Computers and Electronics in Agriculture 179 (2020) 105856

Contents lists available at ScienceDirect

Computers and Electronics in Agriculture

journal homepage: www.elsevier.com/locate/compag

Real-time kiwifruit detection in orchard using deep learning on Android™ smartphones for yield estimation

Zhongxian Zhou [a], Zhenzhen Song [a], Longsheng Fu [a,b,c,*], Fangfang Gao [a], Rui Li [a], Yongjie Cui [a]

[a] College of Mechanical and Electronic Engineering, Northwest A&F University, Yangling, Shaanxi 712100, China
[b] Key Laboratory of Agricultural Internet of Things, Ministry of Agriculture and Rural Affairs, Yangling, Shaanxi 712100, China
[c] Shaanxi Key Laboratory of Agricultural Information Perception and Intelligent Service, Yangling, Shaanxi 712100, China

ARTICLE INFO

Keywords:
MobileNetV2
InceptionV3
Quantization
Android
Yield estimation

ABSTRACT

Fast and accurate detection of kiwifruit in orchard under natural environment is the primary technology for yield estimation. Deep learning has become a prevalent way of fruit detection and achieved outstanding results. Besides, easy-carry smartphones are getting popular and powerful. In this paper, single shot multibox detector (SSD) with two lightweight backbones MobileNetV2 and InceptionV3 were employed to develop an Android APP named KiwiDetector for field kiwifruit detection. An 8-bit quantization method was used to compress model size and improve detection speed by quantizing weight tensor and activation function data of convolutional neural networks from 32-bit floating point to 8-bit integer. Detection test was performed on 100 selected kiwifruit field images with resolution of 3,968 × 2,976 pixels using the four models on a HUAWEI P20 smartphone. Results showed that MobileNetV2, quantized MobileNetV2, InceptionV3, and quantized InceptionV3 obtained true detected rate (TDR) of 90.8%, 89.7%, 87.6%, and 72.8%, respectively. The TDR of MobileNetV2 and quantized MobileNetV2 was generally consistent and higher than InceptionV3 and quantized InceptionV3. For processing an image on the smartphone, MobileNetV2, quantized MobileNetV2, InceptionV3, and quantized InceptionV3 took about 163 ms, 103 ms, 1085 ms, and 685 ms with model sizes of 17.5 MB, 4.5 MB, 96.1 MB, and 24.1 MB, respectively. Quantized MobileNetV2 reached a significant TDR with the fastest detection speed and the smallest model size. The results indicated that the proposed Android APP is promising for yield estimation.

1. Introduction

China is the largest cultivator and producer of kiwifruits in the world. In 2018, the cultivation area of kiwifruits in China exceeded 1.68×10^5 ha with the yield of 2.04 million tons. Within China, Shaanxi Province has the highest yield of approximately 9.48×10^5 tons from a cultivation area of 5.3×10^4 ha in the same year (UN Food & Agriculture Organization, 2020). With such a large cultivation area and yield, knowing beforehand the amount of fruit to be harvested leads to better logistics and decisions making in the agricultural industry.

Fruit yield estimation will allow growers to plan fruit storage and sales more precisely and has been relied on time-consuming manual counting, which is thus highly desirable to be automated. Fruit yield estimation before the harvest is a crucial step to predict the required resources for workers such as packing and storage houses for the harvest and distribution resources for the marketing (Dorj et al., 2017; Feng et al., 2020; Zhang et al., 2020ab). Traditional methods for estimating yield are primarily manual, which are laborious and inaccurate. Accordingly, an automated yield estimation approach is necessary.

Fruit detection and counting are the primary critical technologies for automated yield estimation and have been studied by many scholars using conventional image processing methods (Fu et al., 2020b). Song et al. (2014) used a bag-of-words model and achieved a correlation of 94.6% between automatic and manual counting of fruit numbers. Ramos et al. (2017) counted the number of visible fruits on a coffee branch using texture information from a single side image of the branch and then estimated the total fruits, which reached a regression coefficient of 0.984. Bargoti and Underwood (2017) employed the watershed segmentation and circular Hough transform algorithms to detect and count individual apple fruits and reported an F1-score of 0.861. Dorj et al. (2017) applied thresholding, noise removal, watershed segmentation in HSV color space to detect and count citrus, which obtained a correlation coefficient R^2 of 0.93 between manual counting and counting algorithm. Fu et al. (2019) separated linearly clustered kiwifruits by scanning each

* Corresponding author at: College of Mechanical and Electronic Engineering, Northwest A&F University, Yangling, Shaanxi 712100, China.
E-mail addresses: fulsh@nwafu.edu.cn, longsheng.fu@outlook.com (L. Fu).

https://doi.org/10.1016/j.compag.2020.105856
Received 18 September 2020; Received in revised form 23 October 2020; Accepted 24 October 2020
Available online 2 November 2020

detected cluster to find the contact points between the adjacent fruits and drawing a separating line between the two closest contact points, which correctly separated and counted 92.0% of the kiwifruits. These approaches could detect and count fruits when images were captured under controlled lighting condition and had few fruits (Williams et al., 2019). To overcome these limitations, a well-generalized methodology that invariant and robust to brightness, different viewpoints and highly discriminative feature representations is desired.

Deep learning as a powerful technique in the artificial intelligence field is becoming a prevalent way of fruit detection and counting, and achieved promising results on desktop computers (Fu et al., 2020a, 2020c; Gao et al., 2020; Majeed et al., 2020b). Chen et al. (2019) applied Faster R-CNN (Region-based Convolutional Neural Network) based on ResNet-50 for strawberry detection and counting in field, which obtained an average precision (AP) of 0.83. Kestur et al. (2019) proposed MangoNet for mango detection and counting in an orchard and reported an F1-score of 0.844. Hani et al. (2020) combined U-Net with Faster R-CNN for apple detection and counting in an orchard and reached yield estimation accuracy of 97.8%. Santos et al. (2020) utilized Mask R-CNN with ResNet-101 to detect and count grapes in field, which showed a 0.84 F1-score at intersection over union (IoU) of 0.5. These studies showed that deep learning approach can detect and count lots of fruits in a complex background of natural environment and obtained remarkable result. However, these convolutional neural networks (CNNs) have too many convolutional layers and parameters, and thus require specific equipment to perform the huge calculations. Therefore, these CNNs has been mostly applied on desktop computers, whereas it is very inconvenient to be employed in orchard, especially difficult for farmers to estimate yield in orchard by themselves.

Easy-carry smartphones are getting powerful and have been employed for online classification and detection tasks with deep learning. Lu et al. (2017) developed an APP on smartphones to capture an image and upload the image to a remote could computer where Fully Convolutional Network (FCN) with VGG16 (Visual Geometry Group with 16 layers) was employed to perform wheat disease detection, which correctly detected 96.6% of the diseased area in the test images. Rawlani et al. (2018) implemented Faster R-CNN with ZFNet (Zeller Fergus Net) to detect vegetables and fruits on an online server from the images captured by an android phone, which obtained an average recall rate of 91% and a precision rate of 93%. However, these approaches are highly dependent on WiFi or mobile network. Moreover, core functions of these diagnosis systems are integrated into the server, while the smartphones only play the role of capturing images and display results. Besides, the problem of too many convolutional layers and parameters in the classic CNNs is still not solved.

