Research Article

Automated apple defect detection using state-of-the-art object detection techniques

Raheel Siddiqi[1]

Abstract
Defected apples should be sorted out so that only high quality apple products are delivered to the customer. An automated system is therefore needed that can detect apple defects and consequently help in automated apple sorting. Even though apple defect detection has been an area of research for many years, full potential of modern convolutional object detectors needs to be more thoroughly explored for the task. In this paper, two different convolutional object detection systems are proposed that can perform the apple defect detection task. The new systems are based on the SSD and the YOLOv2 object detection frameworks. A dataset of 244 defected apple images is created for the training and testing of the detection systems. Performance results are encouraging for both the proposed systems. However, the SSD-based system has demonstrated much superior performance than the YOLOv2-based system.

Keywords Apple defect detection · Convolutional object detectors · Object detection · Machine vision · SSD · YOLOv2

Mathematics Subject Classification 68T45

1 Introduction

Apple is one of the most consumed fresh fruit [1]. Globally, it is appreciated for its nutrient value and it's pleasant, distinct flavor. Apple industry needs to maintain high quality due to ever-increasing consumer expectations [2]. Since apples are a natural, delicate commodity, they are likely to be harmed or become defected (e.g. during post-harvest operations such as transportation, storage etc.). Defected apples should be sorted out so that only high quality apple products are delivered to the customer [3]. Removal of defected apples is traditionally performed by human labor. Many undesirable inconsistencies are introduced in this manual work because the task is time-consuming and tedious [4]. This leads to poorer product quality and eventually economic losses for both producers and retailers. Therefore, there is need for an automated system that can detect apple defects and consequently help in automated apple sorting.

Some of the common apple diseases include *apple rot*, *apple scab* and *apple blotch* [5]. The study (presented in this paper) mainly focuses on the apple rot disease (as most images used manifest this disease). It is important to note that apple defect detection is still a challenging task [4, 6] because of various reasons such as the occurrence of stem/calyx regions as well as the various ways apple pathologies/defects manifest themselves. In the past, there have been numerous attempts to classify (or grade) apples based on their quality [7–12] or their disease [4, 13]. There has also been an attempt to exploit an image segmentation technique to identify fruit defects on fruit peel [14]. In some other recent studies [3, 15–23], use of specialized technologies such as structured-illumination reflectance imaging (SIRI), X-ray imaging etc. have been shown to be effective in detecting apple defects/bruises.

✉ Raheel Siddiqi, drraheel.bukc@bahria.edu.pk | [1]Department of Computer Science, Bahria University (Karachi Campus), 13 National Stadium Road, Karachi, Pakistan.

SN Applied Sciences (2019) 1:1345 | https://doi.org/10.1007/s42452-019-1393-4

Received: 8 June 2019 / Accepted: 1 October 2019 / Published online: 5 October 2019

In the past, researchers have approached the apple defect detection problem mainly as an image classification problem [4, 7–13]. Only one recent research study [24] tackles the task as an object detection problem. Therefore, there is considerable scope for exploring the full potential of modern convolutional object detectors for the task. The research work presented in this paper is a step in this direction. Two convolutional object detectors (i.e. YOLO and SDD) are exploited separately for apple defect detection task. Performance of the two types of detectors, in the context of apple defect detection problem, is compared and analyzed (in this paper). Section 2 presents the literature review. Section 3 describes the dataset used, while Sect. 4 explains the methods exploited. Section 5 presents implementation details and evaluation results for the two apple defect detection systems. Finally, Sect. 6 concludes the paper by summarizing the main contributions of the research undertaken as well as indicating possible directions for future research work.

2 Literature review

Apple defect detection has been an area of research for more than 30 years [25]. Research work done in this domain can be divided into two categories [1, 8]: (a) researchers apply specialized equipment that operate using non-visible portion of the electromagnetic spectrum, and (b) researchers use machine vision techniques where imaging is based on the visible portion of the electromagnetic spectrum. The techniques that fall in the first category depend on specialized equipment/technologies such as Vis–NIR spectroscopy [19], X-ray imaging [20, 22], structured-illumination reflectance imaging (SIRI) [3, 15, 16], hyperspectral and multispectral imaging [17, 21], magnetic resonance imaging [22] and thermal imaging [23]. The research presented in this paper fall in the second category as a color digital camera is used for apple image acquisition and convolutional object detectors (based on state-of-the-art machine vision technology) are used for defect detection.

Machine vision is considered as a useful and practical tool for apple defect detection because of its simplicity, consistency, low cost and high speed [1, 26]. However, machine vision is primarily used to detect external apple defects. This is due to the lack of sufficient spectral information needed to detect internal quality [1]. Despite this limitation, it is important to note that presence (or absence) of external defect(s) is one of the most influential factor in determining an apple's commercial value and may even be an indication of the apple's internal quality [26], such as its sugar content [27].

