v1 to build a deep feature network. This structure has very strong feature extraction ability and very low computation. The prediction network is input into a multiscale context aggregation (MSCA) module from three different levels of features. The MSCA module can realize the multiplexing and fusion of shallow to deep features, and it can solve the problem of fruit and leaf occlusion and achieve long-range target detection. We will introduce the backbone and prediction network of Light-YOLOv3 in detail.

#### 3.1.1 Backbone network

To improve the efficiency of the model and simplify the model size, we redesigned the network unit. In this process, this paper refers to the design ideas of lightweight network units such as deep separable convolutions [24], ShuffleNet v1 [25], and ShuffleNet v2 [26]. Combined with the characteristics of green mangoes, a lightweight network unit LightNet v1 was designed. The network units in Fig. 4 are all excellent lightweight network elements, whose design goal is to achieve the best model accuracy by using limited computing resources.

Figure 4 (a) is a residual that is module based on depth separable convolution. In its main branch, it firstly uses a  $1 \times 1$  point convolution (PWConv) for dimension upgrading; second, it uses the more efficient depth separable convolution (DWConv) instead of the standard  $3 \times 3$  convolution; and finally, it uses  $1 \times 1$  points Convolution (PWConv) dimensionality reduction. A BN layer and ReLU6 are added after the first two layers. DWConv uses different convolution kernels for each input channel, and its effect is similar to that of a standard convolution. The ratio of the depth separable convolution to the standard convolution FLOPs is

$$\frac{K_h \times K_w \times C_{in} \times W \times H + C_{in} \times C_{out} \times W \times H}{K_h \times K_w \times C_{in} \times C_{out} \times W \times H} = \frac{1}{C_{out}} + \frac{1}{K_{h,w}^2} \tag{1}$$

In formula (1),  $K_H \times K_W$  is the convolution kernel size,  $C_{in}$  is the number of input channels, and  $C_{out}$  is the number of output channels.  $W$  and  $H$  are the width and height of the output feature map respectively. From formula (1), when the convolution kernel size is  $3 \times 3$ , the FLOPs of the deep separable convolution is approximately 1/9 of the standard convolution. Therefore, many lightweight networks use deep separable convolution instead of standard convolution.

Figure 4 (b) is a ShuffleNet v1 network unit. The core of ShuffleNet v1 is the use of two operations: pointwise group convolution [27] and channel shuffle [25]. In lightweight networks, PWConv causes a limited number of channels to be

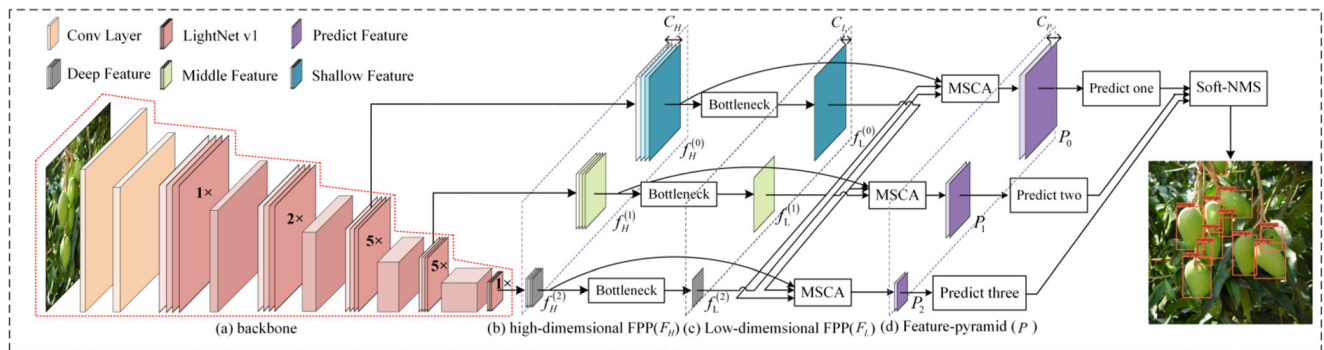


Fig. 3 The network of Light-YOLOv3

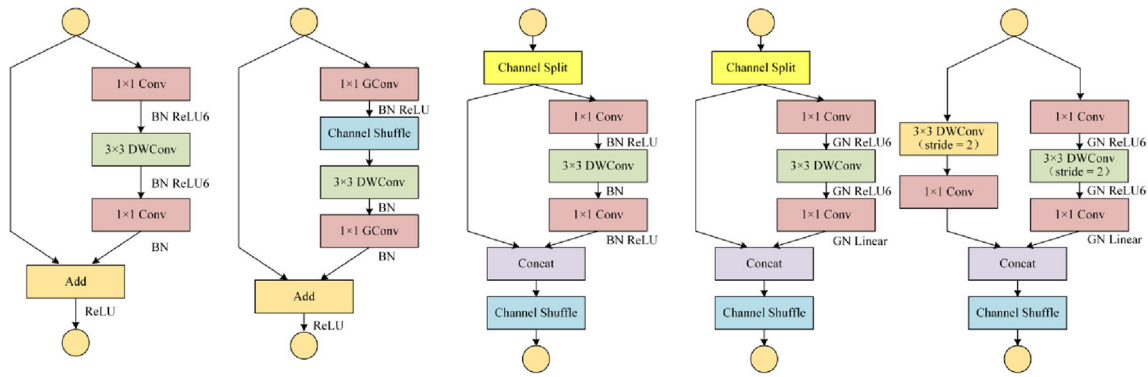


Fig. 4 Some excellent lightweight network units at present

filled with constraints, which will clearly lose the prediction accuracy. However, multiple groups of convolutions stacked together will have a side effect: the output of a certain channel is derived from only a small part of the input channel, as shown in Fig. 5 (a). Such attributes reduce the flow of information between the channel groups and reduce the ability to express information. To solve the above problems, as shown in Fig. 5 (b), the author proposes channel shuffling: rearrange the channels in such a way that the channels can communicate their information with one another.

For packet convolution, when the number of packets increases, the FLOPs can be reduced, but the memory access cost (MAC) can also be increased. The FLOPs and MAC formula of the group convolution are shown in formulas (2) and (3):

$$\text{Group convolution FLOPs} = B = \frac{hwc_1c_2}{g} \tag{2}$$

$$\text{Group convolution MAC} = hwc_1 + \frac{Bg}{c_1} + \frac{B}{hw} \tag{3}$$

In formulas (2) and (3),  $w$  and  $h$  are the width and height of the feature map, and  $c_1$  and  $c_2$  are the number of input

channels and output channels, respectively. Here,  $g$  is the number of groups. It can be seen that when  $g$  increases, the MAC will increase at the same time. Therefore, it is unwise to use the pointwise group convolution in large numbers.

Figure 4 (c) is a ShuffleNet v2 network unit. Compared with the structure of Fig. 4 (b), channel split [26] is added. In the channel split operation, the input feature map is divided into two branches in the channel dimension at the beginning: the number of channels is  $c'$  and  $c - c'$ , respectively, and  $c' = \frac{c}{2}$  is actually implemented. In Fig. 4 (c), the left branch in the structure is equivalently mapped, and the right branch contains 3 consecutive convolutions, while the input and output channels are the same. In the first and third layers, we replace the pointwise group convolution in the structure of Fig. 4 (b) with a  $1 \times 1$  point convolution. Since the two branches are divided into two groups, the output of the two branches is no longer an add operation but a concatenation operation. Finally, we channel shuffle the concatenation result to ensure the exchange of information between the two branches. Its authors found that ShuffleNet v2 is the best in terms of speed and accuracy under the same conditions on the classification task using ImageNet.