To enable smartphones to run CNNs, single shot multibox detector (SSD) approach with two lightweight backbones, MobileNet and Inception, have been proposed. He et al. (2019) detected oilseed rape pests using SSD with MobileNetV1 and InceptionV3 on an Android smartphone, which achieved APs of 67.4% and 62.9% at IoU of 0.6, respectively. Liu et al. (2019) employed SSD with MobileNetV1 and InceptionV3 to develop an Android APP for plant disease classification, which correctly classified 95% and 95.6% of plant diseases on the PlantVillage dataset, respectively. Al-Hami et al. (2020) adopt an 8-bit quantization method to compress AlexNet and VGG16, which reduced their model sizes by 14% and 10%. The above experiments proved that SSD with MobileNet or Inception can be run on smartphones, and the 8-bit quantization method further reduces the model size. However, there is currently no offline fruit detection method for smartphones based on deep learning as we know.

In this study, the first kiwifruit detection Android APP based on deep learning technique has been developed. Four models using SSD with MobileNetV2, quantized MobileNetV2, InceptionV3, and quantized InceptionV3 were trained and converted to TensorFlow Lite models. These models were employed for kiwifruit detection on Android smartphones and evaluated by their performance. The optimal model was selected by comparing correctly detection rate, detection speed and model size to detect kiwifruit for real-time yield estimation.

2. Materials and methods

2.1. Image acquisition

Image data were collected by a smartphone (Huawei P20, Huawei Inc., Shenzhen, China) in an orchard under different illumination conditions. A selfie stick was held by an experimenter and was kept at around 1 m underneath the kiwifruit canopy. The smartphone was fixed on the selfie stick with its rear camera facing up the canopy to collect images. RGB (Red, Green, and Blue) images were taken in late October 2019 during the harvesting season on the most common cultivar 'Hayward' at Meixian Kiwifruit Experimental Station (34°07′39′'N, 107°59′50′'E, and 648 m in altitude) of the Northwest A&F University, Shaanxi, China. In total, 900 original images (450 images each in the morning and afternoon) with 3,968 × 2,976 pixels were collected under different lighting conditions, as shown in Fig. 1. One hundred images were selected randomly from the original images for performance evaluation on smartphones. There were 3,967 kiwifruits in the 100 selected images. Ground truth kiwifruit targets were manually annotated in the remaining 800 original images using rectangular annotations and saved in 'xml' files.

2.2. Data augmentation

Data augmentation, utilizing geometric transformation, increased the number of image datasets from 800 to 4,800. Deep learning training required a large number of images with diverse features to avoid overfitting and non-convergence. The image augmentation from the geometric transformation mainly included image mirroring and rotations. The image mirroring included a horizontal mirror and a vertical mirror. The horizontal mirror transformed the left and right parts of the image centering on the vertical line of the image. The vertical mirroring transformed the upper and lower parts of the image centering on the horizontal centerline of the image. The image rotations were rotating the image by 90°, 180°, and 270°, respectively. Meanwhile, the 'xml' file of the original image was modified correspondingly after the image was augmented. Through the above methods, the original images were augmented 5 times to a dataset of 4,800 images. The images in the dataset were divided into training set (3,840 images, 80%) and test set (960 images, 20%).

2.3. Meta-architecture of SSD

SSD has fast speed and high accuracy for detecting objects by a single deep neural network. Feature extraction network, SSD feature layers and detection layer are the main components of SSD. For this study, MobileNetV2, quantized MobileNetV2, InceptionV3, and quantized InceptionV3 were employed as the feature extraction networks to extract kiwifruit features. The specific implementation process of SSD is shown in Fig. 2. Firstly, SSD feeds input kiwifruit images through the feature extraction network and SSD feature layers to generate feature maps at different scales for the detection layer. Then, the detection layer extracts anchor boxes from different scale feature maps for kiwifruit detection. Lastly, SSD uses the detection layer to produce a series of predicted bounding boxes and confidence scores for kiwifruits that exist in each anchor box, and then generates the final output images. As shown in Fig. 2, an essential character of SSD is that each feature layer is connected to the detection layer. Furthermore, no resample of feature maps is required, which enabling SSD to operate in an entire feedforward manner.

Fig. 1. Kiwifruit images captured by a smartphone at morning (a) and afternoon (b) in the orchard.

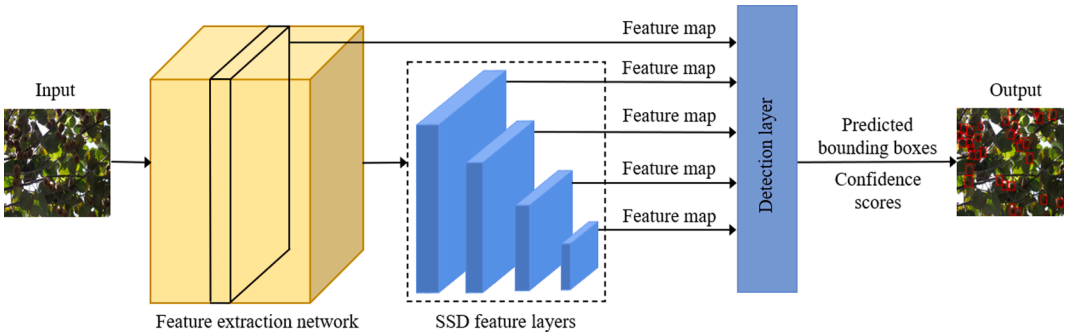

Fig. 2. SSD structure that used for kiwifruit detection.

2.4. Feature extraction network

2.4.1. MobileNetV2

MobileNetV2 is a network released by Google in 2018, designed for mobile and embedded devices (Sandler et al., 2018). Inverted residual module (IRM) is the basic and key unit of MobileNetV2. Fig. 3 showed the diagram of MobileNetV2 that includes 17 IRMs. A complete IRM is firstly performing a pointwise convolution to expand the number of feature map channels to 6 times of the input number, and then operating a depthwise convolution and another pointwise convolution. The depthwise convolution changes the width and height of the input feature maps (Yu et al., 2020). In the second pointwise convolution, the number of feature map channels is decreased to the input number (Buiu et al., 2020). Moreover, batch normalization (BN) layers are attached to each convolution layer to improve the model convergence speed and avoid gradient explosion (Liu and Wang, 2020). ReLU6 (rectified linear unit 6) non-linearity activation function increases the sparsity of the network and reduces the interdependence between parameters, thus being widely used after BN layers (Krizhevsky et al., 2012). However, the ReLU6 function will lead to significant information loss in the case of low-dimensional input (Wang et al., 2020). Therefore, the feature maps are directly passed to the next convolutional layer without using a RuLU6 after the second pointwise convolutional layer and the BN layer.