Many machine vision based systems have been developed in the past to tackle the problem of apple defect detection [1]. Zou et al. [12] developed a computer controlled system for image classification of healthy and defected apples. The system is based on three color cameras that capture nine images for each apple. Taking nine different images ensures that the apple's entire surface area is scanned. The classification algorithm focuses on differentiating apple's stem-end and calyx from defects. Regions of Interest (ROI) are segmented and counted in each image. These ROIs are based on detection of stem-end, calyx and genuine defects found in each image. Since a calyx and stem-end cannot appear together in the same image, an apple is classified as defective if there are two or more ROIs in the same image. Good classification accuracy is reported by Zou et al. [12].

Dubey and Jalal [4, 13] perform apple disease classification by combining color, texture and shape features into a single descriptor. Multi class support vector machine is then used to classify apples into healthy or infected. The infected apples are further classified into three disease categories: blotch, rot and scab. Classification results indicate that the combined single descriptor performs better than the color, texture and shape features standalone.

Zhang et al. [9] uses a lightness correction method to solve the problem of uneven lightness distribution on the apples' surface. Candidate defect regions are extracted and classified as genuine defect or stem/calyx using a weighted RVM classifier. The apples are then classified as healthy or defective based on the type of candidate defect regions i.e. whether they are genuine defects or not. An overall classification accuracy of 95.63% has been reported.

Bhatt and Pant [11] have developed a real-time apple classification system based on back-propagation artificial neural network (ANN). Four categories of apples are used during the system training and testing. The first category has the best apples, while the fourth category has defected apples. The other two categories contain apples of intermediary qualities. The ANN classifies apples based on physical features such as size, color and external defects. The reported classification accuracy is high (around 96%), indicating that ANNs are a good tool to classify apples based on quality.

Sofu et al. [10] have proposed a different real-time apple classification system that sorts three apple types, i.e. golden delicious, starking delicious and granny smith, with sorting accuracy of 73–96%. The system is also capable of identifying defected apples. Three defect types are considered: scab, stain and rot. An image processing software is used to extract apple's surface features such as color, size, stain etc. and C4.5 algorithm is used for the classification purpose. The system's software is fast, simple and flexible.

Fig. 1 Sample defected apple images from the dataset

The system's hardware components include roller, transporter and class conveyors combined with machine vision and control panel units.

Moallem et al. [8] compares the performance of Support Vector Machine (SVM), Multi-Layer Perceptron (MLP) and K-Nearest Neighbor (KNN) classifiers in the grading of golden delicious apples. Some preprocessing steps are first performed to remove stem/calyx regions from genuine defect regions. Features are extracted from genuine defect regions and fed into classifiers to perform the classification task. Two classification tasks are performed: (1) an input apple image is classified as healthy or defected, and (2) an input apple image is classified as first rank, second rank or rejected. SVM classifier performs best in both tasks securing classification accuracies of 92.5% and 89.2%, respectively. Ji et al. [7] have also attempted apple grading using a SVM model (based on particle swarm optimization) and have reported maximum accuracy rate of 91%.

Recently, Tian et al. [24] have proposed a YOLOV3-Dense model for detection of apple lesions. The model is trained on a dataset of 640 defected/healthy apple images collected in two ways: orchard field collection and online collection. Data augmentation techniques like Cycle-Consistent Adversarial Network (CycleGAN) is used to artificially expand the dataset. DenseNet is used as a feature extractor to enhance the detection results of the YOLO-v3 [28] model. This is the first and also the most recent study [24] that has proposed a convolutional object detector for apple defect detection. Prior studies have approached the problem of defect detection as image classification problem rather than object detection problem. The study provides the basis for validating the models presented in this paper.

3 Dataset

The dataset images are collected by a group of three students at Bahria University (Karachi Campus). All the images are taken at Karachi's local fruit market known as *subzi mandi*. All the apples photographed belonged to the 'golden apple' category. More than 300 images were initially taken but some images were discarded due to their poor quality. 244 images were finalized for the *defected apples dataset*.[1] The dataset is further divided into two subsets: train set and test set. Train set contains 218 images while the test set contains 26 images. Figure 1 depicts some sample defected apple images from the dataset. The images in the dataset were originally of very high resolution i.e. 2988 pixels wide and 5312 pixels high. The processing of such high resolution images requires significant computing power and memory. For this reason, all the images of the dataset are resized to smaller dimensions i.e. 280×400.

Each image is taken in a way that the object instance (i.e. a particular defected apple) tend to be large and central. While taking images, a white sheet of paper is placed behind each defected apple in order to ensure uniform and clear background. A number of steps are taken in order to make sure that the dataset is realistic:

[1] The entire dataset (along with the corresponding annotation files) can be downloaded from the following link: https://github.com/raheelsiddiqi2013/apple-defect-detection/blob/master/Defected%20Apple%20Images%20and%20Their%20Annotations.rar.

