Computers and Electronics in Agriculture 168 (2020) 105108

Contents lists available at ScienceDirect

Computers and Electronics in Agriculture

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

Fast implementation of real-time fruit detection in apple orchards using deep learning

Hanwen Kang, Chao Chen*

Department of Mechanical and Aerospace Engineering, Monash University, Melbourne, Australia

ARTICLE INFO

Keywords:
Fruit detection
Deep learning
Real-time
Data labelling
Robotic harvesting

ABSTRACT

To perform robust and efficient fruit detection in orchards is challenging since there are a number of variances in the working environments. Recently, deep-learning have shown a promising performance in many visual-guided agriculture applications. However, deep-learning based approaches requires labelling on training data, which is a labour-intensive and time-consuming task. In this study, a fast implementation framework of a deep-learning based fruit detector for apple harvesting is developed. The developed framework comprises an auto label generation module and a deep-learning-based fruit detector 'LedNet'. The Label Generation algorithm utilises the multi-scale pyramid and clustering classifier to assist fast labelling of training data. LedNet adopts feature pyramid network and atrous spatial pyramid pooling to improve the detection performance of the model. A light-weight backbone is also developed and utilised to improve computational efficiency. From the experimental results, LedNet achieves 0.821 and 0.853 on recall and accuracy on apple detection in orchards, and its weights size and inference time are 7.4 M and 28 ms, respectively. The experimental results show that LedNet can perform real-time apple detection in orchards robustly and efficiently.

1. Introduction

Robotic fruits harvesting is one of the most challenging task in the automatic agriculture (Zhao et al., 2016). A typical fruit-harvesting robot comprises two subsystems: a vision system and manipulator system (Lehnert et al., 2016). The vision system detects and localises fruits and guides the manipulator to detach fruits from trees. However, a robust and efficient fruit detection algorithm in orchards is challenging as there are many variances such as illumination changing and occlusion between fruits, branches and leaves. Previous studies (Hashimoto, 2003; Kapach et al., 2012) have pointed out that a robust and efficient vision system is the key to the success of the robotic fruits harvesting.

In recent years, deep-learning has become state of the art in many tasks within computer vision, including image classification (He et al., 2016), segmentation (Wang et al., 2018), and object detection (Redmon and Farhadi, 2017). Compared to the traditional machine-learning approaches, deep-learning has strong adaptability to variances within the working scene (Kamilaris and Prenafeta-Boldu, 2018), making it a promising approach in many vision tasks. Deep-learning based object detection can be classified into two classes (Lin et al., 2017): two-stage detector and one-stage detector. The representative of the two-stage detectors is the Region Convolution Neural Network (RCNN), including RCNN (Girshick et al., 2014), Fast/Faster RCNN (Ren et al., 2015), and Mask RCNN (He et al., 2017). A RCNN model has two network branches: a Region Propose Network (RPN) branch and a classification branch. RPN proposes the Region Of Interest (ROI) of foreground class, while the classification branch classifies and estimates boundary box for each ROI. Compared to the traditional ROI searching strategies such as exhaust searching and selective searching (Uijlings et al., 2013), RPN makes the ROI searching a trainable task, improving the performance and computational efficiency of the model. The one-stage detector was developed more recently than the two-stage detector. It combines the RPN branch and classification branch into a single network, leading to more concise architecture and better computational efficiency. You Only Look Once (YOLO) (Redmon et al., 2016; Redmon and Farhadi, 2017, 2018) is one of the most representative aspects of the one-stage detector, it achieves state of the art performance in object detection with high computation speed. One-stage detector has been applied in many vision-based robotic tasks, such as automatic driving (Laroca et al., 2018), UAV monitoring (Tijtgat et al., 2017), and automation agriculture (Zhong et al., 2018).

Deep-learning based detection algorithms are gradually applied in sensing in agriculture environment. The authors of Sa et al. (2016) adopted the Faster-RCNN on the detection of multiple class fruits, including apple, sweet pepper, and melon. Faster-RCNN achieved a better

* Corresponding author.
E-mail addresses: hanwen.kang@monash.edu (H. Kang), chao.chen@monash.edu (C. Chen).

https://doi.org/10.1016/j.compag.2019.105108
Received 29 July 2019; Received in revised form 12 November 2019; Accepted 12 November 2019
Available online 27 November 2019

detection performance and running speed compared to the previous work (McCool et al., 2016). In Hao et al. (2016), the authors adopted the graph-based segmentation and Neural Network (NN) model to perform the maize tassel segmentation. In Bargoti and Underwood (2017), the authors utilised a convolution neural network to perform semantic segmentation of the apple fruit, then a Watershed Segmentation (WS) and Circular Hough Transform (CHT) algorithms are used to detect the apple. Later, the authors of Bargoti and Underwood (2017) utilised the Faster-RCNN model to perform the in-field fruit detection, and they achieved the detection of apple and mangos with F_1 score higher than 0.9. In Li et al. (2017), the authors utilised the Fully Convolution Network (FCN) to perform the semantic segmentation of cotton, and their results showed that FCN outperformed the traditional segmentation algorithms. The authors of Yang et al. (2019) developed an FPN strengthened Mask-RCNN in the strawberry detection under a non-structured environment, the good results in both detection and instance segmentation tasks are shown from their work. The authors of Majeed et al. (2018) adopted the segnet (Badrinarayanan et al., 2017) to perform the segmentation of the apple branch in the orchard, an accuracy of 0.93 on branch segmentation was reported in their work. Lin et al. (2019) used the FCN (Long et al., 2015) to perform the semantic segmentation of the guava fruit and branch, then estimating the grasping posture of the fruit based on the segmentation. Their result generated better results compared to the traditional vision algorithms. In Tian et al. (2019), a customized YOLO-V3 network was applied to apple detection. The authors modified the YOLO-V3 by using the DenseNet (Huang et al., 2017) to enhance the feature extraction ability, and the designed one-stage detector outperformed Faster-RCNN and original YOLO-V3 in their work. In Koirala et al. (2019), the authors adopted the YOLO architecture in the yield estimation of mango fruit, accurate detection performance was reported from their work. In addition, deep-learning is also being applied in many agriculture applications(Kamilaris and Prenafeta-Boldu, 2018), such as yield estimation using remote sensing (Kussul et al., 2017) and crop monitoring using UAV (Yang et al., 2019).

The deep-learning based algorithms rely on backward propagation to train its parameters (Srivastava et al., 2015). With a proper architecture and training mechanism, the deep-learning model can fit the training data in good approximation and generalization. However, a significant difference between working scene and training data may lead to poor performance of the trained model (Shin et al., 2016). Transfer learning, which applies field data to adjust the pre-trained network to fit a specific task, is widely used in many applications (Weiss and Khoshgoftaar, 2016). However, labelling on training data is a time-consuming and labour-intensive task (Papandreou et al., 2015). Therefore, self-labelled algorithms which can programmatically generate the label on the training data has became an important issue in the implementation of the deep-learning approaches(Ratner et al., 2017).

In this study, a fast implementation framework of deep-learning based fruit detector is developed. This framework includes two components: an auto label generation module and a one-stage detector LedNet. The auto label generation module is used to accelerate the labelling on training data, the LedNet is used to perform the real-time detection of fruits in orchards. The pipeline of the framework is shown in Fig. 1. The auto label generation is achieved by a Clustering-RCNN (C-RCNN) algorithm to perform the quasi-good label prediction. The LedNet utilises the Feature Pyramid Network (FPN) and Atrous Spatial Pyramid Pooling (ASPP) to enhance the detection performance. Meanwhile, to improve computation efficiency of the model, a light-weight backbone is developed in this work.