2.4.2. InceptionV3

InceptionV3 proposed by Google in 2015 can also be run on Android smartphones (Szegedy et al., 2016). It has three core modules, i.e., InceptionV3 module I (IM I), InceptionV3 module II (IM II) and InceptionV3 module III (IM III), as shown in Fig. 4. IM I, IM II, and IM III have the same operating principle, that is, increasing the depth and width of the network while leading to fewer computational cost. In the three modules, $n \times n$ convolution kernel is divided into two convolution kernels of $1 \times n$ and $n \times 1$, which greatly reduced network parameters and significantly improved detection speed (Shen et al., 2019; Xiao et al., 2020). The last layer of the three modules is Filter concat layer, consisting of an LRN (Local Response Normalization) layer and a Depthconcat layer, wherein the Depthconcat layer fuses the features extracted by convolutional layers. Finally, InceptionV3 uses an average pooling layer and a dropout layer before a fully connected layer, and add a softmax layer for the forward propagation to avoid vanishing gradient problem (Zhuang and Zhang, 2019; Zhao et al., 2020). Each convolutional layer is followed by a BN layer and a ReLU non-linearity.

2.5. Model quantization

Quantization is a process of constraining data values from a dense set (e.g. floating point numbers) to a relatively discrete set (e.g. integers) for optimizing models for smaller model size, faster inference speed and lower power consumption. In conventional CNN layers deployed in TensorFlow, there are a lot of weight tensor and activation function data that are stored as 32-bit floating point numbers (Zebin et al., 2019). In our experiment, an 8-bit quantization method was applied to accelerate and compress models simultaneously. Floating point data of 32-bit was quantized to 8-bit integer data that have less bit width for reducing the model size in a factor of 4. Per input channel and per layer quantization were used to quantize weight tensor and activation function data,

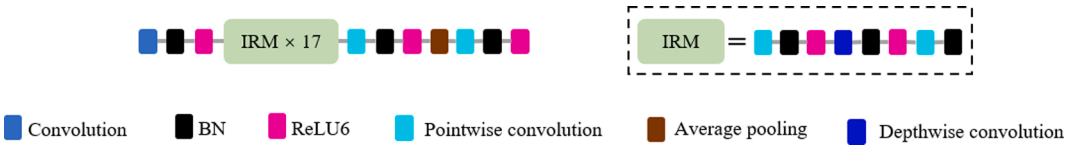

Fig. 3. Diagram of MobileNetV2, which includes 17 inverted residual module (IRM).

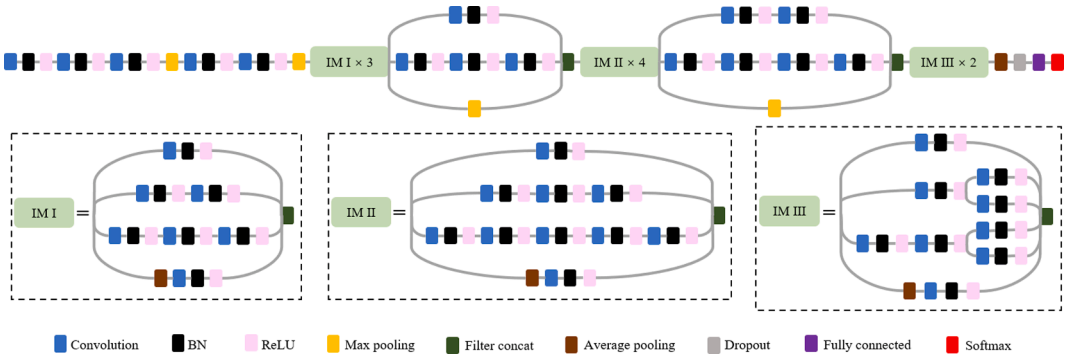

Fig. 4. Diagram of InceptionV3 and its three core parts, i.e., InceptionV3 module I (IM I), InceptionV3 module II (IM II), and InceptionV3 module III (IM III).

respectively.

2.6. Network training

The training hardware included a desktop computer equipped with a CPU of Intel Core i5 6400 (2.7 GHz), 16 GB of RAM, and an NVIDIA GTX 1080 8 GB GPU, running on a Windows 10 64-bit system. The software tools included CUDA 9.0, CUDNN 7.5, Python 3.6, and Microsoft Visual Studio 14.0. The experiments were implemented on the TensorFlow platform.

For a fair comparison, SSD with MobileNetV2, quantized MobileNetV2, Inception V3, and quantized InceptionV3 were trained with the same dataset and training parameters. BN was applied after every layer, and the standard weight decay was set to 0.00004. The learning rate decayed by 0.05 per epoch with the initial learning rate of 0.003, and a batch size of 12. Two hundred thousand iterations were set to analyze the training process.

Four TensorFlow checkpoint (ckpt) models were trained based on pre-trained models that have been trained the COCO (Common Objects in COntext) dataset to reduce the training time. TensorFlow provides a fine-tuning strategy to use the transfer learning to retrain the final layer of the feature extraction network for new objects (Majeed et al., 2020a; Zhang et al., 2020ab). In our experiment, the fine-tuning strategy was applied to decrease the time taken to train from scratch by taking a fully trained model for COCO dataset and retraining for new classes from existing weights.

2.7. Design for Android platform

2.7.1. TensorFlow Lite

TensorFlow Lite is a set of tools and designed to perform deep learning on smartphones, which enables on-device deep learning inference with low latency. It includes two components of TensorFlow Lite converter and TensorFlow Lite interpreter for APP development (Ye et al., 2020). Firstly, two scripts *summarize_graph and graph_transforms* of TensorFlow were used to transform the ckpt models to pb (protocol buffer) models. Secondly, TensorFlow Lite Converter was applied to convert the pb models to TensorFlow Lite models. Finally, TensorFlow Lite interpreter was configured to run the TensorFlow Lite models on smartphones and make use of hardware resources of smartphones to improve detection speed of these models further (Zebin et al., 2019).

2.7.2. APP development

The first kiwifruit detection APP is named as KiwiDetector and contains four modules: image acquisition module, image preprocessing module, fruit detection module, and result display module, as shown in Fig. 5. The image acquisition module obtains a kiwifruit image via smartphone camera or photo album. The image preprocessing module resizes the input kiwifruit image into a fixed size of 300 × 300 resolution, which can accelerate the detection speed and reduce the interference of the background to the result, so that the model can detect kiwifruits more efficiently. The fruit detection module contains the TensorFlow Lite interpreter and the TensorFlow Lite model. The results display module displays object class labels, confidence scores, and predicted bounding boxes of kiwifruits in the image. Additionally, object class labels and confidence scores have been hidden to display the detection results clearly. The APP development environment is based on the Windows 10 operating system and Java language and is configured by Android Studio 3.4.2, JDK 1.8.0, and JRE 1.8.0.

The interface of the kiwifruit detection APP is shown in Fig. 6. Its initial interface includes an image display unit, a camera button, and an album button. Users can take or choose a kiwifruit image by the two buttons. The kiwifruit detection result will be shown in the image display unit immediately.