1. A mobile phone camera has been used to capture all the images. The use of professional cameras like DSLR has been avoided. This is done because the object detection models, presented in this paper, are designed, trained and evaluated to work on more realistic image data (rather than just perfect images taken through professional cameras). The mobile phone used to take the dataset images is Samsung Galaxy Grand Prime Pro (model number: SM-J250F) with eight mega pixel camera.
2. Images are taken from a variety of different angles, poses and distances. In some images, apple's calyx/stem region is more prominent/focused. In others, the stem/calyx area is hidden or partially hidden.
3. Lighting conditions have also been varied from image to image. This is achieved by changing the number of LED tube lights switched on at the time of image capture.

A particular area of the apple's surface is considered defected if the lesion has grown greater than 10 mm in diameter. Lesions are result of some apple disease or decay. Lesion areas manifest themselves as dark brown or black patches and are easily distinguishable from the healthy areas of the apple. For this study, the author has only included apple images where the apple lesion is localized to a particular area of the apple's surface and has not advanced to such a stage that it has deformed or destroyed the entire apple.

The dataset also contains corresponding annotation files. There is one annotation file per image in the dataset. Each annotation file is saved as an XML file in PASCAL VOC [29, 30] format and is created using the LabelImg tool [31].

4 Methods

Significant performance improvements have been achieved in the field of object detection as a result of using Convolutional Neural Networks (CNNs) [32–34]. Modern convolutional object detectors are now capable enough to be used in consumer products (e.g. Google photos) and fast enough to be used in mobile devices [32]. In addition, convolutional object detectors are nowadays adapted to perform a more diverse set of tasks such as customized object detection for indoor robots [35], incorporation of temporal and contextual information into object detection in videos [36] and detection of masses in mammograms for breast cancer diagnosis [37].

Some well-known and widely used convolutional object detectors include Faster R-CNN [38], YOLO [39], SSD [40] and R-FCN [41]. Convolutional object detectors can be sub-divided into two broad categories [34]: (1) *region-based* e.g. Faster R-CNN [38] and R-FCN [41], and (2) *proposal-free* e.g. YOLO [39] and SSD [40].

Faster R-CNN [38], like other region-based methods, performs detection in two stages. In the first stage, region proposal network (RPN) extract object proposals while in the second stage these proposals are passed to the fully connected layer for classification and prediction of bounding boxes. Region-based methods (including Faster R-CNN) are very accurate but have high computational cost (i.e. low frame rate) [32, 34] and therefore are not usually considered the best option for embedded devices [40].

For the purpose of the research presented in this paper, the author has experimented with YOLO and SSD. YOLO and SSD directly predict object's category and position i.e. no region proposals are computed. This makes them faster than region-based detectors [34]. Rather than requiring per proposal classification operation, these proposal-free detection frameworks apply a single neural network to the full image [32, 40, 42] i.e. a single network evaluation yields predictions [42]. The following sub-sections present brief descriptions of YOLO and SSD:

4.1 YOLO: you only look once

Processing images with YOLO is a three step process [39]. The system (1) resizes input image to 448×448, (2) feeds the resized image to a CNN, and (3) filter the resulting detections using non-max suppression algorithm. As the name suggests, you only look once (YOLO) to predict which objects are present and what is their location in the image. YOLO tackles the object detection task as a single regression problem, predicting bounding box coordinates and associated class probabilities directly from image pixels [39] in one evaluation. The whole detection pipeline is very simple (based on a single convolutional neural network), making YOLO extremely fast [39, 42, 43].

Apart from being extremely fast, YOLO has other benefits [39]. YOLO looks at the entire image during training and testing, therefore it implicitly encodes contextual information. This makes YOLO more capable of distinguishing background patches in an image from actual objects. Therefore, number of background errors is much less for YOLO compared with other detection frameworks. YOLO has also proven to be very good at learning generalizable representations for objects. Compared to other detection frameworks, YOLO performs better when trained on natural images (such as VOC 2007 dataset) and tested on artwork (such as the Picasso dataset [44] and the People-Art dataset [45]).

Redmon et al. [39] states that major drawback of YOLO is its poor accuracy compared with other state-of-the-art detection systems. YOLO struggles in object localization. For this purpose, a new improved model is proposed in

Table 1 Performance of Faster R-CNN, YOLO, YOLOv2 and SSD detection frameworks on PASCAL VOC 2007

Detection framework	Input image resolution	Mean average precision (mAP)	Frame per second (FPS)
Faster R-CNN [38]	–	73.2	7
YOLO [39]	–	63.4	45
YOLOv2 [42]	288×288	69.0	91
YOLOv2 [42]	416×416	76.8	67
YOLOv2 [42]	544×544	78.6	40
SSD [40]	300×300	74.3	46
SSD [40]	512×512	76.8	19

[42] called YOLOv2 which is more accurate and faster than prior detection frameworks (see Table 1). The author has used YOLOv2 for his experiment.