Fig. 5 Channel shuffle with two stacked group convolutions. GConv stands for group convolution. a) two stacked convolution layers with the same number of groups. Each output channel relates to only the input channels within the group. No cross talk; b) an equivalent implementation to a) using channel shuffle

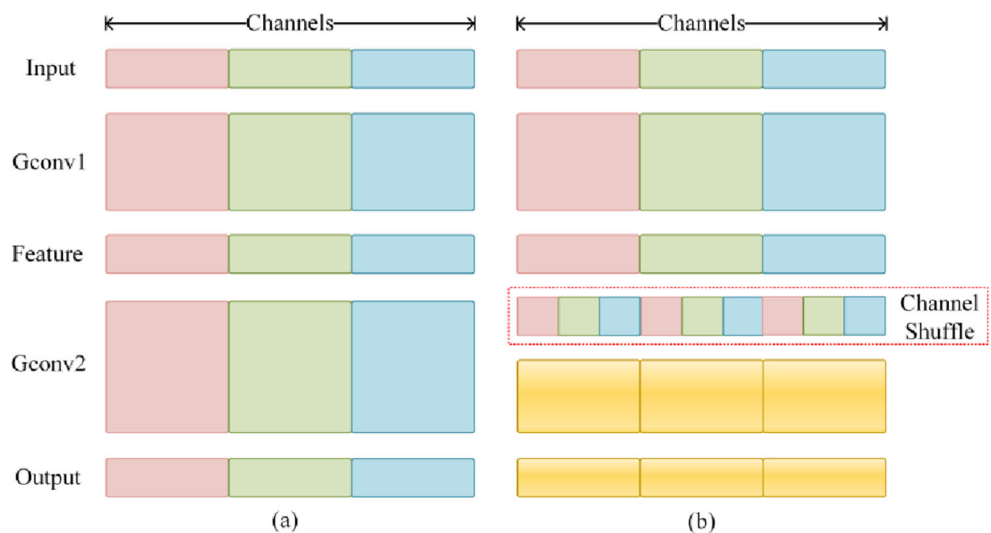


Figure 4 (d) is a lightweight network unit that we designed called LightNet v1. There are two main improvements to Fig. 4 (c): First, we replace all Batch Normalization (BN) layers [28] with Group Normalization (GN) layers [29]. The reason is that in the model training stage, the size of the batch size often affects the final training results. GN is an improved algorithm for a higher error rate of BN when the batch size is small, because the calculation result of the BN layer depends on the current batch data. When the batch size is small (such as 2, 4), the average sum of the batch data variance is less representative, and therefore, it has a greater impact on the final result. GN is basically not affected by the batch size. Second, we replaced the original third-layer ReLU activation function with a linear activation function. Because the output dimension of the DWConv layer is low (for low-dimensional space, linear mapping will save features, while nonlinear mapping will destroy features), continued application of ReLU will bring about the problem of information loss. Therefore, the third layer uses Linear instead of ReLU to ensure more complete output information. The structure of Fig. 4 (e) is the downsampling module designed according to Fig. 4 (d). We get rid of the previous channel split, and we copy one input for each branch. Each branch is a lower sample with a band = 2. Finally, we do concatenate and channel shuffle to ensure information exchange between the two branches. After the above operations, the space of the feature map is halved, and the number of channels is doubled. Finally, we redesigned the backbone using the structure in Fig. 4 (d) and (e). The details of the backbone are shown in Table 1.

The new backbone is equipped with LightNet v1, compared to DarkNet-53 in YOLOv3, and it reduces the amount of computation by 45.38 BN and contributes 1.3 to the F1% of the final model, and the FPS increases by 107.

### 3.1.2 Prediction network

We believe that for the green mangoes detection tasks, effective features should need to have three characteristics: 1. The feature maps contain detailed information of the green mangoes. 2. Features are extracted through a sufficiently deep network. 3. The features should need to contain the semantic information of the green mangoes. Based on the above characteristics, on the premise of ensuring a low amount of computation, the MSCA module is used as the main component of the prediction network. The schematic diagram of the MSCA module is shown in Fig. 6.

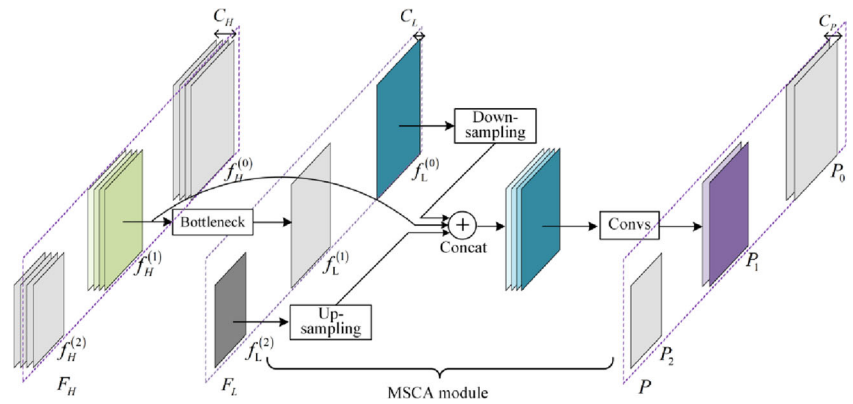
Figure 6 shows the calculation process of the prediction result for  $P_1$ . The feature maps input into the prediction network by the backbone are combined into  $F_H$  and contain three scale feature maps ( $13 \times 13$ ,  $26 \times 26$ ,  $52 \times 52$ ).  $C_H$  is the

**Table 1** The details of the backbone network

	Type	Filter	Channel	Stride	Featuremap
	Convolutional	$3 \times 3$	32	1	$416 \times 416$
	Convolutional	$3 \times 3$	16	2	$208 \times 208$
	Conv PW	$1 \times 1$	64	1	
1	Conv DW	$3 \times 3$	64	1	
	Conv PW	$1 \times 1$	16	1	$208 \times 208$
	Conv DW	$3 \times 3$	32	2	$104 \times 104$
	Conv PW	$3 \times 3$	32	1	$104 \times 104$
	Conv PW	$1 \times 1$	128	1	
2	Conv DW	$3 \times 3$	128	1	
	Conv PW	$1 \times 1$	32	1	$104 \times 104$
	Conv DW	$3 \times 3$	64	2	$52 \times 52$
	Conv PW	$3 \times 3$	64	1	$52 \times 52$
	Conv PW	$1 \times 1$	256	1	
5	Conv DW	$3 \times 3$	256	1	
	Conv PW	$1 \times 1$	64	1	$52 \times 52$
	Conv DW	$3 \times 3$	128	2	$26 \times 26$
	Conv PW	$3 \times 3$	128	1	$26 \times 26$
	Conv PW	$1 \times 1$	512	1	
5	Conv DW	$3 \times 3$	512	1	
	Conv PW	$1 \times 1$	128	1	$26 \times 26$
	Conv DW	$3 \times 3$	256	2	$13 \times 13$
	Conv PW	$3 \times 3$	256	1	$13 \times 13$
	Conv PW	$1 \times 1$	1024	1	
1	Conv DW	$3 \times 3$	1024	1	
	Conv PW	$1 \times 1$	256	1	$13 \times 13$
	Conv PW	$1 \times 1$	1024	1	$13 \times 13$

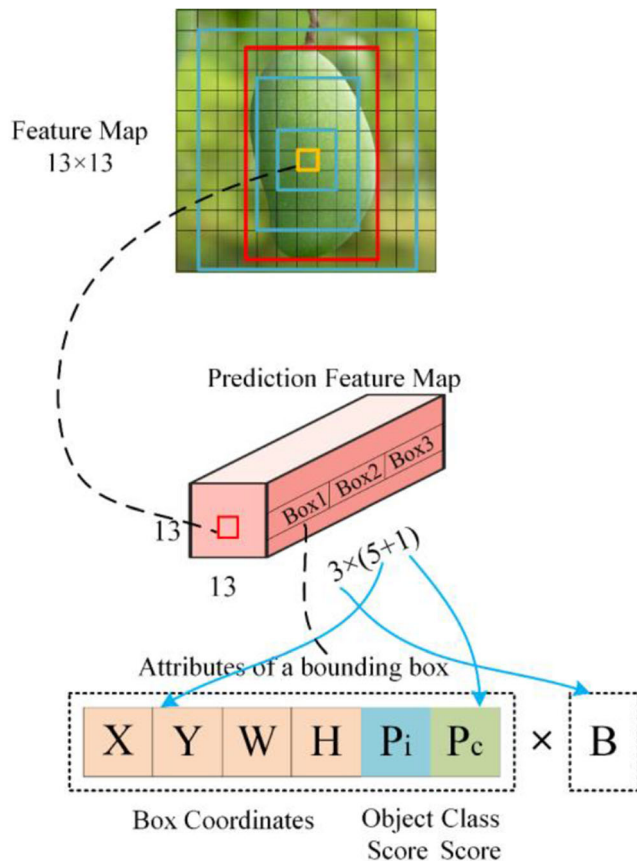
number of channels of the feature map  $f_H^{(1)}$  ( $26 \times 26$ ). First,  $f_H^{(1)}$  is passed through a bottleneck operation to obtain the  $f_H^{(1)}$  of the  $C_L$  channel number. In general,  $C_L = \frac{C_H}{2}$ . Second, we perform the concatenate operation on  $S^2$  after the bottleneck and  $1_i^{obj}$  after downsampling and  $1_i^{obj}$  after upsampling. From the above, the MSCA module can concatenate feature maps from the three scales of the backbone. Compared with the two-scale fusion of YOLOv3, the MSCA module contains more location information and semantic information on the green mangoes. Because MSCA uses LightNet v1 for feature extraction, the calculation of MSCA is much lower than that of YOLOv3's prediction network. Finally, the concatenated feature maps are subjected to a series of LightNet v1 operations to obtain the prediction result  $P_1$ . The calculation process of  $P_0$  and  $P_2$  is similar to that of  $P_1$ . Compared with the prediction network in YOLOv3, the new MSCA equipped with LightNet v1 reduces the amount of computation by 10.65 BN and contributes 1.2% to F1% of the final model, and the FPS increases by 55.