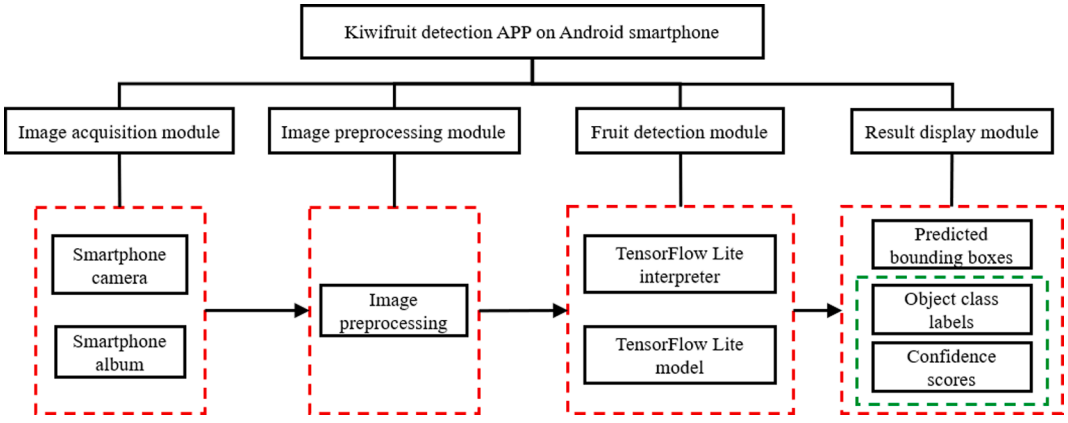

Fig. 5. The overall structure of the kiwifruit detection APP.

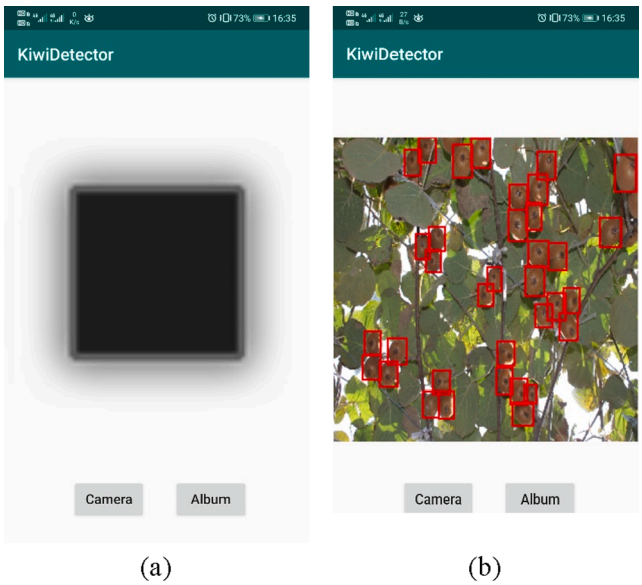

Fig. 6. Kiwifruit detection APP interface. (a) APP initial interface. (b) Detection result display interface.

2.8. Detection result categories

In this experiment, kiwifruit detection results consisted of true detected kiwifruits, missing detected kiwifruits, and false detected kiwifruits. Red rectangles without any other marks that generated by models were regarded as true detected kiwifruits, as shown in Fig. 7. There were three experienced yield estimation testers believed that such red rectangles could be used to count the number of kiwifruits correctly. The missing detected kiwifruits referred to those undetected kiwifruits, as shown by yellow circles in the Fig. 7a. The false detected kiwifruits were those detected kiwifruits but not true kiwifruits or detected repeated, including branches detected as kiwifruit (Fig. 7b), leaves detected as kiwifruit (Fig. 7c), and repeated detected kiwifruit (Fig. 7d).

2.9. Evaluation indicators

The performance of the four models is evaluated by loss-iteration curves, intersection over union (IoU), true detected rate (TDR), missing detected rate (MDR), false detected rate (FDR), model size, and detection speed.

IoU is an evaluation indicator for position accuracy. It measures how good are the real object boundary on an image and the prediction boundary generated by the detection model. In this paper, the IoU was set to 0.5 and defined in Eq. (1).

$$IoU = \frac{\text{area}(\text{truebbox} \cap \text{predictedbbox})}{\text{area}(\text{truebbox} \cup \text{predictedbbox})} \tag{1}$$

TDR, MDR, and FDR are indicators that evaluate the detection performance of the model on the smartphone. Manual counting was applied to assess the three indicators of the model on the Huawei P20 smartphone (Hisilicon Kirin 970 CPU and 6 GB of RAM). The formulas for calculating TDR, MDR, and FDR are as follows.

$$TDR = \frac{N_{td}}{N_{gt}} \tag{2}$$

$$MDR = \frac{N_{md}}{N_{gt}} \tag{3}$$

$$FDR = \frac{N_{fd}}{N_{ad}} \tag{4}$$

where N_{td}, N_{md}, N_{gt}, N_{fd}, and N_{ad} represent the number of true detected kiwifruits, missing detected kiwifruits, ground truth kiwifruits in the 100 selected original images, false detected kiwifruits, and all detected kiwifruits.

The model size and detection speed of the four deep learning models were also compared in this paper. The two indicators can intuitively reflect the performance of these models on smartphones.

3. Results and discussion

3.1. Training assessment of the four models

The performance of models is remarkably influenced by the times of iterations. Results of the training loss curves of SSD with MobileNetV2, quantized MobileNetV2, InceptionV3, and quantized InceptionV3 were shown in Fig. 8. The trend of curves of MobileNetV2 and quantized MobileNetV2 were basically the same. InceptionV3 and quantized InceptionV3 also had a similar curve tendency. It was obvious that the oscillation amplitude of curves of the quantized models was smaller than that of the unquantized models. MobileNetV2 and quantized MobileNetV2 had faster convergence speed and better convergence results than the other two models during training. The final loss of InceptionV3 and quantized InceptionV3 was around 4.8, while that of MobileNetV2 and quantized MobileNetV2 was about 2.1. The loss curves for MobileNetV2 and quantized MobileNetV2 began to stabilize after 48,000 iterations. However, the loss for InceptionV3 and quantized InceptionV3 was slowly decreasing. The results demonstrated that quantized MobileNetV2 had the potential to achieve an expected detection performance.

3.2. Validation of the four models on field images

3.2.1. Comparison of MobileNetV2 and InceptionV3

MobileNetV2 had a better TDR than InceptionV3. As shown in Table 1, the TDR of MobileNetV2 was the highest (90.8%), followed by

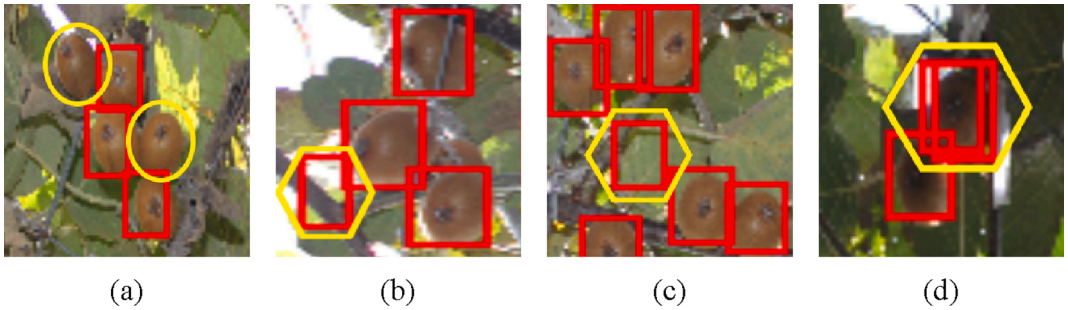

Fig. 7. Examples of true detected kiwifruits (red rectangles without any other marks), missing detected kiwifruits (yellow circle), and false detected kiwifruits (yellow hexagons) that includes branches detected as kiwifruit (b), leaves detected as kiwifruit (c), and repeated detected kiwifruit (d).