4.2 SSD: single shot multibox detector

As the name suggests, SSD needs only a single step (or shot) to detect multiple objects within an image.[2] Like YOLO, the approach encapsulates all computations in a single convolutional neural network [40] and therefore provides a unified framework for both training and inference.

SSD divides the output space into a set of default boxes over different aspect ratios and scales per feature map location [40]. At training time, these default boxes are matched to the ground truth boxes. At the time of prediction, the network produces scores for the presence of each object category in each default box and adjusts the default box to match the object shape.

Table 1 presents performance of YOLOv2 and SSD frameworks on PASCAL VOC 2007 dataset. YOLOv2 appears to have an edge over SSD both in terms of accuracy as well as speed. For both YOLO and SSD, an increase in input image resolution results in increase of detection accuracy (i.e. mean average precision) and decrease of detection speed (i.e. frame per second). Frame rates presented in Table 1 are all measured on GeForce GTX Titan X machine.

5 Implementation and evaluation

In order to train an object detection model for a custom object like apple defect, two options are available: (1) use a pre-trained model and then use transfer learning to learn the new object, or (2) create a model that learns the new object from scratch. The author chose transfer learning because training becomes much quicker and less training data is required (if transfer learning is used).

The experiments (presented in this paper) are carried out on a NVIDIA GeForce GTX 1050 Ti machine with Intel Core i7-7700HQ 64-bit processor and 16 GB RAM. The code for the experiments is written in Python 3.6.0 using TensorFlow version 1.10.0. OpenCV library is also needed for the YOLO v2 based apple defect detection experiment. Jupyter Notebook (which is included in the Anaconda package) is exploited to write and present the code.

This section is further sub-divided into three subsections explaining the creation and training of the two apple defect detectors and then later comparing the performance of the two detectors.

5.1 Setting up and training the SSD-based apple defect detector

The SSD-based detector is developed and trained using *TensorFlow Object Detection API* [46] which is an open source framework based on TensorFlow. Before the training starts, the following steps are performed:

1. TFRecord files are generated for the train and test samples of the defected apples dataset.
2. A pre-trained SSD model (with Mobilenet as feature extractor) is downloaded.[3] The model is pre-trained on COCO dataset [47]. A corresponding configuration file is also setup.

Figure 2 depicts training curve of the SSD-based apple defect detector. Training is carried out for around 16,000 steps and the loss is minimized to around 1.5. After training, inference graph from the new trained model is exported using the relevant checkpoint and configuration files. The trained model is now ready for the testing phase.

5.2 Setting up and training the YOLO-based apple defect detector

YOLO is originally written in a deep learning framework called Darknet [43] that is completely written in C and uses CUDA [48]. This implementation of YOLOv2 is very fast but not very user friendly. Darknet has been translated to

[2] This is in contrast to region-based approaches (like Faster R-CNN) that need two steps (or shots): one for generating region proposals and the other for detecting the object of each proposal.

[3] The pre-trained object detection model is downloaded from this link: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md.

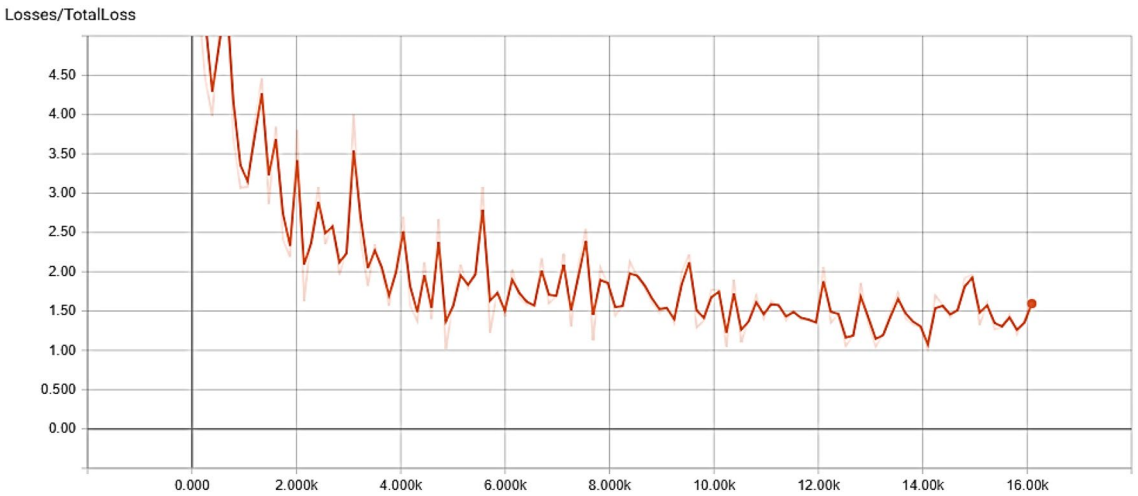

Fig. 2 Training curve for the SSD-based apple defect detector. The graph shows how the loss has evolved over the course of 16,000 training steps

TensorFlow and is available as Darkflow.[4] For the purpose of the experiment (presented in this paper), the author has used *Tiny YOLOv2* model pre-trained on VOC 2007 + 2012 datasets [43]. The weights and configuration files for Tiny YOLOv2 are downloaded from [43].