The rest of the paper is organised as follows. Section 2 and Section 3 introduce the label generation method and detector LedNet, respectively. Section 4 discusses the experiments result. Finally, the conclusions and future work are presented.

2. Auto label generation

C-RCNN adopts the principle of the RCNN, separating the detection task into ROI proposal and classification/regression, which are introduced in Section 2.1 and 2.2, respectively.

2.1. Selective searching for ROI proposing

Segmentation on Multi-level Pyramid: Input image is scaled to $\frac{1}{4}$, $\frac{1}{8}$ and $\frac{1}{16}$ to form the multi-level pyramid in terms of improving the detection performance of objects in different scales. Colour Coherence Vector (CCV) (Pass et al., 1996) and Histogram of Gradient (HoG) (Ng and Henikoff, 2003) are utilised to encode colour and geometry features of objects, which are denoted as V_{ccv} and V_{hog}, respectively. In the forward inference, the concatenated feature vector $V_o = [V_{ccv}, V_{hog}]$ of length N is calculated, producing a $H \times W \times N$ feature map on each pyramid level. H and W are the width and height of feature maps, respectively. Assuming there are m-classes $C = [c_1, c_2, ..., c_m]$ and the probability of a feature vector V_o belongs to the class i is donated as $p(V_o|c_1)$. The classifier assigns the V_o to the class i with the highest probability, as follow:

$$\textbf{index}\ i = argmax(p(V_o|c_1), p(V_o|c_2), ..., p(V_o|c_m)) \tag{1}$$

Considering there are multiple distribution kernels within each object class, the Gaussian Mixture Models (GMM) is utilised to model the internal distribution within each class. Assuming there are k-number distributions $D = [d_1, d_2, ..., d_k]$ in the class C_i, the probability that a feature vector V_o belongs to the class C_i is:

$$p(V_o|C_i) = max(p(V_o|d_1), p(V_o|d_2), ..., p(V_o|d_k)) \tag{2}$$

The training of GMM follows the work (Greenspan et al., 2001).

Centre Detection: (1) and (2) are used to perform segmentation on feature maps of each pyramid level. Fruits are assigned as foreground class, other objects are assigned as background classes. Then, the centre detection algorithm is utilised to search the centre of each ROI. Firstly, pixel-connection is used to segment the foreground pixel into Independent Candidate Patch (ICP). Each pyramid level accepts ICP with an acceptable number of foreground pixels (for example, 250–2000). To deal with fruits under occlusion conditions, the CHT is utilised to perform centre detection within each ICP. The workflow of the centre detection is shown in Fig. 2. Finally, a pre-set anchor box is assigned to each centre as output ROIs.

2.2. ROI Post-processing

ROI Classification: Since clustering-based classifier has limited accuracy in the object classification. To increase the accuracy of proposed ROIs, a modified resnet network is utilised, which is introduced in Section 3.2. A resize operation on proposed ROIs is performed before classification. Then, the ROI which is classified as not belong to the fruit class will be deleted from the ROI list.

ROI Regression: boundary box regression can be recast as a template matching, which is expressed in (3). That is, given a template and flowing mask A and B. The template mask A is the average appearance of the apple. K stands the Normalised Cross-Correlation (NCC) value (Yoo and Han, 2009) between template and flowing masks. Eq. (3) estimates the scale S, rotation matrix R, and offset T to perform the matching.

$$\max\{K\}, \quad K = NCC(A, S * (R * B + T)) \tag{3}$$

R is a 2×2 identity matrix as no rotation applied in the matching, while the NCC is expressed as follows:

$$NCC(f(x,y), t(x,y)) = \frac{\sum_{x,y}[f(x,y) - \bar{f}][t(x,y) - \bar{t}]}{\sqrt{\sum_{x,y}[f(x,y) - \bar{f}]^2 \sum_{x,y}[t(x,y) - \bar{t}]^2}} \tag{4}$$

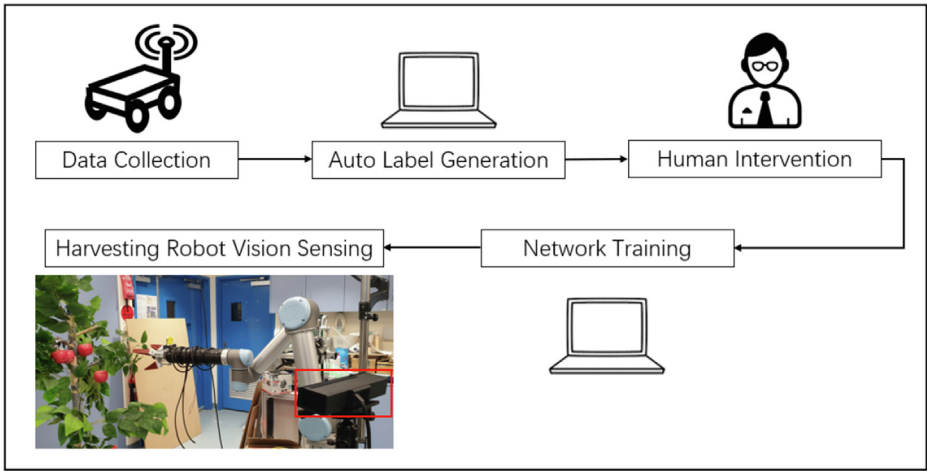

Fig. 1. Pipeline of the framework, from data collection, auto label generation to the detector training.

In the (4), $f(x, y)$ and $t(x, y)$ the template and flowing mask are shown. $\bar{f}$ and $\bar{t}$ are the mean of the $f(x, y)$ and $t(x, y)$, respectively. Eq. (3) is solved by local searching of the S and T to maximise the NCC between the template and mask. The NCC value between the template and flowing masks is returned as the confidence score for each ROI.

3. LedNet: network model

3.1. Network architecture

Multi-level Feature Fusion: Compared to the YOLO-V1 (Redmon et al., 2016) and YOLO-V2 (Redmon and Farhadi, 2017), YOLO-V3 (Redmon and Farhadi, 2018) improves the detection performance on objects of different scales by using FPN. Different levels of network always comprise different information. For example, feature maps on a lower level and higher level of the network comprise spatial and semantic information of objects, respectively. FPN can fuse the features of objects from different levels of the network to improve the feature extraction ability of the model on detection. LedNet utilises a three-level FPN to process feature maps from C3 (1/8), C4 (1/16) and C5 (1/32) of the backbone, which is shown in Fig. 3. Feature maps from different levels are fused by using an adding operation. Each level of FPN is used to detect objects within a specific scale range. For example, C5 level and C3 level of the LedNet detect objects in large scale and small scale, respectively. Each level of FPN has two subnets for boundary box regression and classification. The design of regression subnet follows the work in (Redmon and Farhadi, 2018) and two anchor-boxes are utilised on each level of FPN. (see Figs. 4–6).

Multi-scale Feature Fusion: One difference between the one-stage detector and the two-stage detector is the way of encoding features of ROIs. The two-stage detector uses the corresponding area of ROIs on feature maps to perform the classification and boundary box estimation. The one-stage detector encodes features of ROIs by using the fixed size convolution kernel. However, such processing cannot properly cover the corresponding area of ROIs, as the fixed size convolution kernel may over-cover or under-cover the area of ROIs. LedNet utilises the ASPP to encode the properly area of ROIs for the following classification and boundary box estimation. The ASPP is developed and utilised in GoogLeNet (Szegedy et al., 2015) for classification and DeepLab (Szegedy et al., 2016) for segmentation. The principle of the ASPP is to use dilation convolution kernel with different rate to encode the multi-scale features into feature maps. In the LedNet, an ASPP which includes a 1x1 convolution kernel, three 3x3 dilation convolutions (dilation rates are 1,3,6), and a 3 x 3 max-pooling operation is applied. Three dilation convolution kernels are used to cover the possible area of ROIs on feature maps. The max-pooling layer in the ASPP block aims to amplify the signal of small objects, to increase the recall of such objects in the detection. From the experiment, max-pooling operation may also introduce noise in the detection. Hence, a 3 x 3 convolution kernel is applied after max-pooling layer to filter such noises.