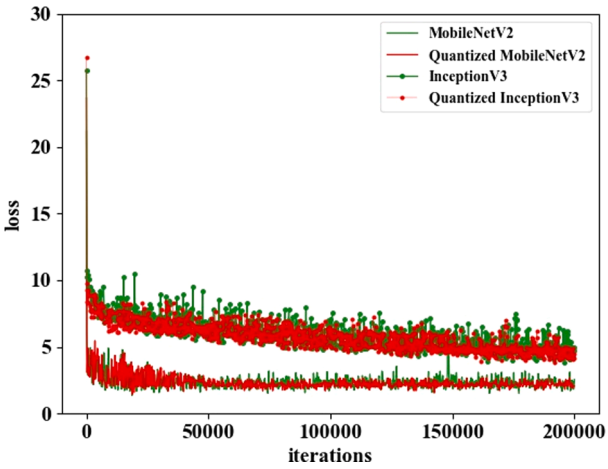

Fig. 8. Training loss curves of the four models.

Table 1
Results of kiwifruit detection with the four models.

	TDR (%)	MDR (%)	FDR (%)	Model size (MB)	Detection speed (ms/image)
MobileNetV2	90.8	9.2	3.5	17.5	163 ± 4.6^{b}
Quantized MobileNetV2	89.7	10.3	2.2	4.5	103 ± 1.1^{a}
InceptionV3	87.6	12.4	3.2	96.1	1085 ± 7.0^{d}
Quantized InceptionV3	72.8	27.2	1.9	24.1	685 ± 6.1^{c}

quantized MobileNetV2 (89.7%) and then InceptionV3 (87.6%). The lowest TDR of 72.8% was obtained by quantized InceptionV3. The kiwifruit detection results of the four models were shown pictorially in Fig. 9. The detection results in our study are similar to that on fruits classification (Xiang et al., 2019), which correctly classified 96.6% and 89.5% of fruits for MobileNetV2 and InceptionV3, respectively.

MobileNetV2 performed better than InceptionV3, especially when detecting images containing many kiwifruits. As shown in Table 1 and Fig. 9, MobileNetV2 and quantized MobileNetV2 obtained higher TDRs and better detection results than that of quantized InceptionV3. It can be observed from Fig. 9a, Fig. 9b, Fig. 9c, Fig. 9e, Fig. 9f, and Fig. 9g that there was no obvious difference in detection performance between MobileNetV2, quantized MobileNetV2, and InceptionV3. However, MobileNetV2 and quantized MobileNetV2 showed better detection results than InceptionV3 when detecting many kiwifruits, as shown in Fig. 9i, Fig. 9j, and Fig. 9k. As shown by the yellow circles (missing detected kiwifruits) of Fig. 9k, the detection results of InceptionV3 had certain adjacent and overlapped kiwifruit leakage phenomenon, while most kiwifruits in the same position were detected in Fig. 9i and Fig. 9j. The results are different from that of He et al. (2019) and Liu et al. (2019), which used MobileNetV1 and InceptionV3 to detect rape pests and classify plant diseases. In the two studies, MobileNetV1 showed a higher detection or classification performance than InceptionV3. The reason was that those two studies were working on simple images that normally contained only 1 to 7 objects, while the images used in our experiment contained at least 30 kiwifruits.

MobileNetV2 worked faster than InceptionV3 on smartphones. Deep learning needs lots of computation and resources, whereas smartphones are often resource-limited because of their small size. Small size models can reduce the cost of resources and improve the detection speed greatly. As shown in Table 1, the model size of MobileNetV2 (17.5 MB)

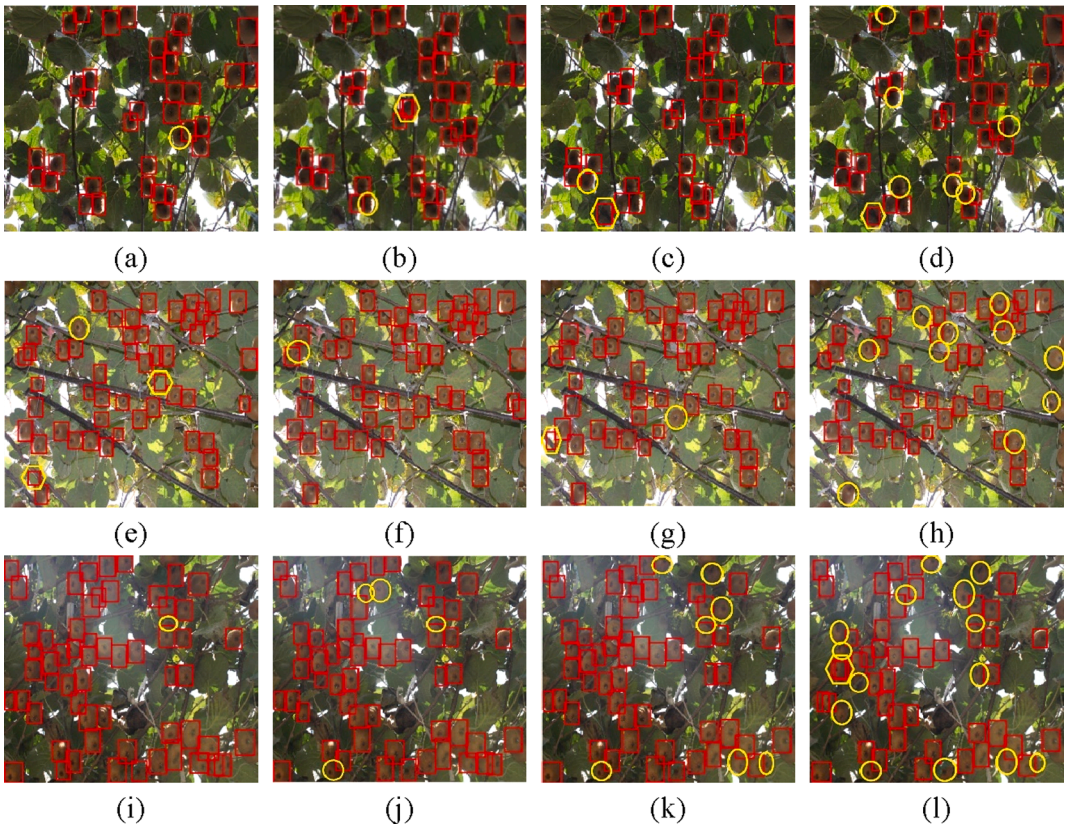

Fig. 9. Examples of kiwifruit images detected by SSD with MobilenetV2 (first column), quantized MobilenetV2 (second column), InceptionV3 (third column) and quantized InceptionV3 (fourth column). The red rectangles are referring to the true detected kiwifruits. While yellow circles and hexagons are manually marked and indicating the missing detected kiwifruits and false detected kiwifruits respectively.

was smaller than InceptionV3 (96.1 MB), while the detection speed of MobileNetV2 (163 ms/image) was faster than InceptionV3 (1085 ms/image). In the same trend, quantized MobileNetV2 and quantized InceptionV3 obtained the detection speeds of 103 ms/image and 685 ms/image with the model sizes of 4.5 MB and 24.1 MB. There was a significant difference in detection speed among MobileNetV2, quantized MobileNetV2, InceptionV3, and quantized InceptionV3. Similarly, Liu et al. (2019) developed an Android plant disease classification APP with MobileNetV1 and InceptionV3, which gained the detection speeds of 134 ms/image and 171 ms/image with the model sizes of 17.1 MB and 87.5 MB.