Before the training starts, a copy of the configuration file is made. The original configuration file is kept unchanged. On the other hand, some slight modifications are made to the copy of the configuration file by adjusting the number of classes in the last layer and the number of filters in the second last layer of the convolutional neural network. The number of classes is set to 1 because there is only one object class i.e. 'apple defect'. The number of filters is set to $5 * (classes + 5)$ (as specified in [48]), which is equal to 30 since number of classes is 1.

Training is carried out with a batch size of 16 images and learning rate of 1e−05. The model is trained for 1875 steps and when the training stops, both the loss and the moving average loss have become less than 1 (which is good enough). The YOLO-based apple defect detector is now ready for the testing phase.

5.3 Comparative performance of the two detectors

The two apple defect detectors are evaluated using 26 test images.[5] Like train images, all the test images contain apple defects such as rot, blotch or bruise. The author has performed evaluation of the two detectors using the *PASCAL VOC 2012 challenge* metrics[6] [30]. The two metrics used in the PASCAL VOC challenge are: (a) Precision-Recall curve, and (b) Average Precision. Before the evaluation results are presented, explanation of some key terms (involved in the evaluation process) is given below:

- *Intersection Over Union (IOU)* is defined as the area of overlap between the predicted bounding box (B_p) and the ground truth bounding box (B_{gt}) divided by the area of union between them. This is expressed by the formula:

$$IOU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})}$$

- *True Positive* (T_p) refers to a correct detection i.e. a detection where IOU is greater than or equal to IOU threshold.
- *False Positive* (F_p) refers to an incorrect detection i.e. a detection where IOU is less than IOU threshold.
- *False Negatives* (F_n) are ground truth objects with no matching detection.
- *Precision* (P) is defined as the number of T_p divided by the sum of T_p and F_p:

$$P = \frac{T_p}{T_p + F_p} = \frac{T_p}{all\ detections}$$

[4] Darkflow repository is available at https://github.com/thtrieu/darkflow.

[5] Jupyter Notebooks presenting the experiments carried out to detect apple defects are given at the following repository: https://github.com/raheelsiddiqi2013/apple-defect-detection.

[6] The implementation of the 'PASCAL VOC 2012 challenge' metrics, exploited by the author, is available at: https://github.com/rafaelpadilla/Object-Detection-Metrics.