**Fig. 6** Multiscale context aggregation (MSCA) module



Taking the prediction output of  $P_1$  ( $13 \times 13 \times 18$ ) as an example, the prediction information contained in  $P_1$  is shown in Fig. 7.

We used k-means clustering to calculate 9 anchors of the green mango dataset: ( $14 \times 27$ ), ( $23 \times 36$ ), ( $40 \times 52$ ), ( $89 \times 151$ ), ( $91 \times 174$ ), ( $113 \times 198$ ), ( $157 \times 243$ ), ( $220 \times 284$ ), and ( $327 \times 389$ ). K-means uses the Euclidean distance. The ( $13 \times 13$ ), ( $26 \times 26$ ), and ( $52 \times 52$ ) size feature maps use three anchors in turn, and each grid predicts three boxes. For the above three scales of feature maps, the parameters of each box include  $x, y, w, h$ , the category probability  $P_i$  of a green mango and the confidence  $P_c$ .



**Fig. 7** Prediction for a  $13 \times 13$  feature map

### 3.1.3 Filter of bounding boxes

Our prediction network generates multiple prediction boxes. Therefore, a non-maximum suppression (NMS) algorithm is used to suppress those redundant prediction boxes. Suppression is an iterative-traversal-elimination process. The NMS algorithm is shown in Table 2. The process can be best described in one sentence: First, it sorts all detection boxes on the basis of their scores. It selects the detection box  $M$  with the highest score and suppresses all of the other detection boxes whose IOU calculated with  $M$  is greater than a certain threshold. This process is recursively applied on the remaining boxes.

It can be seen that using the NMS algorithm makes it slightly difficult to detect green mangoes when there are dense, occlusions of green mangoes and leaves, and so on. If the IOU of  $b_i$  and  $M$  is greater than a certain threshold, the NMS algorithm resets the score of  $b_i$  to zero and deletes  $b_i$ . This operation could cause the prediction boxes of the overlapping fruits to be deleted by NMS, which affects the accuracy of the model. To solve the above problems, we use the Soft-NMS algorithm to replace the NMS algorithm in YOLOv3. Soft-NMS has a modification to the traditional greedy NMS algorithm in which we overlap instead of setting the score to zero as in NMS. The algorithm is shown in Table 3:

In Table 3,  $B$  is the list of initial detection boxes;  $S$  contains the corresponding detection scores, and the function  $f(IOU(M, b_i))$  is defined as follows:

$$f(IOU(M, b_i)) = \begin{cases} 1, & IOU(M, b_i) < N_t \\ e^{-\frac{IOU(M, b_i)^2}{\sigma}}, & IOU(M, b_i) \geq N_t \end{cases} \quad (4)$$

where  $b_i$  is the box number;  $M$  is the highest score,  $N_t$  is the set threshold and  $\sigma$  is a hyper parameter. It can be seen that the Soft-NMS algorithm adds a penalty function compared to the NMS algorithm. Intuitively, if the IOU calculated by a prediction box and  $M$  exceeds a certain threshold, the prediction box will not be deleted, but its score will be reduced accordingly. The following experiments that the F1% of the model using soft-NMS is 0.4 higher than the performance of NMS.

**Table 2** Non-maximum suppression

Algorithm 1: NMS

**Input:**  $B = \{b_1, \dots, b_N\}$ ,  $S = \{s_1, \dots, s_N\}$

1.  $D \leftarrow \{\}$
2. **while**  $B \neq \text{empty}$  **do**
3.      $m \leftarrow \text{argmax } S$
4.      $M \leftarrow b_m$
5.      $D \leftarrow D \cup M$ ;  $B \leftarrow B - M$
6.     **for**  $b_i$  in  $B$  **do**
7.         **if**  $\text{iou}(M, b_i) \geq N_i$  **then**
8.              $B \leftarrow B - b_i$ ;  $S \leftarrow S - S_i$
9.     **end**
10.    **end**
11. **end**
12. **return**  $D, S$

### 3.2 Loss function design

The loss function of the training network for green mango detection consists of four parts: the first part is about the prediction of the green mango center coordinates, as shown in eq. (5); the second part is the prediction about the green mango boundary box, as shown in eq. (6); the third part is about the prediction of the confidence in green mango, as shown in eq. (7); and the fourth part is the prediction of the category of green mango, as shown in eq. (8).

$$Loss_1 = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B \ell_{ij}^{obj} \left[ (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \tag{5}$$

**Table 3** Soft-NMS

Algorithm 2: Soft-NMS

**Input:**  $B = \{b_1, \dots, b_N\}$ ,  $S = \{s_1, \dots, s_N\}$

1.  $D \leftarrow \{\}$
2. **while**  $B \neq \text{empty}$  **do**
3.      $m \leftarrow \text{argmax } S$
4.      $M \leftarrow b_m$
5.      $D \leftarrow D \cup M$ ;  $B \leftarrow B - M$
6.     **for**  $b_i$  in  $B$  **do**
7.          $S_i \leftarrow S_i \cdot f(\text{IOU}(M, b_i))$
8.     **end**
9. **end**
10. **return**  $D, S$

In eq. (5),  $x_i$  and  $y_i$  are the coordinates of the predicted object, and  $\hat{x}_i$  and  $\hat{y}_i$  are the coordinates of the actual object.

$$Loss_2 = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B \ell_{ij}^{obj} \left[ \left( \sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left( \sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \tag{6}$$

In eq. (6),  $w_i$  and  $h_i$  are the width and height of the predicted object, and  $\hat{w}_i, \hat{h}_i$  are the width and height of the actual object.

$$Loss_3 = \sum_{i=0}^{S^2} \sum_{j=0}^B \ell_{ij}^{obj} \left[ \hat{C}_i \log(C_i) + (1 - \hat{C}_i) \log(1 - \hat{C}_i) \right] - \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^B \ell_{ij}^{noobj} \left[ \hat{C}_i \log(C_i) + (1 - \hat{C}_i) \log(1 - \hat{C}_i) \right] \tag{7}$$

In eq. (7),  $C_i$  is the confidence of the predicted object, and  $\hat{C}_i$  is the confidence of the actual object;

$$Loss_4 = \sum_{i=0}^{S^2} \ell_{ij}^{obj} \sum_{c \in \text{classes}}^B \left[ \hat{p}_i(c) \log(p_i(c)) + (1 - \hat{p}_i(c)) \log \hat{p}_i(1 - p_i(c)) \right] \tag{8}$$

In eq. (8),  $P_i$  is the category probability of the predicted object, and  $\hat{P}_i$  is the category probability of the actual object;

The total loss is:

$$\text{Total Loss Function} = \sum_{i=1}^4 Loss_i \tag{9}$$

where,  $\lambda_{coord}$  and  $\lambda_{noobj}$  are weights that are set to balance the weight of each loss. We set  $\lambda_{coord} = 5$  and  $\lambda_{noobj} = 0.5$ .  $S$  is a grid cell,  $B$  is a bounding box,  $obj$  contains an object, and  $noobj$  does not contains an object.

### 3.3 CLAHE-mango

Illumination is one of the important interference factors in the detection of green mangoes. During the daytime, due to uneven light exposure, green mangoes often appear backlit and smooth. At night, due to the lack of light, the green mango is easily integrated into the background. The green mango images acquired in the above two situations greatly affect the accuracy of the detection network. In view of this fact, based on the CLAHE algorithm [30], we study the characteristics of green mangoes and propose a green mango image enhancement algorithm (CLAHE-Mango) under natural lighting conditions. First, the pixel value is evenly distributed to solve the problem of high local brightness of the green mango image, such as backlighting and smoothing. Second, the contrast threshold is set to improve the clarity of a night green mango and improve the detection accuracy of the method. Since the green mango with too high contrast will lose most of its features, which is not conducive to detection, we set clipLimit = 3 (contrast threshold). The ratio of a green mango individual to the image size in the green mango data set tends to 1:54. Therefore, we set titleGridSize = (8, 8) as



the grid size for pixel equalization. The enhanced result is shown in Fig. 8.