Backbone: LedNet can use different classification networks as a backbone, such as resnet and Darknet. To improve computation efficiency of the model on the embedded computing device, a light-weight backbone is developed, which is included in the following section.

3.2. Light-weight backbone

The light-weight backbone LW-net has nine bottleneck resnet blocks and five down-sampling blocks. Both the resnet block and down-sampling block adopt the residual shortcut design. The shortcut of the

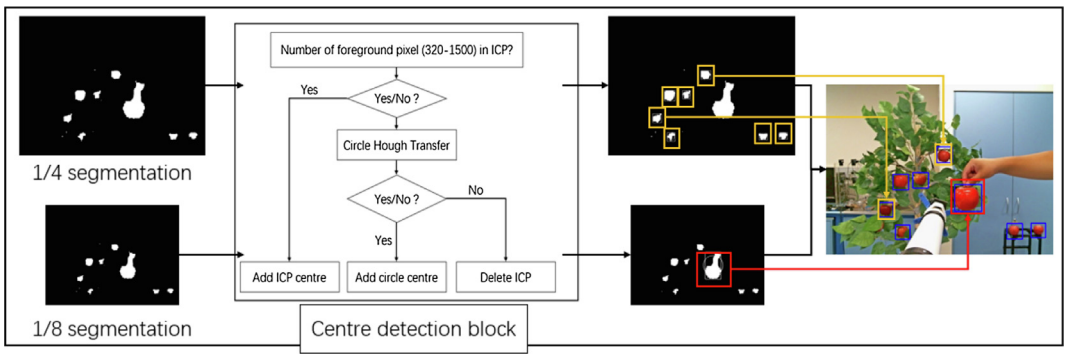

Fig. 2. centre detection algorithm for detecting extracting the potential centre from the segmentation. Yellow box is detected from 1/4 scale segmentation, while the red box is from the 1/8 segmentation. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

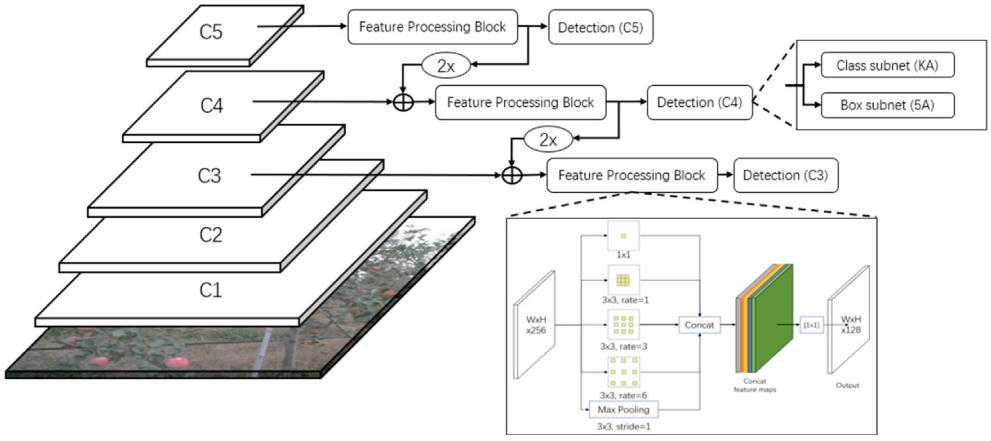

Fig. 3. The architecture of the LedNet, it utilises the 3-levels FPN and ASPP is used in the feature processing block. A and K are the number of pre-set anchor-box on each pyramid and object classes, respectively.

resnet block pass the information of the input feature map, while the shortcut of the down-sampling blocks pass the information of the feature tensor which is processed by the max-pooling of the input feature maps. The stride of the second convolution layer in down-sampling block is 2 to keep the consistency of the size of the feature map. The down-sampling block is used to replace the max-pooling layer to minimise the information loss during down-sampling. The LW-net is pre-trained with Cifar-10/100. The total weights size of the LW-net is 3.46 MB, and the validation accuracy of the model on the Cifar-10 and Cifar-100 is 92.7% and 71.6%, respectively.

3.3. Data processing and training

3.3.1. Data acquisition

800 images are collected from the orchard at Qingdao, China by using Kinect-v2. Another 300 images of apples in different scenes are collected to increase the diversity of the training data. In addition, 100 images of scenes without apples are also included in the training data. In total, 800 images are used as training data, while the remaining images are used for validation.

3.3.2. Data Augmentation

The distance between camera to apple trees is between 0.5 and 1.5 m. The average size of Fuji apples are between 80 mm and 100 mm. Therefore, most of the apples are presented as small objects in the training data. To avoid under-fitting of the model leaded by imbalance distribution of object scale, a objects amplification algorithm is utilised. Firstly, a patch (round 160–480 pixels) is cropped from the image and resized to the training resolution under possibility of 0.5. Then, this image will take another possibility of 0.5 to repeat the previous procedure. The distribution of object scale in training data before and after augmentation is shown in Table 1. The training resolution is 320 × 320, to increase number of images in each training batch. Other augmentation methods, including saturation, brightness, contrast, rotation, and flip are also utilised.

3.3.3. Focal-loss training

The training loss of LedNet includes three terms: confidence score, boundary box regression, and classification. The boundary box regression and classification follow the same equation which are used in the YOLO-v3, while the training of the confidence score utilise the focal-loss and MSE in the training (for a confidence score of objects of the foreground-class and background-class, respectively), as follows:

$$Loss_{obj} = \sum^{obj} -\alpha(1-p_t)^{\gamma} log(p_t) + \sum^{noobj} \beta(p_t)^2 \quad (5)$$

p_t and γ are confidence score and focusing parameter, respectively. α and β are the weights of the foreground-class term and background-class term, respectively. The distribution of the loss value of Mean

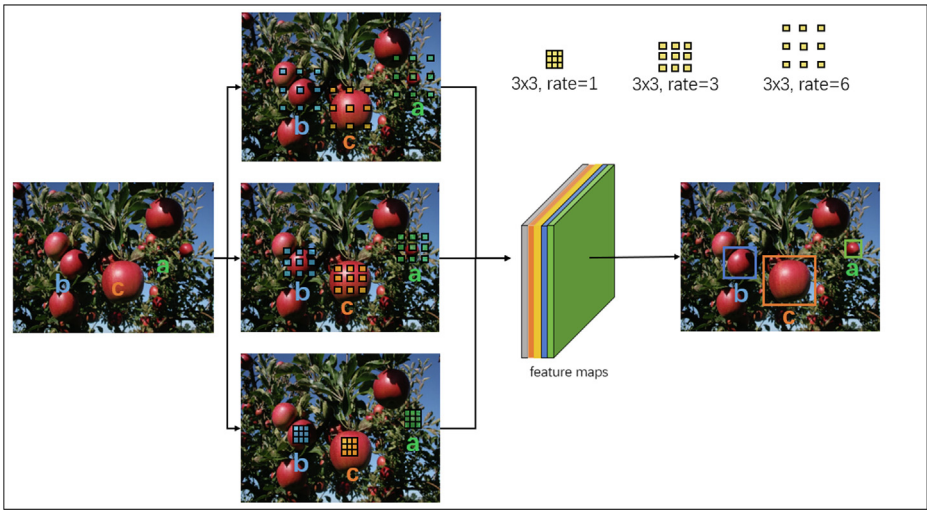

Fig. 4. Demonstration of ASPP of multi-scale feature fusion. Dilation convolution kernels with different rate are used to cover the ROI region to improve the feature extraction ability.