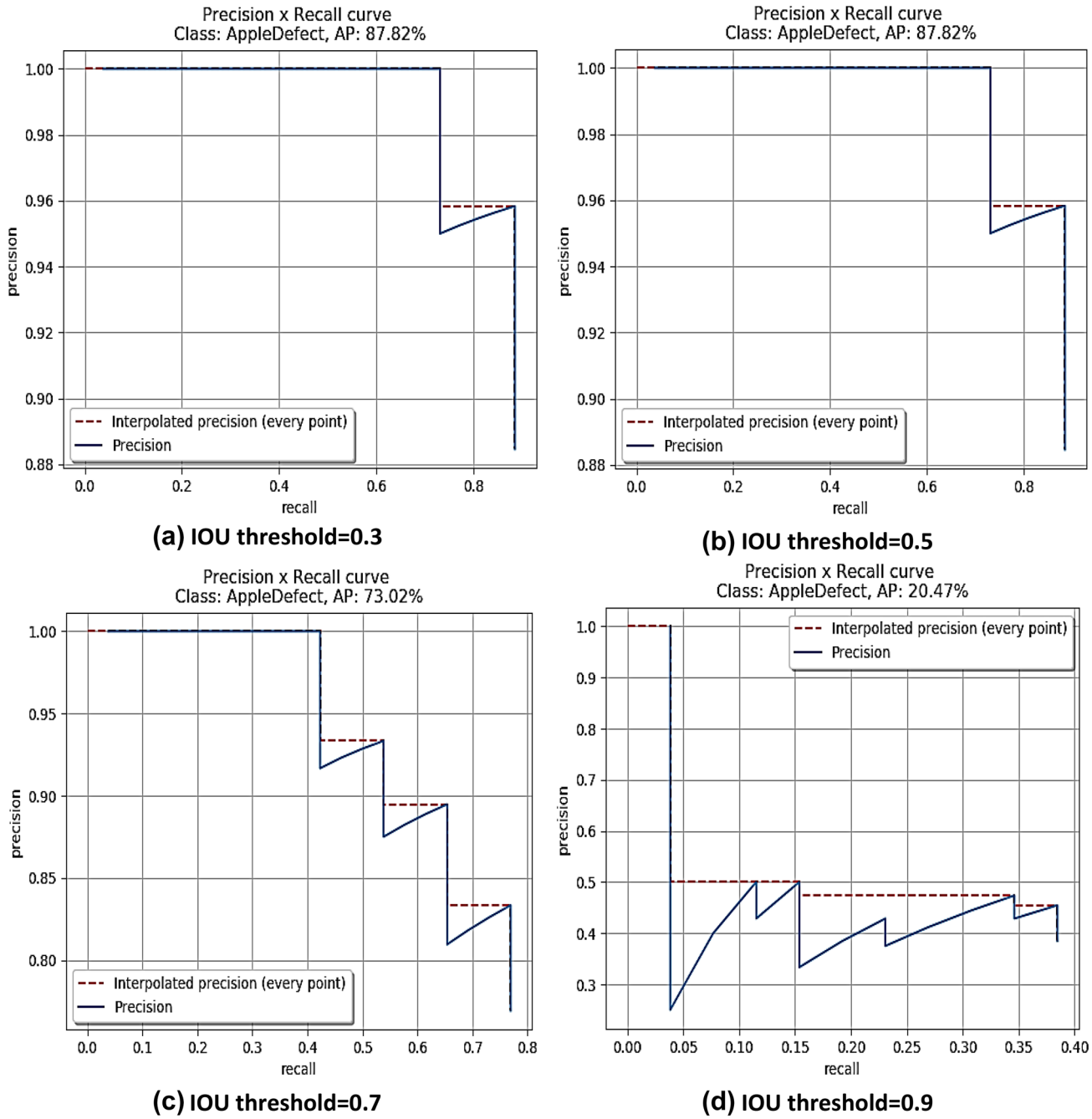

Fig. 3 Precision-recall curves for the SSD-based apple defect detector

- *Recall* (*R*) is defined as the number of T_p divided by the sum of T_p and F_n:

$$R = \frac{T_p}{T_p + F_n} = \frac{T_p}{all\ ground\ truths}$$

- *Precision-recall curve* is one of the metric used in the PASCAL VOC 2012 challenge [30]. An object detector of a particular class is considered good if its precision stays high as recall increases. A poor object detector needs to significantly lower precision to attain higher recall values.
- *Average Precision (AP)* is the precision averaged across all recall values between 0 and 1. Up until 2009, AP was calculated using the 11-point interpolation method [29]. However, from 2010 onwards the method of computing AP changed to use all data points (rather than interpolating only in the 11 equally