3.2.2. Comparison of quantized and unquantized models

Quantized models had faster detection speed and smaller model size than unquantized models. As shown in Table 1, quantized MobileNetV2 achieved a detection speed of 103 ms/image, which was faster than MobileNetV2 (163 ms/image). The model size of MobileNetV2 and quantized MobileNetV2 was 17.5 MB and 4.5 MB. InceptionV3 (1085 ms/image) and quantized InceptionV3 (685 ms/image) showed the same trend with the model sizes of 96.1 MB and 24.1 MB. The results are similar to Al-Hami et al. (2020), which used the 8-bit quantization method to quantify AlexNet and VGG16 (Visual Geometry Group with 16 layers). Compared with AlexNet and VGG16, the model size of quantized AlexNet and quantized VGG16 was compressed by 14% and 10%, respectively. Meanwhile, the inference speed of quantized AlexNet and quantized VGG16 increased by 20% and 16%.

Quantized MobileNetV2 performed better than MobileNetV2. As shown in Table 1, MobileNetV2 gained the TDR of 90.8% and the MDR of 9.2%, while quantized MobileNetV2 obtained the TDR of 89.7% and the MDR of 10.3%. It can be observed from Fig. 9 that the number of red rectangles (true detected kiwifruits) and yellow circles in the first column are almost same as that in the second column. Similar to Al-Hami et al. (2020), the detection rates of the quantized AlexNet and the quantized VGG16 were 1.2% lower than that of AlexNet and VGG16. Particularly, two kiwifruits overlapped were detected as one kiwifruit, which was the main reason leading to the decline in the TDR of MobileNetV2 and quantized MobileNetV2. This phenomenon demands a further research. MobileNetV2 obtained an FDR of 3.5%, while quantized MobileNetV2 gained a lower FDR of 2.2%. As shown by yellow hexagons (false detected kiwifruits) of Fig. 9b, the false detected kiwifruits of quantized MobileNetV2 was repeated detected kiwifruit. The false detected kiwifruits of MobileNetV2 were leaves detected as kiwifruit, as shown in Fig. 9e. Although the TDR of Quantized MobileNetV2 was lower than that of MobileNetV2, the detection speed of Quantized MobileNetV2 is 37% faster than that of MobileNetV2. Furthermore, there will not be a significant impact on yield estimation accuracy with such a decline of 1.1% in the TDR. Quantized MobileNetV2 is more suitable for yield estimation than MobileNetV2.

InceptionV3 performed better than quantized InceptionV3. As shown in Table 1, InceptionV3 obtained the MDR of 12.4%, which was 14.8% lower than quantized InceptionV3 (27.2%). There were many more yellow circles in the sample images in the fourth column than in the third column, as shown in Fig. 9. Similar to MobileNetV2 and quantized MobileNetV2, the case of two kiwifruits overlapped were detected as one kiwifruit was also a main factor causing the rise in the MDR of InceptionV3 and quantized InceptionV3. Quantized InceptionV3 had a lower FDR of 1.9% than InceptionV3 (3.2%). The reason was that the N_{ad} of quantized InceptionV3 was much smaller than InceptionV3 that resulted in reduction of the N_{fd}. As shown by yellow hexagons of Fig. 9c and Fig. 9g, InceptionV3 could falsely detected leaves and branches as kiwifruits. As shown in Fig. 9d and Fig. 9l, the false detected kiwifruits of quantized InceptionV3 included leaves detected as kiwifruit and repeated detected kiwifruit. In addition, the TDR of quantized InceptionV3 (72.8%) was lower than InceptionV3 (87.6%). The results indicated that quantized InceptionV3 will cause considerable errors in the results of yield estimation.

3.2.3. Results of other studies on kiwifruit detection

The detection speed of our models and three other state-of-the-art kiwifruit detection using LeNet, ZFNet, and VGG16 by Fu et al. (2018a), Fu et al. (2018b), and Liu et al. (2020) was concluded and analyzed. The KiwiDetector APP were tested on images with resolution of 3,968 × 2,976 pixels on a HUAWEI P20 smartphone. As shown in Table 1, quantized MobileNetV2 obtained a detection speed of 103 ms/image, followed by MobileNetV2 (163 ms/image), quantized InceptionV3 (685 ms/image), and InceptionV3 (1085 ms/image). Fu et al. (2018a) employed a Inter (R) Core (TM) i3 CPU, reporting processing of a kiwifruit image with 2,352 × 1,568 resolution in 8100 ms. Fu et al. (2018b) reported processing time using a GeForce GTX960M 4 GB GPU on a kiwifruit image with a resolution of 2,352 × 1,568 at 274 ms. Liu et al. (2020) reported average detection speed of 134 ms per image with a resolution of 512 × 424 on an NVIDIA TITAN XP 12 GB GPU. Obviously, our models can detect high-resolution kiwifruit images efficiently on devices with limited resources. Moreover, quantized MobileNetV2 has the fastest speed among the above models, and thus has the potential for further real-time yield estimation.

3.3. Further works on kiwifruit detection for real-time yield estimation

For this project, the ultimate objective of the kiwifruit detection APP is to realize real-time yield estimation in the orchard. In general, a processing rate of greater than 30 frames per second (less than 33 ms per image) is regarded as a real-time speed for yield estimation (Koirala et al., 2019). In the above experiments, the detection speed (103 ms/image) of quantized MobileNetV2 on HUAWEI P20 smartphone did not meet the real-time requirement. Therefore, a latest Redmi K30 Pro (Qualcomm Snapdragon 865 CPU and 8 GB of RAM) smartphone was used to detect kiwifruits, which reached a detection speed of 51 ms/image using quantized MobileNetV2. The results demonstrated that powerful smartphone hardwares can reduce the detection time of models. In particular, the capability of processing chip of smartphone is the most important factor affecting the detection speed (Fu et al., 2018c). If these models are tested on a smartphone with a better processing chip, the detection speed will be further accelerated. The performance of smartphones has been improving with the rapid development of science technology. Smartphones in the future will have better hardware conditions and more optimized systems, therefore, will further improve the detection speed and help to realize real-time yield estimation.