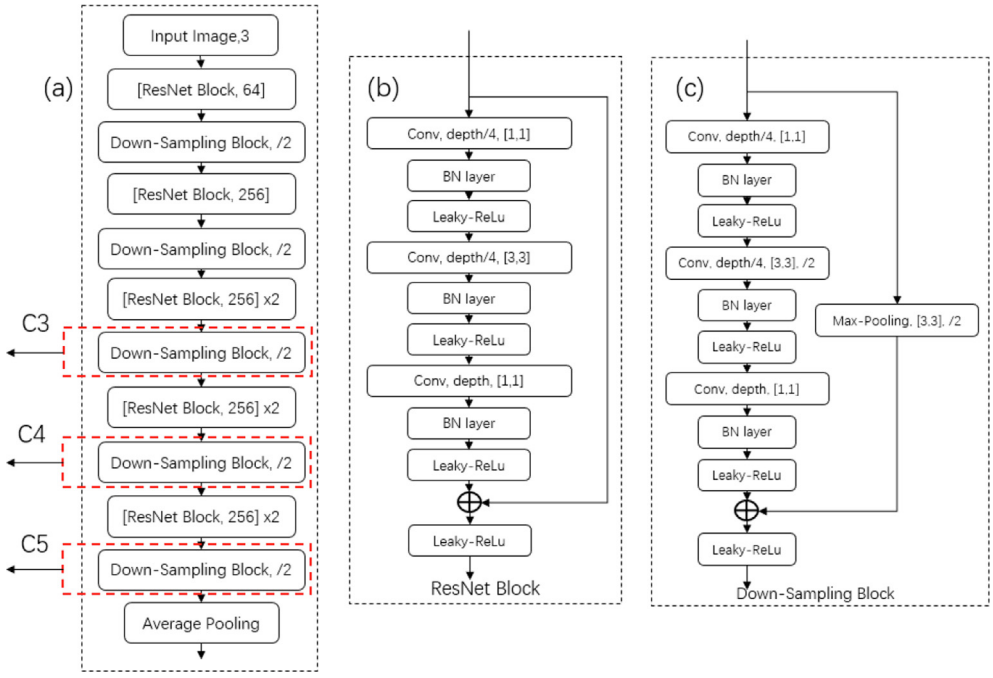

Fig. 5. Architecture of LW-net (a), it has 9 resnet blocks (b) and 5 down-sampling blocks (c).

Square Error (MSE) and focal-loss with different value of γ along the p_t are shown in Fig. 7. For foreground-class objects which have a predicted value of p_t larger than 0.5 are well-classified samples. In this region, focal loss has similar loss value compared to the MSE. When foreground-class objects are in the under-classified region ($p_t < 0.5$), focal-loss provides a much larger loss compared to the MSE. MSE is utilised to train the confidence score of background-class objects, as extremely imbalance between the number of foreground-class objects and background-class objects is presented in the training of the one-stage detector. α, β and γ are set as 1,0.5 and 2 based on the experiment results. The Adam-optimiser is applied and the learning rate and decay rate of optimiser are 0.001 and 0.9/epoch, respectively.

Table 1
Distribution of different scale of object in the training dataset.

Iteration	Augmentation	Small	Median	Large
150 epochs	Yes	41%	38%	21%
150 epochs	No	62%	30%	8%

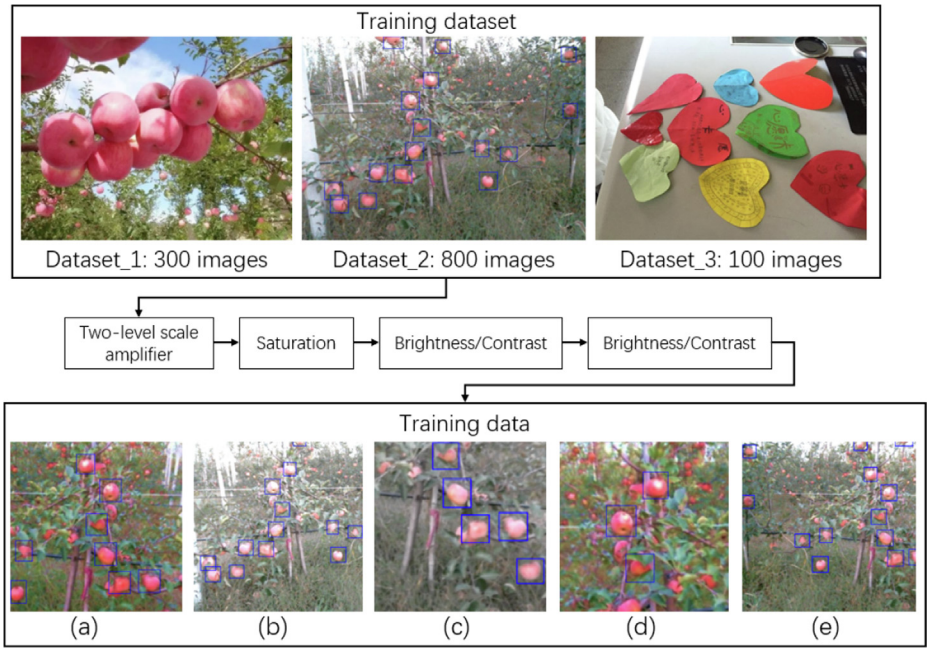

Fig. 6. Training data is comprised by dataset-1 (supplement data), dataset-2 (orchard data), and dataset-3 (other scenes). The data augmentation results are shown in (a)–(e).

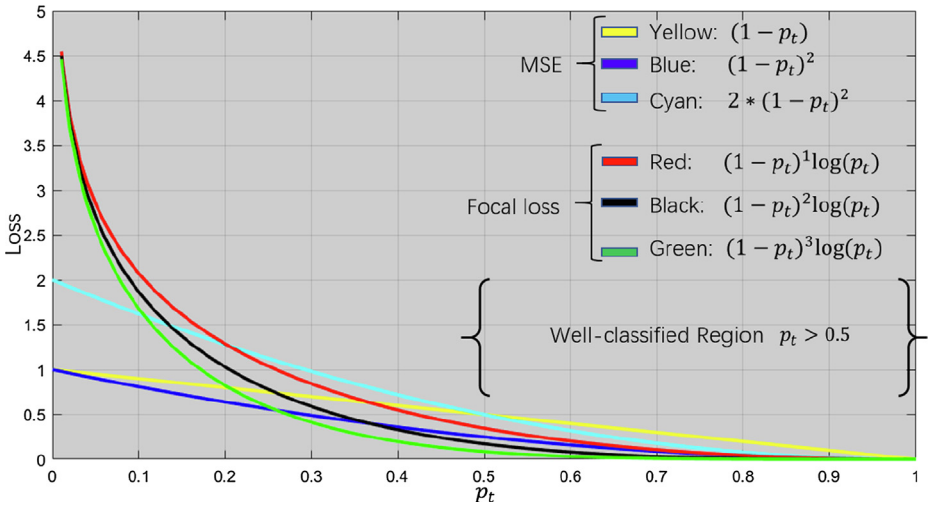

Fig. 7. Loss value of focal loss function and MSE function along the object confidence score p_t.