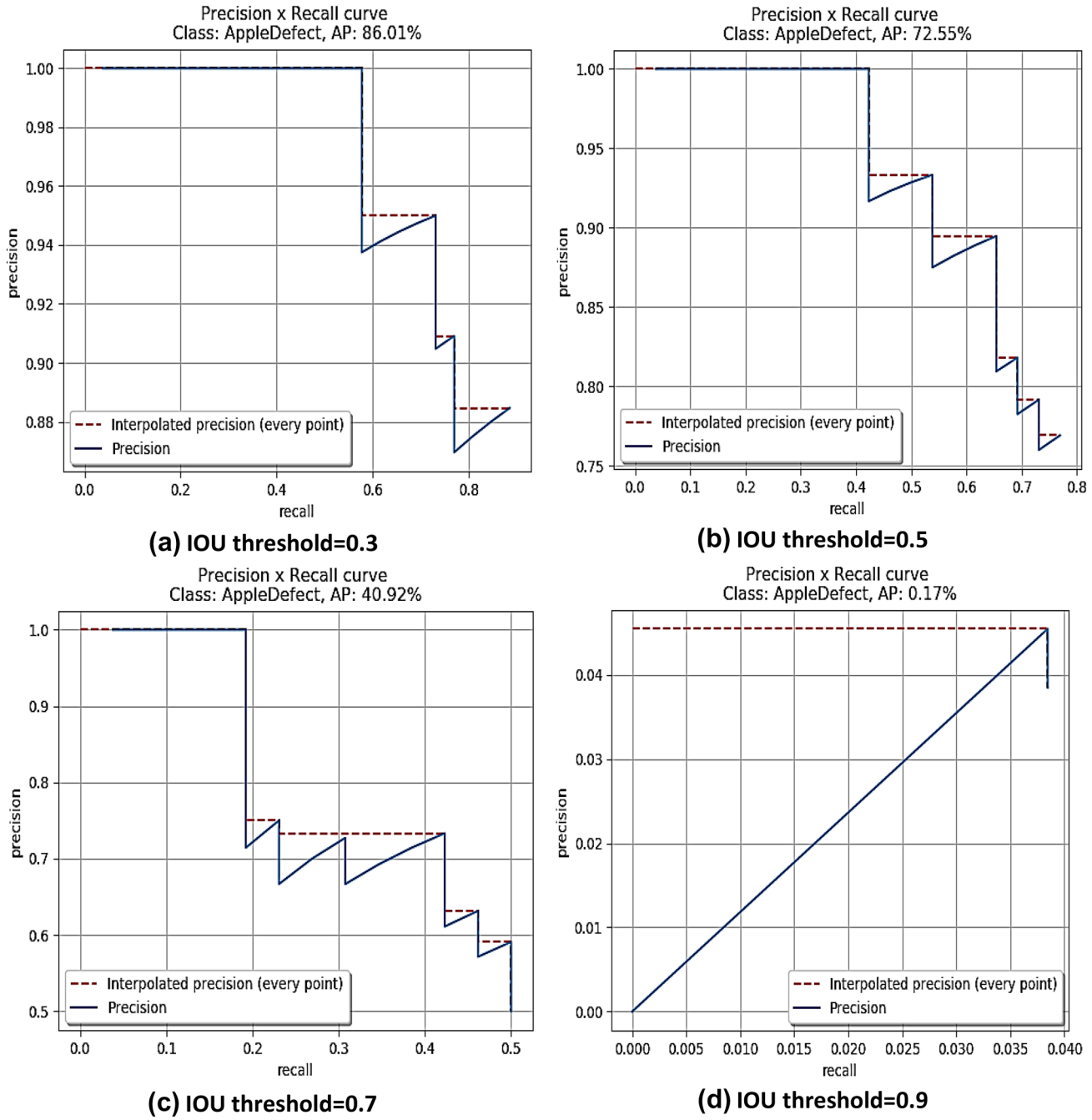

Fig. 4 Precision-recall curves for the YOLO-v2 based apple defect detector

spaced points) [30]. By interpolating all data points, the AP can be interpreted as an approximated Area Under the Curve (AUC) of the precision-recall curve.

Figure 3 presents four precision-recall curves for the SSD-based apple defect detector. These curves are for different IOU threshold values (i.e. 0.3., 0.5, 0.7, 0.9). The approximated AUC or AP remains stable and high for the first two curves (i.e. at threshold 0.3 and 0.5) but then the AP starts declining. The detector's performance at 0.7 IOU threshold is reasonable because precision degrades gradually with rising recall values resulting in an AP of 0.73. The detector's worst performance (i.e. AP of 0.204) is at 0.9 IOU threshold and this is quite understandable as detectors normally struggle at such high thresholds [32].

Figure 4 presents the four corresponding precision-recall curves for the YOLOv2-based apple defect detector. Again, the accuracy (i.e. AP) of the detector degrades

Table 2 Performance comparison of the two proposed models with the state-of-the-art model [24]

	AP@0.3IOU	AP@0.5IOU	AP@0.7IOU	AP@0.9IOU	FPS
SSD-based apple defect detector	0.878	0.878	0.730	0.204	77.5
YOLOv2-based apple defect detector	0.860	0.725	0.409	0.0017	64.0
Tian et al.'s model [24]	0.871	0.842	0.677	0.153	69.1

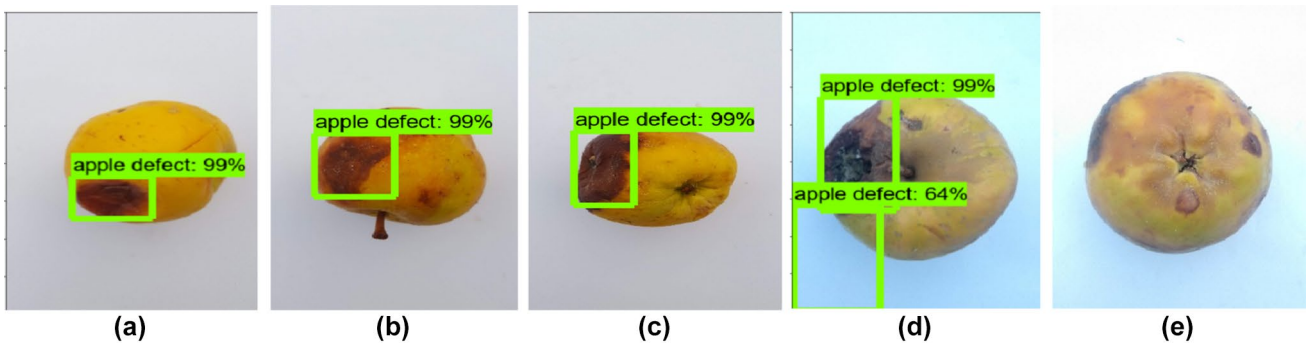

Fig. 5 Sample detections by SSD-based detector. **a**–**c** Depicts fully correct detections. **d** Depicts partially correct detection and **e** depicts an image with no detections