Taking the night green mango image as an example, the content of Fig. 8 (a) is dark, which is not conducive to detection, and its histogram also proves the following point: the gray value of Fig. 8 (c) is clustered between 0 and 50. In Fig. 8 (b), the enhanced green mango becomes clear and visible, and its histogram in Fig. 8 (d) has gray values that are well distributed between 0 and 255. CLAHE-Mango is the same as the auxiliary lighting source when the robot actually picks the green mangoes. It is also an auxiliary means, and it contributes 0.6% to the F1% of the final trained model. Therefore, we believe that CLAHE-Mango is one of the advantageous ways to solve the problem of green mango detection.

### 3.4 Light-YOLOv3 training and detecting process

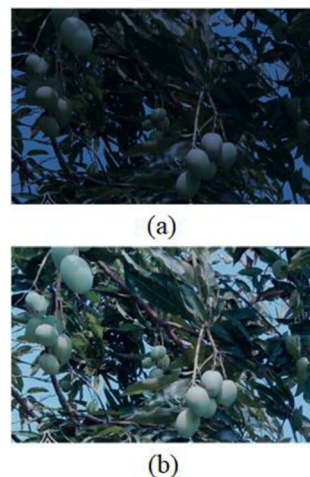
The basic process of the fast detection method of green mangoes in complex scenes by the picking robots includes two parts: model training and green mango detection. As shown in Fig. 9, model training is to input the green mango data set and the corresponding labels into Light-YOLOv3, perform iterative training, and obtain a fully trained model. Green mango detection is the step in which the picking robot obtains the image through the camera and sends it to the trained model to obtain the position of the green mangoes in the image.

## 4 Experiments

### 4.1 Green mango dataset

The experimental orchard is located in the Changjiang mango planting base in Hainan, China. The experimental mango variety is green mango, and the fruit is oblong. In July 2019, we used the Hikvision AI camera DS-2CD7T47DWD-IZ (sensor type: 1/1.8 “Progressive Scan CMOS; lens: 4 mm~9 mm,

**Fig. 8** Contrast results of image enhancement algorithms applied to green mangoes. (a) Night green mango image; (b) enhanced night green mango image; (c) histogram of (a) diagram; (d) histogram of (b) diagram



F1.6; maximum image size:  $2560 \times 1440$ ), and the distance of each crown is 1 to 2 m at multiple angles for the image collection. To ensure the diversity of the green mango dataset and the balance of the training, we separately collected images during the day and night, and the illumination of the green mango images collected during the day was more complicated. Part of the green mango dataset is shown in Fig. 10.

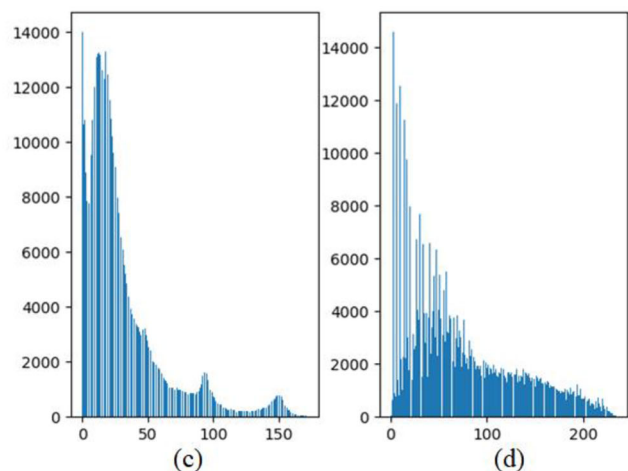
In the experiments in this paper, a total of 1500 green mango images from actual orchards and 500 green mango images from Internet sources were collected in JPG format. To shorten the model training time, we uniformly cropped and reduced the images to  $416 \times 416$  pixels. To improve the accuracy of the detection model, we accounted for the various appearances and forms of green mangoes in the labeling process, and we used Labeling to manually label 8432 green mangoes in 2000 images. The labeling information is saved in the format of the PASCALVOC [31] dataset, which contains the category and bounding box of the object. The parameter details of the green mango dataset are shown in Table 4.

In the orchard, under natural light, the green mangoes block each other or reflect against the light, which results in shadows on the fruit surface. The shadow makes the color of the fruit very different from the color of the diffuse reflection area under normal light, which affects the quality of the green mango images. We use the green mango image enhancement algorithm on the green mango dataset to improve the image quality and help to improve the network detection accuracy.

### 4.2 Experimental environment and evaluation index

#### 4.2.1 Experimental environment

In this experiment, the operating system used for training is Ubuntu 18.04, the test framework is TensorFlow, the processor is i7 8700k 3.70 GHz, 16 GB RAM, and the graphics card is Nvidia GeForce GTX1080. We used the CUDA version 10.0 parallel computing frameworks and



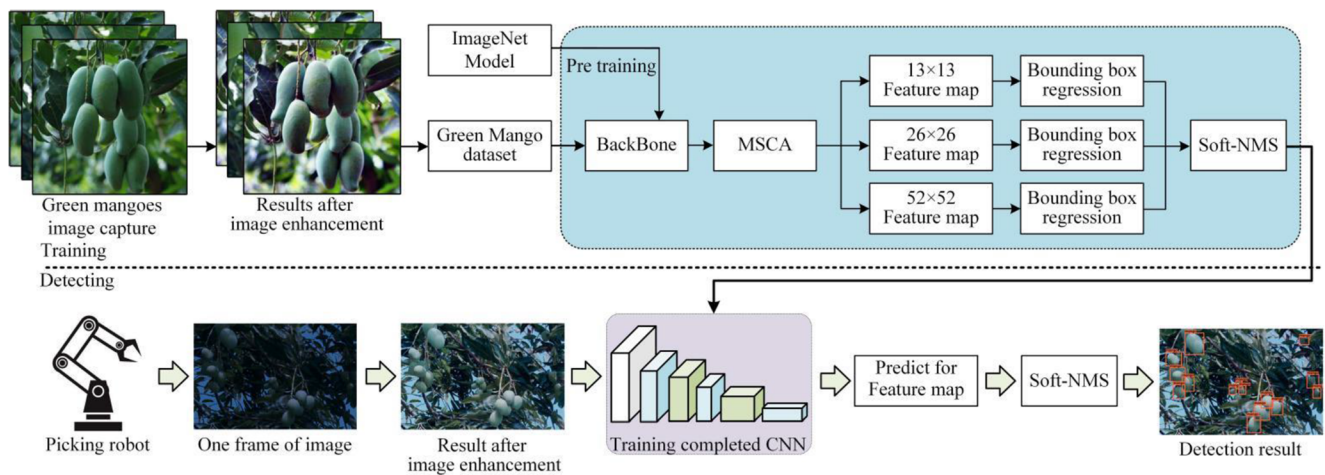


Fig. 9 The training and detecting process of the green mango detection method

CUDNN version 7.3 deep neural network acceleration library, and we used Python language programming to implement the training and detecting in the green mango detection network model.

### 4.2.2 Evaluation index

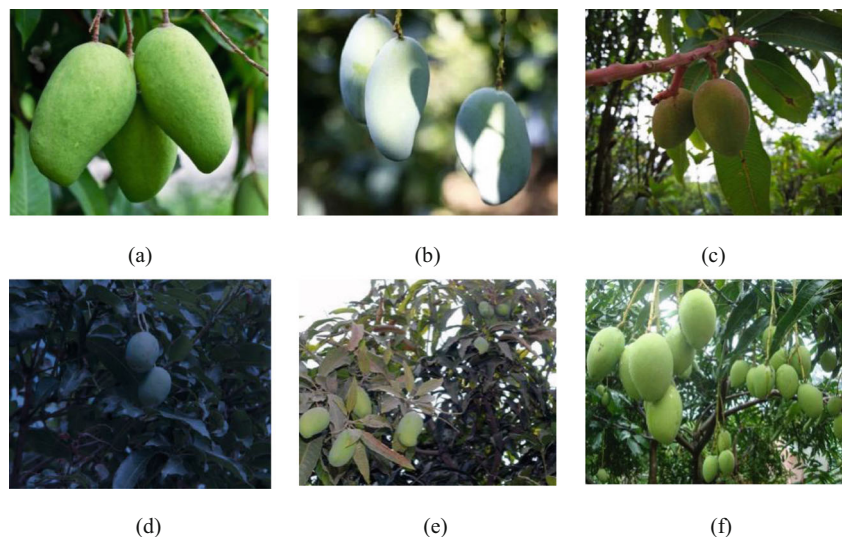
In this paper, F1%, FLOPs, volume, FPS and time (ms) are used to evaluate Light-YOLOv3 and other methods. The calculation formulas for the precision ( $P$ ) and F1% ( $F1$ ) are shown in eq. (10).

$$P = \frac{TP}{TP + FP}, R = \frac{TP}{TP + FN}, F1 = \frac{2PR}{P + R}, \quad (10)$$

where  $P$  is the precision rate,  $R$  is the recall rate,  $TP$  is the number of true positive samples,  $FP$  is the number of false

positive samples, and  $FN$  is the number of false negative samples.