4. Experiment and discussion

4.1. Evaluation methods

In the experiment, AP is applied as the evaluation index. The AP evaluation includes several important indexes: Intersection of Union (IoU), *Precision*, and *Recall*. IoU calculates the overlap ratio between the boundary box of the prediction (*pred*) and ground-truth (*gt*). *Precision* measures the accuracy of the prediction, while the *Recall* measures how good the detector finds all *gt*. The formulation of the above three indexes is as follows.

$$IoU = \frac{Area_{pred} \cap Area_{gt}}{Area_{pred} \cup Area_{gt}} \tag{6}$$

$$Precision = \frac{TruePositive(TP)}{TruePositive(TP) + FalsePositive(FP)} \tag{7}$$

$$Recall = \frac{TruePositive(TP)}{TruePositive(TP) + FalseNegative(FN)} \tag{8}$$

The Precision-Recall curve forms the $P - R$ curve and the Area Under Curve (AUC) is the *AP* value. The *m* in the AP_m is stand the threshold(%) of IoU value between *pred* and *gt*. Another commonly used evaluation index F_1 score is also used in the evaluation, as formulated below.

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{9}$$

4.2. Experiment on auto label generation

4.2.1. Implementation detail

C-RCNN algorithm requires sampling to train the classifier for segmentation on the multi-level pyramid for ROI proposal. The foreground objects include apple, leaf, branch and land, and the threshold for being background is 0.3. That is, the region will be classified as background when the possibility of this region belonging to any foreground objects is lower than 0.3. The data sampling is to choose a pixel within a foreground objects. Then, the corresponding feature vector of the neighbour region (for example, 48 x 48) of this pixel will be calculated and saved for the training. Each class of foreground object requires 50 to 100 samples to train the classifier.

4.2.2. Performance evaluation

150 images are randomly selected to perform the evaluation of the C-RCNN algorithm. The evaluation results are shown in Table 2 and Fig. 8.

(a) to (c) of Figs. 8 show the labels generated by the C-RCNN on the orchard data. variances of illumination conditions and fruit appearances are the major factors that affect the performance of the label generation. Uneven-light minimising and colour enhancement measurements are utilised to process the images. On the orchard images, C-RCNN achieves 73% and 66% on the recall and accuracy, respectively. The average IoU of the generated label is 69%. (d) to (f) of Figs. 8 show the labels generated by the C-RCNN on the supplement data. There are many human-made objects in the supplement images data which cannot be fully included in the training. Therefore, to perform label generation on those image data is more complicated than the case on orchard data. In this condition, there is a considerable degeneration on the performance of the C-RCNN algorithm in the label generation. The recall, accuracy and average IoU of the generated labels are 68%, 62% and 63%, respectively. The computational time of the C-RCNN on each image is between 0.5 and 1.8s, depending on numbers of apples in images. The task with the highest computational consumption is the segmentation on the multi-level pyramid for ROI proposal, which takes 0.4–1.4 s for processing.

Table 2
Performance evaluation of Auto Label Generation.

Dataset	F_1	Recall	Accuracy	IoU	Mean time
Orchard	0.68	73%	66%	69%	0.8–1.6 s
Supplement	0.63	68%	62%	63%	0.5–1.8 s

From the experimental results, C-RCNN can perform well in the orchard data, but its performance shows a degeneration when applied in the supplement data. Some approaches can be utilised to improve the performance of the C-RCNN, such as including more objects in training and applying more feature descriptors in the segmentation. However, such measurements would increase the complexity of the algorithm. The average number of apples within an image in the training data is between 10 and 25. Therefore, the total number of apples in the training data is between 10,000 and 25,000. Manual labelling of all apples in the training data is labour-intensive and time-consuming. With the assistance of the C-RCNN algorithm, labelling of training data is accomplished within two days.

4.3. Self comparison on LedNet

Influence on Data Augmentation: Data augmentation is important in network training. Two different data augmentation methods are utilised to evaluate the influence of the data augmentation to the detection performance. The first method 'method-A' is introduced in Section 3.3.2, it utilises two-level scale amplification to balance the

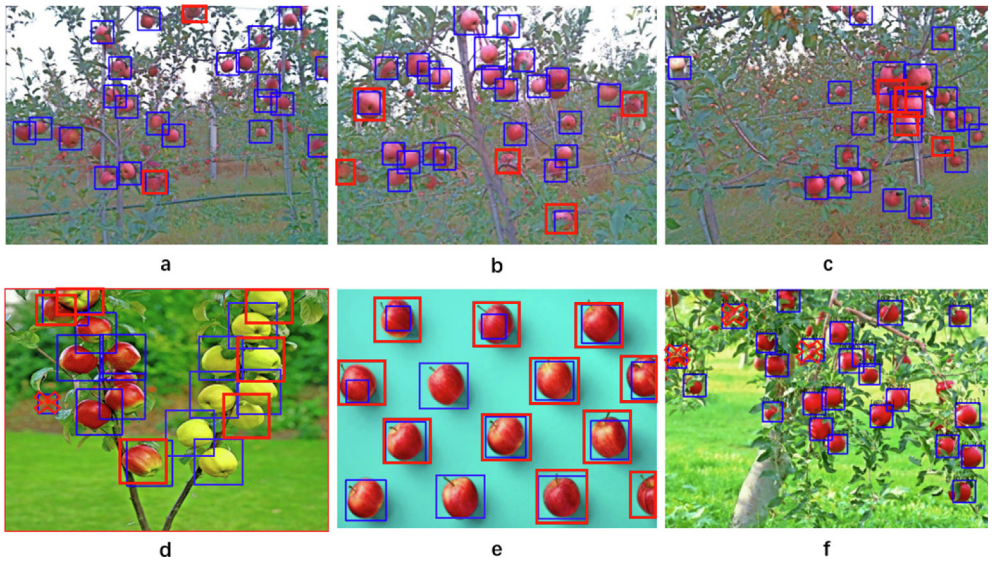

Fig. 8. Label prediction result using C-RCNN algorithm. Blue boxes are detected by the C-RCNN algorithm, red boxes and cross are the examples of adjustment of boundary-box and deletion after manual revision. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 3
Evaluation on different augmentation methods.

Methods	AP_{50}	AP_{small}	AP_{median}	AP_{large}	IoU
method-A	0.826	0.832	0.817	0.763	86.7%
method-B	0.797	0.818	0.778	0.652	78.3%

Table 4
Evaluation on different loss functions.

Loss function	AP_{50}	$F1$	Recall	Accuracy	IoU
Focal loss	0.826	0.832	0.82	0.85	86.3%
MSE	0.811	0.816	0.817	0.831	85.4%

Table 5
Evaluation on different backbones.

Models	AP_{50}	$F1$	Recall	Accuracy	IoU
LedNet (LW-net)	0.826	0.834	0.821	0.853	86.3%
LedNet (resnet-50)	0.834	0.84	0.833	0.854	86.4%
LedNet (resnet-101)	0.843	0.849	0.841	0.864	87.2%
LedNet (darknet-53)	0.833	0.842	0.83	0.857	86.3%

Table 6
Computation time and weights size of the LedNet with different backbones (on GTX-1080Ti).

Model	Inference time	weights size
LedNet (LW-net)	28 ms	7.4 M
LedNet (resnet-50)	38 ms	112 M
LedNet (resnet-101)	46 ms	188 M
LedNet (darknet-53)	36 ms	176 M

Table 7
Detection performance comparison between different detectors.