In the future, a kiwifruit counting function based on the 'Kiwi Detector' APP will be developed to accomplish real-time video yield estimation by using a more powerful Android smartphone. Counting kiwifruits in images will demand fruit growers to capture a lot of images, which will be time-consuming and reduce the convenience for automated yield estimation. A real-time video counting approach will be more suitable for automated yield estimation. An experimenter can walk in the orchard with a selfie stick fixed a smartphone below the kiwifruit canopy to shoot a video and simultaneously obtain counting results.

4. Conclusions

Kiwifruit yield estimation provides valuable information for planning harvest schedules, where accurate kiwifruit detection is fundamentally important. Deep learning can produce impressive and robust detection results. Furthermore, smartphones are becoming more and more popular, and a yield estimation system is very meaningful if it can be run on smartphones directly.

This study firstly developed an Android APP named KiwiDetector to detect kiwifruits for further yield estimation using SSD with MobileNetV2, quantized MobileNetV2, InceptionV3, and quantized InceptionV3. Detection test was performed on 100 selected kiwifruit field images with resolution of 3,968 × 2,976 pixels using the four models on a HUAWEI P20 smartphone. The results showed that MobileNetV2,

quantized MobileNetV2, InceptionV3, and quantized InceptionV3 reached TDRs of 90.8%, 89.7%, 87.6%, and 72.8%, respectively. The fastest detection speed of 103 ms/image was achieved by quantized MobileNetV2 with the smallest model size of 4.5 MB. The detection speed enables the APP to detect at near real-time. Based on the results, it can be concluded that the APP has the potential to achieve real-time yield estimation with high efficiency using quantized MobileNetV2.

The detection performance of the APP is still insufficient when encountering overlapped kiwifruits, which will be the future works. Additionally, the detection speed of the APP does not meet the requirements of real-time detection. In future research, some latest powerful smartphones will be used to accelerate the detection speed, thereby further realizing real-time video yield estimation.

CRediT authorship contribution statement

Zhongxian Zhou: Data curation, Investigation, Writing - original draft. **Zhenzhen Song:** Investigation, Methodology, Writing - review & editing. **Longsheng Fu:** Conceptualization, Data curation, Methodology, Supervision, Writing - review & editing. **Fangfang Gao:** Writing - review & editing. **Rui Li:** Methodology, Writing - review & editing. **Yongjie Cui:** Methodology, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was supported by the China Postdoctoral Science Foundation funded project (2019M663832); Fundamental Research Funds for the Central Universities of China (2452020170); National Natural Science Foundation of China (grant number 31971805); International Scientific and Technological Cooperation Foundation of Northwest A&F University (grant number A213021803).

References

Al-Hami, M., Pietron, M., Casas, R., Wielgosz, M., 2020. Methodologies of compressing a stable performance convolutional neural networks in image classification. Neural Process. Lett. 51, 105–127. https://doi.org/10.1007/s11063-019-10076-y.

Bargoti, S., Underwood, J.P., 2017. Image segmentation for fruit detection and yield estimation in apple orchards. J. F. Robot. 34, 1039–1060. https://doi.org/10.1002/rob.21699.

Buiu, C., Danaila, V., Raduta, C.N., 2020. MobileNetV2 ensemble for cervical precancerous lesions classification. Processes 8, 595. https://doi.org/10.3390/pr8050595.

Chen, Y., Lee, W.S., Gan, H., Peres, N., Fraisse, C., Zhang, Y., He, Y., 2019. Strawberry yield prediction based on a deep neural network using high-resolution aerial orthoimages. Remote Sens. 11, 1584. https://doi.org/10.3390/rs11131584.

Dorj, U.-O., Lee, M., Yun, S., 2017. An yield estimation in citrus orchards via fruit detection and counting using image processing. Comput. Electron. Agric. 140, 103–112. https://doi.org/10.1016/j.compag.2017.05.019.

Feng, A., Zhou, J., Vories, E.D., Sudduth, K.A., Zhang, M., 2020. Yield estimation in cotton using UAV-based multi-sensor imagery. Biosyst. Eng. 193, 101–114. https://doi.org/10.1016/j.biosystemseng.2020.02.014.

Fu, L., Feng, Y., Elkamil, T., Liu, Z., Li, R., Cui, Y., 2018a. Image recognition method of multi-cluster kiwifruit in field based on convolutional neural networks. Trans. Chinese Soc. Agric. Eng. 34, 205–211. https://doi.org/10.11975/j.issn.1002-6819.2018.02.028.

Fu, L., Feng, Y., Majeed, Y., Zhang, X., Zhang, J., Karkee, M., Zhang, Q., 2018b. Kiwifruit detection in field images using Faster R-CNN with ZFNet. IFAC-PapersOnLine 51, 45–50. https://doi.org/10.1016/j.ifacol.2018.08.059.

Fu, L., Feng, Y., Wu, J., Liu, Z, Gao, F., Majeed, Y., Al-Mallahi, A., Zhang, Q., Li, R., Cui, Y., 2020a. Fast and accurate detection of kiwifruit in orchard using improved YOLOv3-tiny model. Precis. Agric. https://doi.org/10.1007/s11119-020-09754-y.

Fu, L., Gao, F., Wu, J., Li, R., Karkee, M., Zhang, Q., 2020b. Application of consumer RGB-D cameras for fruit detection and localization in field: A critical review. Comput. Electron. Agric. 177, 105687. https://doi.org/10.1016/j.compag.2020.105687.

Fu, L., Liu, Z., Majeed, Y., Cui, Y., 2018c. Kiwifruit yield estimation using image processing by an Android mobile phone. IFAC-PapersOnLine. 51, 185–190. https://doi.org/10.1016/j.ifacol.2018.08.137.

Fu, L., Majeed, Y., Zhang, X., Karkee, M., Zhang, Q., 2020c. Faster R-CNN-based apple detection in dense-foliage fruiting-wall trees using RGB and depth features for robotic harvesting. Biosyst. Eng. 197, 245–256. https://doi.org/10.1016/j.biosystemseng.2020.07.007.

Fu, L., Tola, E., Al-Mallahi, A., Li, R., Cui, Y., 2019. A novel image processing algorithm to separate linearly clustered kiwifruits. Biosyst. Eng. 183, 184–195. https://doi.org/10.1016/j.biosystemseng.2019.04.024.

Gao, F., Fu, L., Zhang, X., Majeed, Y., Li, R., Karkee, M., Zhang, Q., 2020. Multi-class fruit-on-plant detection for apple in SNAP system using Faster R-CNN. Comput. Electron. Agric. 176, 105634. https://doi.org/10.1016/j.compag.2020.105634.

Hani, N., Roy, P., Isler, V., 2020. A comparative study of fruit detection and counting methods for yield mapping in apple orchards. J. F. Robot. 37, 263–282. https://doi.org/10.1002/rob.21902.

He, Y., Zeng, H., Fan, Y., Ji, S., Wu, J., 2019. Application of deep learning in integrated pest management: a real-time system for detection and diagnosis of oilseed rape pests. Mob. Inf. Syst. 2019, 4570808. https://doi.org/10.1155/2019/4570808.