Fig. 10 Some of the images in the green mango dataset (a) Side-lit green mangoes; (b) green mangoes under smooth light; (c) backlit green mangoes; (d) green mangoes under dark light; (e) shaded green mangoes; (f) dense green mangoes



## 4.3 Experiment results

### 4.3.1 Light-YOLOv3 detection results on the green mango dataset

In Table 5, we compare Light-YOLOv3 and state-of-the-art object detection methods. All of the methods are trained with the green mango data set and optimal parameters. The comparison information involved in Table 5 includes F1%, FLOPs, volume, FPS, and time (ms). The FPS and time of the two-stage object detection methods are far lower than those of one-stage detection methods result, and thus, so they do not participate in the discussion. Finally, we embed Light-YOLOv3 and the comparative one-stage detection methods into the current two excellent mobile chips to study the performance of Light-YOLOv3 on embedded platforms. Adreno 640 (672 MHz) and Mali-G76 (720 MHz) are the latest mobile GPUs developed by Qualcomm and ARM, respectively. The former is integrated into the Snapdragon 855plus CPU,

**Table 4** Details of the green mango dataset

Time	Daytime			Night	Total
	Side Light	Direct Sunlight	Back Light		
Number of images	500	500	500	500	2000
Number of bounding boxes	2100	2165	2071	2096	8432

and the latter is integrated into the Kirin 990 CPU. Table 5 shows the performance of Light-YOLOv3 and its comparison methods on the above two embedded platforms.

It can be seen from Table 5 that the one-stage object detection methods are significantly faster than the two-stage detection methods. The five indicators of Light-YOLOv3 are clearly better than YOLOv3. The detection time of Light-YOLOv3 is 1/5 of YOLOv3, and the FLOPs is 1/6 of YOLOv3. The F1% of Light-YOLOv3 is not only better than that of Faster R-CNN, R-FCN, and Mask R-CNN but also faster than the two-stage methods. Light-YOLOv3 can achieve real-time effects on both the Adreno 640 and Mali-G76 platforms. In the comparison method, only YOLOv3-tiny can achieve a real-time effect, but its accuracy is too low to be applied to the actual production environment. We use the fully trained model to test part of the green mango dataset, as shown in Fig. 11.

According to Fig. 11, it can be concluded that Light-YOLOv3 can successfully detect green mangoes in side light, direct light, backlight, dense, long distance, occlusion and special angle scenes. To further verify the effectiveness of the model, the efficiency of the method must be tested under various practical conditions. YOLOv3 is the original method, and YOLOv3-tiny is currently one of the most suitable methods for embedded platforms. Therefore, the following experiments use the number of green mangoes (single, multiple, dense), light intensity (side light, direct light, backlight)

and extreme conditions (night, long distance, special angle) as control variables. We compare the test results of Light-YOLOv3, YOLOv3, and YOLOv3-tiny under the above conditions, and we use F1% to evaluate the performance of the three methods.

#### 4.3.2 Experiment a: Comparison of the results of the detection methods under different numbers of green mangoes

In the actual picking process of the robot, as the distance between the camera and the green mango tree changes, the number and size of the green mangoes in the acquired image will also change accordingly. When the number of green mangoes is small and the size is large, the outline and texture are clearly visible, and thus, it is easier to detect. However, in the multitarget detection task, with the increase in the number of green mangoes and the reduction in scale, there will be adhesion and occlusion, which are difficult to identify. Therefore, we set up comparative experiments for the detection of green mangoes under different numbers. The control variables were one green mango, multiple green mangoes and dense green mangoes. The detection performances of the three methods under different numbers of green mangoes were compared. The detection results are shown in Fig. 12.

**Table 5** Comparison of the training results of Light-YOLOv3 and state-of-the-art methods on the green mango dataset

	Method	F1%	FLOPs	Volume	GTx1080		Adreno 640		Mali-G76	
					FPS	Time	FPS	Time	FPS	Time
Two-stage	Faster R-CNN	93.8	132.86 BN	528 MB	–	–	–	–	–	–
	R-FCN	96.4	69.27 BN	277 MB	–	–	–	–	–	–
	Mask R-CNN	<b>96.7</b>	<b>61.52 BN</b>	<b>244 MB</b>	–	–	–	–	–	–
One-stage	SSD512	90.1	68.75 BN	275 MB	8	125.0	1	1501.5	1	1388.8
	RetinaNet400	93.2	63.92 BN	252 MB	14	71.4	1	862.1	2	787.4
	YOLOv2	91.6	42.94 BN	192 MB	52	19.2	4	250.2	5	211.8
	YOLOv3	94.2	65.86 BN	237 MB	36	27.8	3	333.3	3	305.3
	YOLOv3-tiny	83.8	7.36 BN	34 MB	213	4.5	17	58.8	19	51.7
	Mini-YOLOv3 [32]	92.1	14.81 BN	56 MB	78	12.8	6	153.8	7	141.0
	Light-YOLOv3	<b>97.7</b>	<b>10.12 BN</b>	<b>44 MB</b>	<b>192</b>	<b>5.2</b>	<b>16</b>	<b>62.5</b>	<b>17</b>	<b>57.3</b>

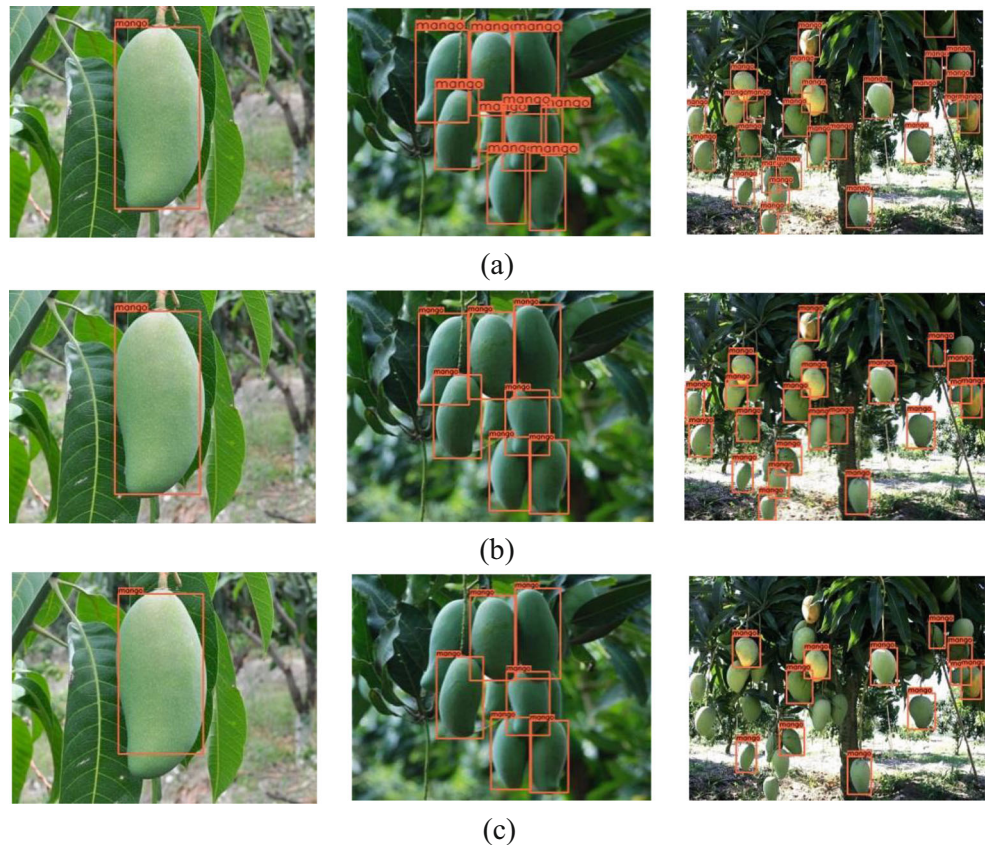


Fig. 11 The results of green mango detection in complex scenes by Light-YOLOv3

In this experiment, we selected a total of 300 images from the test set, which contained 1215 green mangoes. According to the number of green mangoes in the image,

they were divided into one green mango image (100 images that contained 100 objects) and multiple green mango images (150 images that contained 450 objects) and dense