Model	AP_{50}	F_1	Recall	Accuracy	IoU
LedNet (LW-net)	0.826	0.834	0.821	0.853	86.3%
LedNet (resnet-101)	0.843	0.849	0.841	0.864	87.2%
YOLO-V3	0.803	0.803	0.801	0.82	84.2%
YOLO-V3 (Tiny)	0.782	0.783	0.776	0.796	82.4%
Faster-RCNN (VGG)	0.814	0.818	0.814	0.835	86.3%

distribution of object scale in training data. The second method 'method-B' utilises usual data augmentation method to train the network. The performance of LedNet by using different data augmentation methods is shown in Table 3.

The performance of the LedNet trained with 'method-A' is significantly better than the LedNet trained with 'method-B', which are 0.826 and 0.797 on AP_{50}, respectively. Meanwhile, LedNet trained with 'method-A' achieves a balance performance on the detection of objects in different scales. Comparably, the detection performance of the LedNet trained with 'method-B' shows a considerable reduction on the detection of objects in the median and large scale. The LedNet trained with 'method-A' has higher accuracy in boundary box localisation than the LedNet trained with 'method-B', which are 86.7% and 78.3%, respectively.

Influence on Training Loss: This experiment compares the performance of LedNet by training with focal loss and MSE. The evaluation results are shown in Table 4.

From the experiment, the LedNet trained with focal loss achieves better performance than the network trained with MSE. The recall and accuracy of the LedNet trained with focal loss are 0.82 and 0.85, respectively. Comparatively, the recall and accuracy of LedNet trained with MSE are 0.817 and 0.831, respectively. Overall, the LedNet trained with focal loss achieves 0.832 on F_1 score, which is 1.6% higher than the LedNet trained MSE.

Influence on Backbone: LedNet can utilise different networks as the backbone. This experiment compares the performance of the LedNet with different backbones, including LW-net, resnet-50, resnet-101, and darknet-53. The evaluation results of comparison in detection performance and computational efficiency are shown in Tables 5 and 6, respectively.

Table 5 shows that LedNet with resnet-101 outperforms the other models, it achieves 0.849, 0.84 and 0.86 on $F1$ score, accuracy and recall, respectively. Meanwhile, LedNet with resnet-101 also achieves higher accuracy on boundary box localisation compared to the other models. The experimental results indicate that the better performance of the backbone can improve the detection performance of the model. On the other hand, LedNet with light-weight backbone LW-net shows a balance performance on fruit detection and computational efficiency. It achieves 0.834, 0.821, and 0.853 on $F1$ score, recall, and accuracy, respectively. The weight size and computational time of the LedNet with LW-net are 7.4 M and 28 ms. Comparably, the weight size and computational time of the LedNet with resnet-101 are 188 M and 46 ms. The LedNet with resnet-101 and LW-net are applied in the comparison to state of the art as they outperform in comparison of

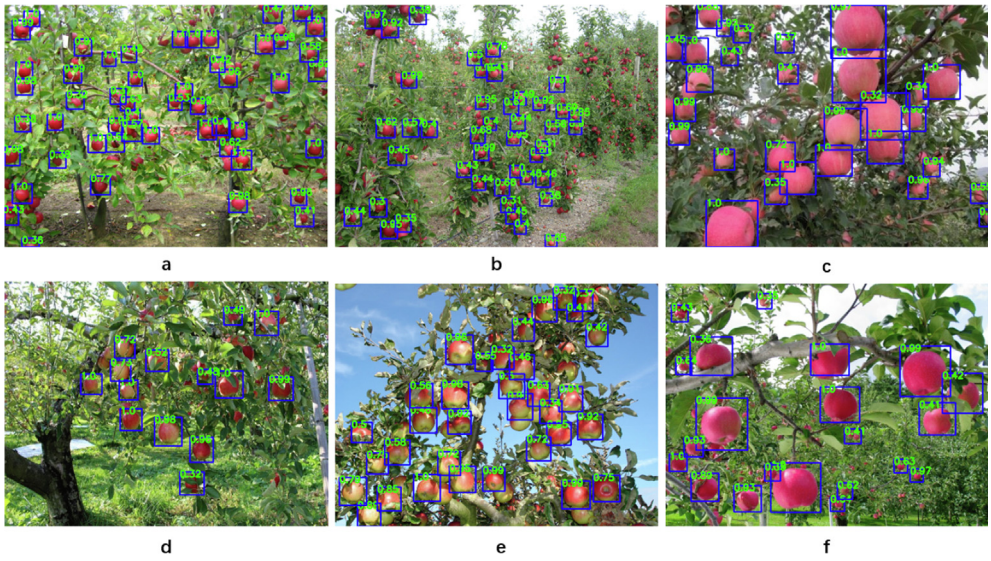

Fig. 9. Detection results of apple using LedNet(LW-net) in supplement dataset (green number is the confidence score). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

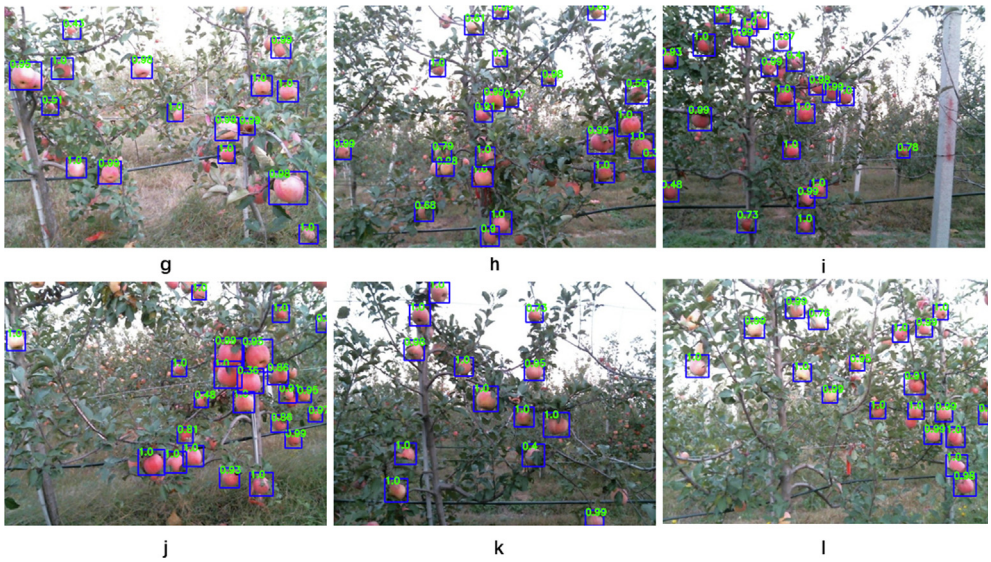

Fig. 10. Detection results of apple using LedNet(LW-net) in orchard dataset (green number is the confidence score). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 8
Computation time and weights size of different models on GTX-1080TI (resolution of images are 640 x 480)

Model	Average time	weights size
LedNet (LW-net)	28 ms	7.4 M
LedNet (resnet-101)	46 ms	188 M
YOLO-V3	45 ms	248 M
YOLO-V3 (Tiny)	30 ms	35.4 M
Faster-RCNN (VGG)	145 ms	533 M

detection performance and computational efficiency, respectively.