Kestur, R., Meduri, A., Narasipura, O., 2019. MangoNet: a deep semantic segmentation architecture for a method to detect and count mangoes in an open orchard. Eng. Appl. Artif. Intell. 77, 59–69. https://doi.org/10.1016/j.engappai.2018.09.011.

Koirala, A., Walsh, K., Wang, Z., McCarthy, C., 2019. Deep learning - method overview and review of use for fruit detection and yield estimation. Comput. Electron. Agric. 162, 219–234. https://doi.org/10.1016/j.compag.2019.04.017.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. ImageNet classification with deep convolutional neural networks. In: 26th Annu. Conf. Neural Inf. Process. Syst. 2012, NIPS 2012. Sydney, Australia, pp. 1097–1105. Retrieved from https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Liu, J., Wang, X., 2020. Early recognition of tomato gray leaf spot disease based on MobileNetv2-YOLOv3 model. Plant Methods. 16, 83. https://doi.org/10.1186/s13007-020-00624-2.

Liu, Y., Feng, Q., Wang, S., 2019. Plant disease identification method based on lightweight CNN and mobile application. Trans. Chinese Soc. Agric. Eng. 35, 194–204. https://doi.org/10.11975/j.issn.1002-6819.2019.17.024.

Liu, Z., Wu, J., Fu, L., Majeed, Y., Feng, Y., Li, R., Cui, Y., 2020. Improved kiwifruit detection using pre-trained VGG16 with RGB and NIR information fusion. IEEE Access 8, 2327–2336. https://doi.org/10.1109/ACCESS.2019.2962513.

Lu, J., Hu, J., Zhao, G., Mei, F., Zhang, C., 2017. An in-field automatic wheat disease diagnosis system. Comput. Electron. Agric. 142, 369–379. https://doi.org/10.1016/j.compag.2017.09.012.

Majeed, Y., Karkee, M., Zhang, Q., Fu, L., Whiting, M.D., 2020a. Determining grapevine cordon shape for automated green shoot thinning using semantic segmentation-based deep learning networks. Comput. Electron. Agric. 171, 105308. https://doi.org/10.1016/j.compag.2020.105308.

Majeed, Y., Zhang, J., Zhang, X., Fu, L., Karkee, M., Zhang, Q., Whiting, M.D., 2020b. Deep learning based segmentation for automated training of apple trees on trellis wires. Comput. Electron. Agric. 170, 105277. https://doi.org/10.1016/j.compag.2020.105277.

Ramos, P.J., Prieto, F.A., Montoya, E.C., Oliveros, C.E., 2017. Automatic fruit count on coffee branches using computer vision. Comput. Electron. Agric. 137, 9–22. https://doi.org/10.1016/j.compag.2017.03.010.

Rawlani, H., Saita, J., Zambre, V., Priya, R.L., 2018. Deep Learning based approach to suggest recipes. In: 2018 Int. Conf. Smart City Emerg. Technol, ICSCET 2018. Mumbai, India. https://doi.org/10.1109/ICSCET.2018.8537350.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C., 2018. MobileNetV2: inverted residuals and linear bottlenecks. In: Proc. 2018 IEEE/CVF Conf. Comput. Vis. Pattern Recognit, CVPR 2018. Salt Lake City, UT, United States, pp. 4510–4520. https://doi.org/10.1109/CVPR.2018.00474.

Santos, T.T., de Souza, L.L., dos Santos, A.A., Avila, S., 2020. Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association. Comput. Electron. Agric. 170, 105247. https://doi.org/10.1016/j.compag.2020.105247.

Shen, Y., Yin, Y., Zhao, C., Li, B., Wang, J., Li, G., Zhang, Z., 2019. Image recognition method based on an improved convolutional neural network to detect impurities in wheat. IEEE Access 7, 162206–162218. https://doi.org/10.1109/ACCESS.2019.2946589.

Song, Y., Glasbey, C., Horgan, G., Polder, G., Dieleman, J., van der Heijden, G.W.A.M., 2014. Automatic fruit recognition and counting from multiple images. Biosyst. Eng. 118, 203–215. https://doi.org/10.1016/j.biosystemseng.2013.12.008.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., 2016. Rethinking the inception architecture for computer vision. In: Proc. 29th IEEE Conf. Comput. Vis. Pattern Recognit, CVPR 2016. Las Vegas, NV, USA, pp. 2818–2826. https://doi.org/10.1109/CVPR.2016.308.

UN Food & Agriculture Organization, 2020. Production of kiwi (fruit) by countries. Retrieved 2020-06-25.

Wang, S., Muhammad, K., Hong, J., Sangaiah, A.K., Zhang, Y., 2020. Alcoholism identification via convolutional neural network based on parametric ReLU, dropout, and batch normalization. Neural Comput. Appl. 32, 665–680. https://doi.org/10.1007/s00521-018-3924-0.

Xiang, Q., Wang, X., Li, R., 2019. Fruit image classification based on mobilenetv2 with transfer learning technique. In: Proc. 3rd Int. Conf. Comput. Sci. Appl. Eng, CSAE 2019. Sanya, China. https://doi.org/10.1145/3331453.3361658.

Xiao, D., Cai, J., Lin, S., Yang, Q., Xie, X., Guo, W., 2020. Grapefruit detection model based on ifssd convolution network. Trans. Chinese Soc. Agric. Mach. 51, 28–35 and 97. https://doi.org/10.6041/j.issn.1000-1298.2020.05.003.

Ye, J., Li, X., Zhang, X., Zhang, Q., Chen, W., 2020. Deep learning-based human activity real-time recognition for pedestrian navigation. Sensors 20, 2574. https://doi.org/10.3390/s20092574.

Yu, D., Xu, Q., Guo, H., Zhao, C., Lin, Y., Li, D., 2020. An efficient and lightweight convolutional neural network for remote sensing image scene classification. Sensors 20, 1999. https://doi.org/10.3390/s20071999.

Zebin, T., Scully, P.J., Peek, N., Casson, A.J., Ozanyan, K.B., 2019. Design and implementation of a convolutional neural network on an edge computing smartphone for human activity recognition. IEEE Access 7, 133509–133520. https://doi.org/10.1109/ACCESS.2019.2941836.

Zhang, M., Zhou, J., Sudduth, K.A., Kitchen, N.R., 2020a. Estimation of maize yield and effects of variable-rate nitrogen application using UAV-based RGB imagery. Biosyst. Eng. 189, 24–35. https://doi.org/10.1016/j.biosystemseng.2019.11.001.

Zhang, Z., Flores, P., Igathinathane, C., Naik, D.L., Kiran, R., Ransom, J.K., 2020b. Wheat lodging detection from UAS imagery using machine learning algorithms. Remote Sens. 12, 1835. https://doi.org/10.3390/rs12111838.

Zhao, Y., Zheng, Y., Shi, H., Zhang, L., 2020. Transfer learning-based convolutional neural network image recognition method for plant leaves. Int. J. Circuits. Syst. Signal Process. 14, 56–62. https://doi.org/10.46300/9106.2020.14.9.

Zhuang, X., Zhang, T., 2019. Detection of sick broilers by digital image processing and deep learning. Biosyst. Eng. 179, 106–116. https://doi.org/10.1016/j.biosystemseng.2019.01.003.