Fig. 12 Three methods were used to detect different amounts of green mangoes. (a) Detection results of Light-YOLOv3; (b) Detection results of YOLOv3; (c) Detection results of YOLOv3-tiny



green mango images (50 images that contained 665 objects). We use Light-YOLOv3, YOLOv3, and YOLOv3-tiny to detect the test set, and we obtained the positive sample number, the total number of samples, the number of undetected samples and the number of erroneously detected samples, and we calculated the accuracy rate and recall rate to yield the get F1% Value. We repeated the above steps 3 times, taking the average value, and finally, we averaged the three types of results to obtain the comprehensive effect. The final results are shown in Table 6.

It can be seen from the combination of Fig. 12 and Table 6 that in the green mango detection experiment with different densities, the F1% of Light-YOLOv3 is 4.6 percentage points higher than that of YOLOv3, and YOLOv3-tiny performs the worst. We found that the performance of the three methods has a common trend: the smaller the number of green mangoes, the higher the value of F1%. When the number increases from 1 to less than 10, the F1% does not decrease much. When the number of green mangoes in an image exceeds 10, the F1% will be greatly reduced. From the comprehensive results, Light-YOLOv3 is capable of detecting different numbers of green mangoes.

#### 4.3.3 Experiment B: Comparison of the results of the detection methods under different illumination angles

In this experiment, we used the angle of light when shooting green mangoes as a control variable, including side light (85 images that contained 412 objects), direct light (85 images that contained 454 objects) and backlit (80 images that contained 401 objects). Because the complex illumination angle of the dense green mango sample has a large influence on the

**Table 6** Test results of the three methods for images with different numbers of green mangoes

Number	Method	F1%			
		1	2	3	Average
1	YOLOv3	94.4	95.1	94.3	94.6
	YOLOv3-tiny	83.6	84.2	83.9	83.9
	Light-YOLOv3	98.8	98.3	98.7	98.6
>2 ~ 10	YOLOv3	94.1	94.3	94.2	94.2
	YOLOv3-tiny	82.7	83.2	82.4	82.8
	Light-YOLOv3	97.5	97.9	97.2	97.5
>10	YOLOv3	93.5	92.8	93.7	93.3
	YOLOv3-tiny	81.3	80.2	81.7	81.1
	Light-YOLOv3	96.9	97.3	96.8	97.0
Average	YOLOv3	94.0	94.1	94.1	94.1
	YOLOv3-tiny	82.5	82.5	82.6	82.6
	Light-YOLOv3	97.7	97.8	97.6	97.7

detection results, the dense green mango sample will not be considered when selecting the image here. The detection results of the three methods are shown in Fig. 13, and the statistical results are shown in Table 7.

It can be seen from Fig. 13 that the common tendencies of the three methods to detect green mangoes under different illumination angles are as follows: the texture details of green mangoes under side light are clearer, the surface light intensity is uniform, and it is easier to detect. Part of the surface of the green mangoes under direct light is enhanced in brightness, giving it a bright white color with few texture features. When backlighting, the brightness of the green mangoes, and branches and leaves are significantly reduced, and the boundary between the two is not obvious. In the latter two cases, the detection of the green mangoes is difficult.

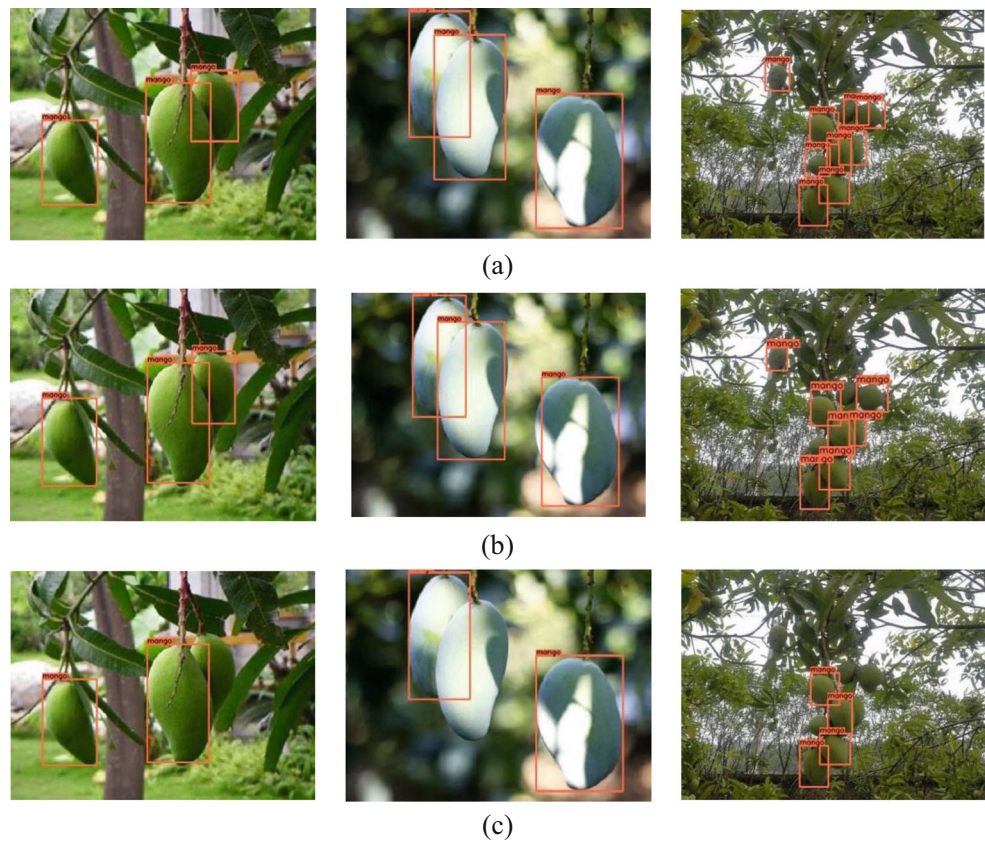
From Table 7, the F1% of Light-YOLOv3 is 10 percentage points higher than that of YOLOv3 and 15 percentage points higher than that of YOLOv3-tiny. The three methods have the same F1% gradient in the three cases. The three methods perform the worst when there is backlighting is back. The reason is that the lack of brightness during the backlighting causes the green mango to have an unclear contour, which lowers the model's F1%.

#### 4.3.4 Experiment C: Comparison of the results of the detection methods under extreme conditions

The experiments in this section mainly discuss the efficiency of robots picking green mangoes under extreme conditions. The three extreme situations are as follows: 1. Considering the possibility of picking robots working around the clock, the detection performance of green mango at night is also important; 2. Considering the difference in the picking distance of different models of robots, we must study methods to detect the green mangoes remotely; 3. Considering the possibility of the robot picking a green mango from various angles, we must research methods to detect the performance on the green mangoes from the upward and downward angles of the green mango (the manipulator picks on the top or bottom of the green mango). This experiment uses night time (100 images that contain 405 targets), long distance (80 images that contain 539 targets) and special angles (80 images that contain 478 targets) as the test set. The calculation method is similar to the previous section. The detection effect of this experiment is shown in Fig. 14, and the statistical results are shown in Table 8.