4.4. Comparison to state of the art

Performance Evaluation: A comparison between LedNet, YOLO-V3, YOLO-V3(tiny), and Faster-RCNN is included in this experiment. YOLO-V3 is the state of the art in the one-stage detector. YOLO-V3 uses FPN in the network architecture to improve the detection performance on different scale objects. YOLO-V3 (Tiny) is the light-weight version of the YOLO-V3, which utilises a light-weight backbone and 2-level FPN to improve the real-time detection performance. On the other hand, Faster-RCNN is state of the art in the two-stage detector. Compared to the one-stage detector, Faster R-CNN has better detection performance while its computation efficiency is lower than the YOLO-V3 as it comprises two tasks within the detection. The performance of the different detectors are shown in Table 7. Figs. 9 and 10 show the detection on the validation dataset by using LedNet (LW-net).

From the experimental results, the two-stage detector Faster-RCNN outperforms the one-stage detector YOLO-V3 in apple detection, including the areas of recall, accuracy, and boundary box localisation accuracy. The experimental results indicate that RPN allows Faster-RCNN to encode information about objects from the proper area of the ROI within the feature maps, leading to a better detection performance compared to the YOLO-V3 model. LedNet with resnet-101 achieves 0.849, 0.841, and 0.864 on F_1 score, recall, and accuracy of the detection, respectively. The evaluation results show that LedNet outperforms the YOLO-V3 and Faster-RCNN in the evaluation. With the

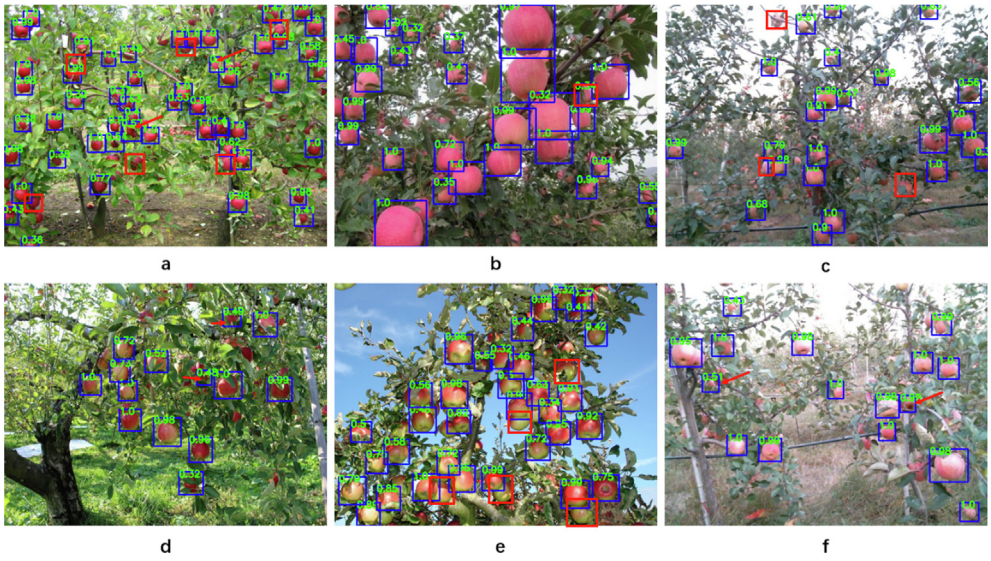

Fig. 11. Detection of apple in occlusion and overlapping conditions.

introduction of the ASPP, LedNet can encode the information of objects from the multi-scale behaviour to improve the detection performance of the network model. Meanwhile, LedNet utilises the more powerful backbone such as resnet-101, to achieve a better performance compared to the Faster R-CNN in fruit detection of apples in orchards.

Computation Efficiency: The comparison of computation efficiency between different detectors is shown in Table 8. The weights size and computational time of the LedNet(LW-net) on an image (640 x 480) with GTX-1080Ti are 7.4 M and 28 ms, respectively. YOLO-V3 (tiny) achieves similar results; the weights size and computational time of YOLO-V3 (tiny) is 35.4 M and 30 ms, respectively. Compared to the YOLO-V3 (tiny), LedNet (LW-net) achieves a better detection performance. The F_1 score and IoU of LedNet with LW-net are 0.826 and 86.3%, which are 4.4% and 3.9% higher than the YOLO-V3 (tiny), respectively. LedNet with resnet-101 achieves similar computation efficiency compared to the YOLO-V3, from the experimental results. Faster-RCNN is a two-stage detector, which includes an RPN and classification network. Therefore, the computational time of Faster-RCNN is 145 ms, which is much longer than the one-stage detector YOLO and LedNet.

4.5. Detection under occlusion and overlapping

Fruit overlapping and occlusion between fruits, branches, and leaves are challenging issues in the fruit detection in orchards. The detection results are shown in Fig. 11, blue boxes are the apples detected by the LedNet, while the red boxes and red arrows are the false-negative and false-positive of the apple detection, respectively. As shown in the results, the apples which are mostly obscured by branches or leaves are still detected by the LedNet, leading to the false-positives in the detection. The apples which overlap with each other may lead to missing detection by LedNet, leading to the false-negatives in detection. From the experimental results shown in Table 7 and Fig. 11, LedNet achieves a good performance in apple detection under the occlusion and overlapping conditions.

5. Conclusion and future work

In this study, a fast implementation framework of a deep-learning based fruit detection algorithm was developed. This framework includes a label generation module and a fruit detector LedNet. The label generation module utilised the multi-level pyramid and clustering-based classifier to assist fast labelling of training data. LedNet utilised the FPN and ASPP to enhance feature extraction ability and detection performance of the model. Meanwhile, a light-weight backbone was developed to improve computation efficiency of the model. From the experimental results, LedNet with resnet-101 achieved 0.841 and 0.864 on recall and accuracy on the fruit detection of apples in orchards, respectively. The LedNet with light-weight backbone achieved 0.821 and 0.853 on recall and accuracy on the apple detection, and the weights size and average computational time are 7.4 M and 28 ms, respectively. Future work will focus on embedding more functions into the LedNet, including growth monitoring, yield estimation and ripeness detection. Moreover, future work will investigate the automatic labelling generation techniques such as generative adversarial network, to further reduce human intervention during the network training.

Declaration of Competing Interest

The authors declare no conflict of interest.

Acknowledgement

This work is supported by ARC ITRH IH150100006 and THOR TECH PTY Ltd. We acknowledge Zijue Chen and Hongyu Zhou for their assistance in the data collection. And we also acknowledge Zhuo Chen for her assistance in preparation of this work.

References

Badrinarayanan, Vijay, Kendall, Alex, Cipolla, Roberto, 2017. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 39 (12), 2481–2495.

Bargoti, Suchet, Underwood, James, 2017a. Deep fruit detection in orchards. In: 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, pp. 3626–3633.

Bargoti, Suchet, Underwood, James P, 2017. Image segmentation for fruit detection and yield estimation in apple orchards. J. Field Robot. 34 (6), 1039–1060.

Girshick, Ross, Donahue, Jeff, Darrell, Trevor, Malik, Jitendra, 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587.

Greenspan, Hayit, Goldberger, Jacob, Eshet, Itay, 2001. Mixture model for face-color modeling and segmentation. Pattern Recogn. Lett. 22 (14), 1525–1536.

Hashimoto, Koichi, 2003. A review on vision-based control of robot manipulators. Adv. Robot. 17 (10), 969–991.

He, Kaiming, Gkioxari, Georgia, Dollár, Piotr, Girshick, Ross, 2017. Mask r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2961–2969.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, Sun, Jian, 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778.

Huang, Gao, Liu, Zhuang, Van Der Maaten, Laurens, Weinberger, Kilian Q, 2017. Densely

connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4700–4708.
Kamilaris, Andreas, Prenafeta-Boldu, Francesc X., 2018. Francesc X Prenafeta-Boldu. Deep learning in agriculture: a survey. Comput. Electron. Agric. 147, 70–90.
Kapach, Keren, Barnea, Eh.ud., Mairon, Rotem, Edan, Yael, Ben-Shahar, Oh.ad., 2012. Computer vision for fruit harvesting robots–state of the art and challenges ahead. Int. J. Comput. Vision Robot. 3 (1/2), 4–34.
Koirala, A., Walsh, K.B., Wang, Z., McCarthy, C., 2019. Deep learning for real-time fruit detection and orchard fruit load estimation: benchmarking of 'mangoyolo'. Precis. Agric. 1–29.
Kussul, Nataliia, Lavreniuk, Mykola, Skakun, Sergii, Shelestov, Andrii, 2017. Deep learning classification of land cover and crop types using remote sensing data. IEEE Geosci. Remote Sens. Lett. 14 (5), 778–782.
Laroca, Rayson, Severo, Evair, Zanlorensi, Luiz A., Oliveira, Luiz S., Gonçalves, Gabriel Resende, Schwartz, William Robson, Menotti, David, 2018. A robust real-time automatic license plate recognition based on the yolo detector. In: 2018 International Joint Conference on Neural Networks (IJCNN). IEEE, pp. 1–10.
Lehnert, Christopher, Sa, Inkyu, McCool, Christopher, Upcroft, Ben, Perez, Tristan, 2016. Sweet pepper pose detection and grasping for automated crop harvesting. In: 2016 IEEE International Conference on Robotics and Automation (ICRA). IEEE, pp. 2428–2434.
Li, Yanan, Cao, Zhiguo, Xiao, Yang, Cremers, Armin B, 2017. Deepcotton: in-field cotton segmentation using deep fully convolutional network. J. Electron. Imaging 26 (5), 053028.
Lin, Guichao, Tang, Yunchao, Zou, Xiangjun, Xiong, Juntao, Li, Jinhui, 2019. Guava detection and pose estimation using a low-cost rgb-d sensor in the field. Sensors 19 (2), 428.
Lin, Tsung-Yi, Goyal, Priya, Girshick, Ross, He, Kaiming, Dollár, Piotr, 2017. Focal loss for dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2980–2988.
Long, Jonathan, Shelhamer, Evan, Darrell, Trevor, 2015. Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431–3440.
Hao, Lu., Cao, Zhiguo, Xiao, Yang, Li, Yanan, Zhu, Yanjun, 2016. Region-based colour modelling for joint crop and maize tassel segmentation. Biosyst. Eng. 147, 139–150.
Majeed, Yaqoob, Zhang, Jing, Zhang, Xin, Longsheng, Fu., Karkee, Manoj, Zhang, Qin, Whiting, Matthew D, 2018. Apple tree trunk and branch segmentation for automatic trellis training using convolutional neural network based semantic segmentation. IFAC-PapersOnLine 51 (17), 75–80.
McCool, Christopher, Sa, Inkyu, Dayoub, Feras, Lehnert, Christopher, Perez, Tristan, Upcroft, Ben, 2016. Visual detection of occluded crop: For automated harvesting. In: 2016 IEEE International Conference on Robotics and Automation (ICRA). IEEE, pp. 2506–2512.
Ng, Pauline C., Henikoff, Steven, 2003. Sift: Predicting amino acid changes that affect protein function. Nucl. Acids Res. 31 (13), 3812–3814.
Papandreou, George, Chen, Liang-Chieh, Murphy, Kevin P, Yuille, Alan L., 2015. Weakly- and semi-supervised learning of a deep convolutional network for semantic image segmentation. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1742–1750.
Pass, Greg, Zabih, Ramin, Miller, Justin, 1996. Comparing images using color coherence vectors. In: ACM Multimedia, vol. 96. Citeseer, pp. 65–73.
Ratner, A., Bach, S., Varma, P., Ré, C, 2017. Weak supervision: The new programming paradigm for machine learning.
Redmon, Joseph, Divvala, Santosh, Girshick, Ross, Farhadi, Ali, 2016. You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788.
Redmon, Joseph, Farhadi, Ali, 2017. Yolo9000: better, faster, stronger. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7263–7271.
Redmon, Joseph, Farhadi, Ali, 2018. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.
Ren, Shaoqing, He, Kaiming, Girshick, Ross, Sun, Jian, 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, pages 91–99, 2015.
Sa, Inkyu, Ge, Zongyuan, Dayoub, Feras, Upcroft, Ben, Perez, Tristan, McCool, Chris, 2016. Deepfruits: A fruit detection system using deep neural networks. Sensors 16 (8), 1222.
Shin, Hoo-Chang, Roth, Holger R., Gao, Mingchen, Le, Lu, Ziyue, Xu, Nogues, Isabella, Yao, Jianhua, Mollura, Daniel, Summers, Ronald M, 2016. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. IEEE Trans. Med. Imaging 35 (5), 1285–1298.
Srivastava,Rupesh K., Greff, Klaus, Schmidhuber, Jürgen, 2015. Training very deep networks. In: Advances in Neural Information Processing Systems, pp. 2377–2385.
Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, Rabinovich, Andrew, 2015. Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9.
Szegedy, Christian, Vanhoucke, Vincent, Ioffe, Sergey, Shlens, Jon, Wojna, Zbigniew, 2016. Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818–2826.
Tian, Yunong, Yang, Guodong, Wang, Zhe, Wang, Hao, Li, En, Liang, Zize, 2019. Apple detection during different growth stages in orchards using the improved yolo-v3 model. Comput. Electron. Agric. 157, 417–426.
Tijtgat, Nils, Van Ranst, Wiebe, Goedeme, Toon, Volckaert, Bruno, De Turck, Filip, 2017. Embedded real-time object detection for a uav warning system. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2110–2118.
Uijlings, Jasper RR, Van De Sande, Koen EA, Gevers, Theo, Smeulders, Arnold W.M., 2013. Selective search for object recognition. Int. J. Comput. Vision 104 (2), 154–171.
Wang, Panqu, Chen, Pengfei, Yuan, Ye, Liu, Ding, Huang, Zehua, Hou, Xiaodi, Cottrell, Garrison, 2018. Understanding convolution for semantic segmentation. In 2018 IEEE Winter Conference on Applications of Computer Vision (WACV). IEEE, pp. 1451–1460.
Weiss, Karl, Khoshgoftaar, Taghi M, Wang, DingDing, 2016. A survey of transfer learning. J. Big Data 3 (1), 9.
Yang, Qi, Shi, Liangsheng, Han, Jinye, Zha, Yuanyuan, Zhu, Penghui, 2019. Deep convolutional neural networks for rice grain yield estimation at the ripening stage using uav-based remotely sensed images. Field Crops Res. 235, 142–153.
Yoo, Jae-Chern, Han, Tae Hee, 2009. Fast normalized cross-correlation. Circuits Syst. Signal Process. 28 (6), 819.
Yang, Yu., Zhang, Kailiang, Yang, Li, Zhang, Dongxing, 2019. Fruit detection for strawberry harvesting robot in non-structural environment based on mask-rcnn. Comput. Electron. Agric. 163, 104846.
Zhao, Yuanshen, Gong, Liang, Huang, Yixiang, Liu, Chengliang, 2016. A review of key techniques of vision-based control for harvesting robot. Comput. Electron. Agric. 127, 311–323.
Zhong, Yuanhong, Gao, Junyuan, Lei, Qilun, Zhou, Yao, 2018. A vision-based counting and recognition system for flying insects in intelligent agriculture. Sensors 18 (5), 1489.