As seen from Fig. 14, Light-YOLOv3 can successfully detect green mangoes under three extreme conditions; YOLOv3 can detect only some of the green mangoes; YOLOv3-tiny can detect only some of the green mangoes, those closest to the camera. At night, due to the lack of light sources, unlike the backlight of the daytime, there is also a diffuse reflection of the green mangoes. Most parts of the

**Fig. 13** Detection results of green mangoes by three methods under different illumination. (a) Detection results of Light-YOLOv3; (b) Detection results of YOLOv3; (c) Detection results of YOLOv3-tiny



green mangoes appear black. In this case, the green mangoes have almost the same color, nature, or texture characteristics. The green mangoes in this case greatly affect the detection accuracy of the methods. Due to the multiscale structure, Light-YOLOv3 and YOLOv3 can detect long-distance green mangoes better. Clearly, YOLOv3-tiny is unable to detect

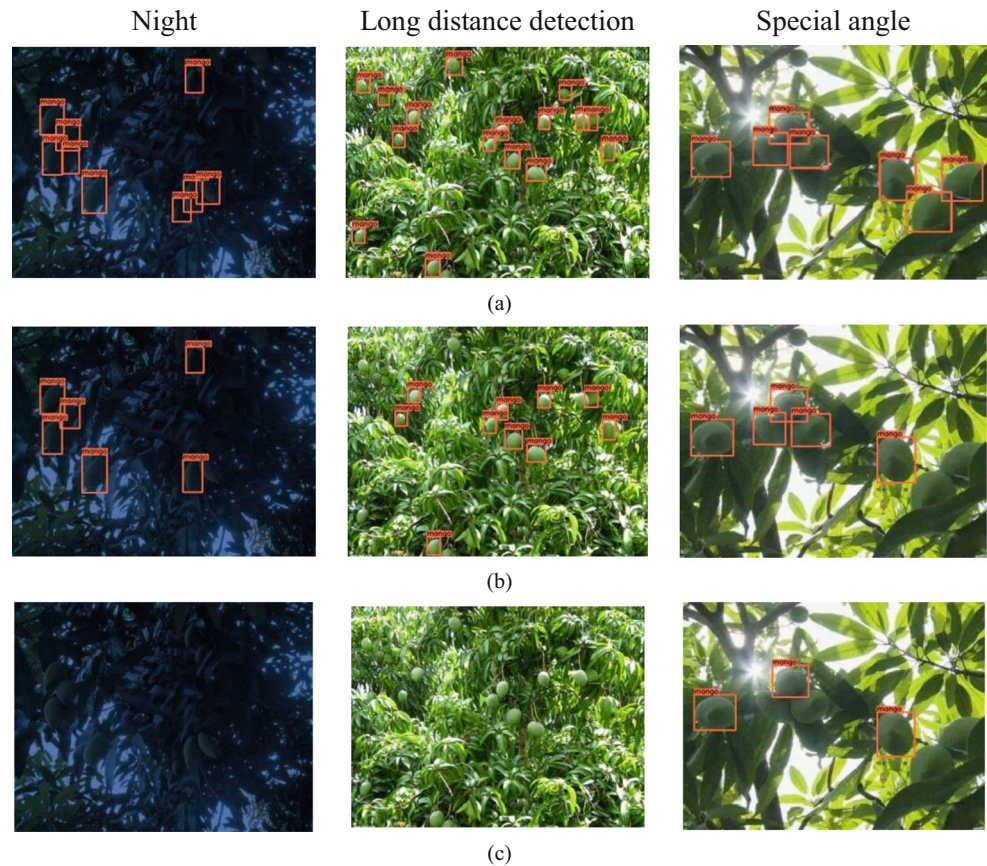
green mangoes at long distances. The green mango's top and bottom angles are both elliptical, with a shadow in the middle, and the shape is quite different from the normal angle of view of the green mango. It can be seen from Table 8 that Light-YOLOv3 has the highest F1% in three cases. At night, Light-YOLOv3 is 4 percentage points higher than YOLOv3 and 29.9 percentage points higher than YOLOv3-tiny. Light-YOLOv3 is 3.9 percentage points higher than YOLOv3 at long distances. In particular, YOLOv3-tiny has almost no ability to detect green mangoes at a long distance. Because its backbone network has only 7 layers, it cannot extract higher-level semantic features, and thus the detection F1% is low. Under special angles, Light-YOLOv3 is 2.9 percentage points higher than YOLOv3 and 20.9 percentage points higher than YOLOv3-tiny.

**Table 7** Detection results of the three methods for different illumination of green mango images

Illumination angles	Method	F1%			
		1	2	3	Average
Side Light	YOLOv3	94.5	94.1	93.8	94.1
	YOLOv3-tiny	83.8	84.2	83.7	83.9
	Light-YOLOv3	98.2	97.6	98.1	98.0
Direct Light	YOLOv3	93.7	93.0	92.9	93.2
	YOLOv3-tiny	82.4	82.8	81.9	82.4
	Light-YOLOv3	97.4	97.2	97.3	97.3
Back Light	YOLOv3	92.0	93.1	91.8	92.3
	YOLOv3-tiny	81.5	80.7	81.3	81.2
	Light-YOLOv3	96.5	96.3	95.7	96.2
Average	YOLOv3	93.4	93.4	92.8	93.2
	YOLOv3-tiny	82.5	82.5	82.3	82.5
	Light-YOLOv3	97.4	97.0	97.0	97.2

In summary, with the YOLOv3-tiny method, it is more difficult to detect green mangoes in dense, backlit, night, long distance and other environments, and the F1% is usually low. YOLOv3 can detect most green mangoes, but it requires the most computation. In contrast, the detection results of Light-YOLOv3 are better than those of YOLOv3 in various situations, and its computation amount is 1/5 of YOLOv3. Three sets of comparative experiments prove that Light-YOLOv3 can detect green mangoes in complex scenes, making it possible for green mango picking robots equipped with this visual method to work around the clock.

**Fig. 14** Detection results on the of three methods for green mangoes under extreme conditions. (a) Detection results of Light-YOLOv3; (b) Detection results of YOLOv3; (c) Detection results of YOLOv3-tiny



#### 4.4 Model analysis

This section discusses mainly the progressive performance improvement from YOLOv3 to Light-YOLOv3. These operations include LightNet v1 (depth separable convolution, Channel Shuffle, Group Normalization), MSCA module, Soft-NMS and CLAHE-Mango. To prove that these operations are beneficial to the performance of our method, after each operation, we calculated the corresponding FLOPs, F1%

**Table 8** Three methods to detect images of green mangoes in extreme conditions

Extreme condition	Method	F1%			
		1	2	3	Average
Night	YOLOv3	92.2	91.6	91.9	<b>91.9</b>
	YOLOv3-tiny	67.4	65.3	64.9	<b>65.9</b>
	Light-YOLOv3	96.3	95.5	95.7	<b>95.8</b>
Long distance detection	YOLOv3	90.3	91.2	90.7	<b>90.7</b>
	YOLOv3-tiny	0.0	0.0	0.0	<b>0.0</b>
	Light-YOLOv3	94.4	95.1	94.3	<b>94.6</b>
Special angle	YOLOv3	89.9	90.6	89.6	<b>90.0</b>
	YOLOv3-tiny	70.3	73.2	72.5	<b>72.0</b>
	Light-YOLOv3	93.4	92.8	92.6	<b>92.9</b>

and FPS, which are the performance results under Nvidia GeForce GTX1080, as shown in Table 9.

From Table 9, in the design process of the backbone, first, simply using the deep separable convolution for YOLOv3 can reduce its model calculation by 25.72 BN. The core of the deep separable convolution is DWConv, and the combined structure of DWConv and PWConv is similar to the standard effect. Therefore, the F1% of YOLOv3 with the deep separation convolution will not be significantly different, but its FPS doubles. Channel split can divide the channel dimension of the input feature map into two branches, and the two branches are combined together through the concat operation after a series of convolution operations. The concat results are subjected to channel shuffle to ensure the full communication of the two branches. Using this operation reduces the amount of computation by almost half compared to the standard depth separable convolution, and the F1% value is increased by almost 1%, while the FPS is increased by 65. In the training process, to avoid the impact of the batch size on the training results, we replaced Group Normalization with Batch Normalization, which increased the model F1% by 0.4. DWConv, channel split and group normalization are the core of LightNet v1. We replaced LightNet v1 with the residual unit of YOLOv3, which reduced the calculation of the model by 45.38 BN, while F1% increased by 1.3, and FPS increased by 107.

**Table 9** Analysis of the performance improvement brought by each module in Light-YOLOv3

Method	LightNet v1			MSCA	Soft-NMS	CLAHE-Mango	FLOPs	F1%	FPS
	DWConv	Channel Split	GN						
YOLOv3							65.86 BN	94.2	36
	√						38.34 BN	93.9	75
	√	√					21.75 BN	95.1	140
	√	√	√				20.48 BN	95.5	143
	√	√	√	√			9.83 BN	96.7	198
	√	√	√	√	√		10.12 BN	97.1	192
Light-YOLOv3	√	√	√	√	√	√	10.12 BN	97.7	192

To further improve the model's ability to detect green mangoes at long distances, we used the improved MSCA module, which combines details and semantic information in different layers of the network. This structure also helps to detect green mangoes in the case of occlusion. LightNet v1 in the MSCA module was used to reduce the amount of computation and improve the efficiency. Compared with the original network, the calculation volume of the network model using MSCA is reduced by half, F1% is greatly increased by 1.2%, and FPS is increased by 55. To improve the model's ability to detect overlapping green mangoes, we replaced Soft-NMS with NMS to enable two green mangoes with larger IOU values to also be detected. The F1% of the models using Soft-NMS increased by 0.4. Illumination is one of the important interference factors in green mango detection. From the green mango dataset, we can use CLAHE-Mango to assist in the detection of green mangoes at night. We used CLAHE-Mango in both training and detection to help detect green mangoes in low-light scenes. The F1% value of the model using the CLAHE-Mango is increased by 0.6. In summary, it is necessary to adopt the operations in Table 9 for YOLOv3. The above three sets of comparative experiments also proved the actual effect of the above improvements.

## 5 Conclusions

In this paper, we have proposed a fast detection method for green mangoes, Light-YOLOv3. First, combining the characteristics of green mangoes such as color, texture and shape, a lightweight network unit was designed to replace the Res Net unit in YOLOv3, which greatly improved the detection speed of the method. Second, an MSCA module is used to concatenate multilayer features and make predictions, solving the problem of insufficient position information and semantic information of the prediction feature map in YOLOv3, and it effectively improves the detection effect of the green mango. To solve the overlapping occlusion problem of green

mangoes, we used Soft-NMS instead of NMS. Finally, combined with the characteristics of green mangoes, a green mango image enhancement (CLAHE-Mango) algorithm under natural lighting conditions is proposed to improve the accuracy of the detection method. Experimental results show that Light-YOLOv3 has the advantages of less memory occupation, high accuracy, and fast detection speed, and it can complete the detection and positioning tasks of green mangoes in complex scenarios. This method can provide a reference for early yield estimation of fruit trees and automatic picking of green fruits.

**Acknowledgments** The authors are grateful for collaborative funding support from the Natural Science Foundation of Shandong Province, China (ZR2018MEE008), the Key Research and Development Program of Shandong Province, China (2017GSF20115).

## References

1. He Z, Xiong J, Lin R, Zou X, Tang L, Yang Z (2017) A method of green litchi recognition in natural environment based on improved LDA classifier. *Computers & Electronics in Agriculture* 140:159–167
2. Lu J, Sang N (2015) Detecting citrus fruits and occlusion recovery under natural illumination conditions. *Computers & Electronics in Agriculture* 110:121–130
3. Li H, Lee W, Wang K (2016) Immature green citrus fruit detection and counting based on fast normalized cross correlation (FNCC) using natural outdoor colour images. *Precis Agriculture* 17(6):678–697
4. Rajneesh B, Won S, Saumya S (2013) Green citrus detection using fast Fourier transform (FFT) leakage. *Precis Agric* 14(1):59–70
5. Lu J, Lee W, Gan H, Hu X (2018) Immature citrus fruit detection based on local binary pattern feature and hierarchical contour analysis. *Biosyst Eng* 171:78–90
6. Stein M, Bargoti S, Underwood J (2016) Image based mango fruit detection localisation and yield estimation using multiple view geometry. *Sensors* 16(11):57–64
7. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. *Nature* 521:436–444
8. Xu Z, Jia R, Liu Y, Zhao C, Sun H (2020) Fast method of detecting tomatoes in a complex scene for picking robots. *IEEE Access* 8:55289–55299



9. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 580–587
10. Girshick R (2015) Fast R-CNN. IEEE international conference on computer vision (ICCV) 1440–1448
11. Ren S, He K, Girshick R, Sun J (2015) Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell 39:1137–1149
12. Dai J, Li Y, He K, Sun J (2016) R-FCN: object detection via region based fully convolutional networks. Neural information processing systems 379–387
13. He K, Gkioxari G, Dollár P, Girshick R (2017) Mask R-CNN. Proc IEEE international conference on computer vision (ICCV) 2961–2969
14. Sa I, Ge Z, Dayoub F (2016) DeepFruits: A Fruit Detection System Using Deep Neural Networks. Sensors 16(8):888–911
15. Liu W, Anguelov D, Erhan D, Szegegy C, Reed S et al (2016) SSD: single shot multibox detector. European Conference on Computer Vision 21–37
16. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: unified real-time object detection. IEEE conference on computer vision and pattern recognition (CVPR) 779–788
17. Redmon J, Farhadi A (2017) YOLO9000: Better Faster Stronger. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 7263–7271
18. Redmon J, Farhadi A (2018): YOLOv3: an incremental improvement. [online] Available: <https://arxiv.org/abs/1804.02767>
19. Lin T, Goyal P, Girshick R, He K, Dollár P (2017) Focal loss for dense object detection. IEEE international conference on computer vision (ICCV) 2980–2988
20. Kim S, Kook H, Sun J, Kang M (2018) Parallel feature pyramid network for object detection. The European Conference on Computer Vision (ECCV) 234–250
21. Bodla N, Singh B, Chellappa R (2017) Soft-NMS — improving object detection with one line of code. IEEE international conference on computer vision (ICCV) 5562–5570
22. Neubeck A, Vangool L, (2006) Efficient non-maximum suppression. 18th international conference on pattern recognition 3: 850–855
23. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 770–778
24. Sandler M, Howard A, Zhu M, Zhmoginov A, and Chen L (2018) Mobilenetv2: inverted residuals and linear bottlenecks. IEEE conference on computer vision and pattern recognition (CVPR) 4510–4520
25. Zhang, X, Zhou, X, Lin M, Sun J (2017) ShuffleNet: an extremely efficient convolutional neural network for Mobile devices. Conference on Computer Vision and Pattern Recognition (CVPR) 6848–6856
26. Ma N, Zhang X, Zheng H (2018) ShuffleNet v2: practical guidelines for efficient CNN architecture design. European Conference on Computer Vision (ECCV) 1440–1448
27. Krizhevsky A, Sutskever I, Hinton G (2017) ImageNet classification with deep convolutional neural networks. Commun ACM 60(6):84–90
28. Ioffe S, Szegegy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. International conference on machine learning 448–456
29. Wu Y, He K (2019) Group normalization. Int J Comput Vis 128(3): 742–755
30. Reza A (2004) Realization of the contrast limited adaptive histogram equalization (clahe) for real-time image enhancement. Journal of Vlsi Signal Processing System for Signal 38(1):35–44
31. Everingham M, Eslami A, Gool L, Williams C, Winn J (2014) The PASCAL visual object classes challenge: a retrospective. Int J Comput Vis 111:98–136
32. Mao Q, Sun H, Liu Y (2019) Mini-YOLOv3: real-time object detector for embedded applications. IEEE Access 7:133529–133538

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Zhi-Feng Xu** was born in Shandong, China, in 1994. He received the B. S. degree from Shandong university of technology, China, in 2018. He is currently pursuing the M.S degree with Shandong university of science and technology. His research interest includes image processing and deep learning.



**Rui-Sheng Jia** is currently a full professor at the College of Computer Science and Engineering, Shandong University of Science and Technology, China. He has more than 30 first-author publications and has more than 25 co-author publications. His research interest includes artificial intelligence, big data processing, information fusion, microseismic monitoring and inversion.



**Hong-Mei Sun** received the B.S. and M.S. degree in computer science from Shandong University of Science and Technology, China, in 1995 and 2005, respectively. She is currently a associate professor at the College of Computer Science and Engineering, Shandong University of Science and Technology, China, and is the leader of a Key Research and Development Projects of Shandong Province, China. She has four first-author publications

and has five co-author publications. Her research interests include microseismic monitoring technology and software engineering.



**Qing-Ming Liu** was born in Shandong, China, in 1995. He received the B. S. degree from Shandong University of Science and Technology, China, in 2018. He is currently pursuing the M.S degree with Shandong University of Science and Technology. His research interests include image processing and deep learning.



**Zhe Cui** was born in Shandong, China, in 1998. He received the B. S. degree from Shandong university of Science and technology, China, in 2019. He is currently pursuing the M.S degree with Shandong university of science and technology. His research interest includes image processing and deep learning.