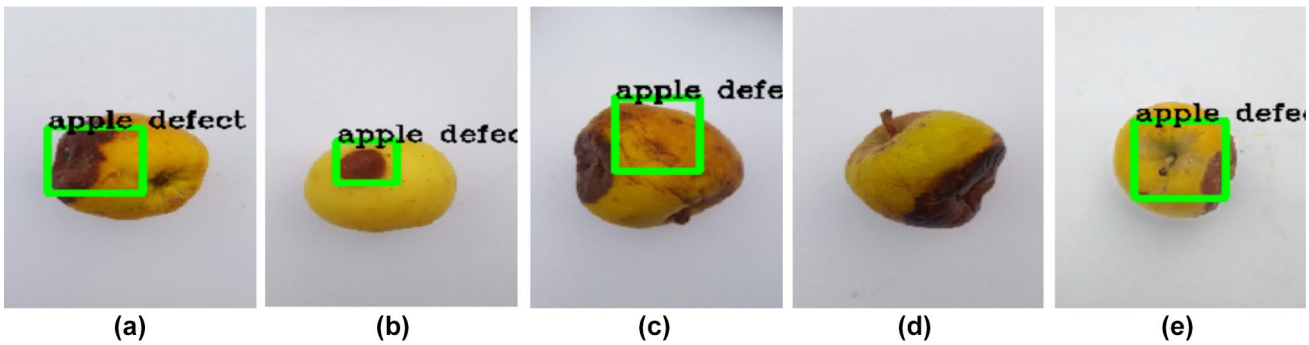

Fig. 6 Sample detections by YOLOv2-based detector. **a**, **b** Depicts fully correct detections. **c** Depicts partially correct detection. **d** Depicts no detection while **e** is an example of incorrect detection

with increasing IOU threshold. But compared with the SSD-based detector, the AP (or the approximated AUC) of this detector degrades more sharply as the threshold is increased. The detector completely fails at 0.9 threshold. However, the detector's performance at lower thresholds (i.e. 0.3, 0.5 and 0.7) is decent but still lower than the performance of the SSD-based detector for the same threshold settings.

Table 2 compares performance of the two proposed models with the state-of-the-art model presented in [24]. The state-of-the-art model has also been trained and evaluated on the same dataset of 244 images (presented in Sect. 3). It should be remembered that no data augmentation is performed for any of the experiments presented in this paper. The SSD-based detector outperforms the other two detectors both in terms of accuracy as well as speed. The SSD-based detector produced highest AP at all the four IOU threshold settings. It has also produced results at a much faster rate of 77.5 FPS compared with the 64 FPS and 69.1 FPS of the other two detectors. Of all the three models, the YOLO-v2 based detector demonstrated worst performance in terms of accuracy.

Some sample outputs produced by the two proposed detectors are presented now. Figure 5 depicts some sample apple defect detections by the SSD-based detector. The first three images of Fig. 5 are examples of perfect detection by the SSD-based system. However, the fourth image, i.e. image (d), depicts partially correct detection. The reason for calling it a partially correct detection is that even though the detector makes correct bounding box (with high confidence) over the main defect, the system erroneously makes another bounding box just adjacent

to the first box. The second bounding box hardly contains any apple defect. The (e) image of this figure is an example of no detection where the system completely fails to detect the defect on the apple's skin.

Figure 6 presents some sample detections by the YOLOv2-based detector. The first two images [i.e. (a) and (b)] are examples of perfect detection. Image (c) is an example of partially correct detection as the detector puts the bounding box over a minor defect and fails to detect the main defect on the apple's skin. Image (d) is an example of false negative where the detector fails to detect an obvious and a very large defect on the apple's surface. Image (e) is an example of incorrect detection (i.e. false positive) because the apple's stem has been incorrectly identified as a defect.

6 Conclusion

In this paper, some state-of-the-art object detection frameworks (i.e. SSD and YOLOv2) have been exploited for the task of apple defect detection. A dataset of 244 images, containing images of defected apples, have been created. Two different apple defect detectors, one based on SSD and the other on YOLOv2, are created, trained and tested on the dataset. Experimental results indicate decent performance by the two detectors but there is still considerable room for further improvement. Directions for future research include:

1. Other well-known and well-established object detection frameworks such as Faster R-CNN [38], R-FCN [41] etc. also needs to be applied for the apple defect detection task. Their respective performances for this task need to measured and analyzed.
2. A more extensive dataset of defected apple images needs to be created. This will ensure better trained models that produce higher accuracy rates. Larger test set will enable a more thorough performance evaluation.
3. Imperfect images also needs to be included in the dataset. Such images should include images with (realistic) complicated background, images where the defected apple is partially occluded by some other stuff, images where camera is not well focused etc. Inclusion of such images in the dataset will enable training and evaluation of the detection models for realistic situations.
4. Carefully crafted data augmentation techniques may be exploited to increase the size of the train set. This may help reduce overfitting and improve accuracy rates.

The author plans to extend the research (presented in this paper) based on the above mentioned directions.

Acknowledgements The defected apple images dataset was collected and compiled by three students of Bahria University (Karachi Campus): Rizwan Hussain, Sohaib Ahmed and Taha.

Compliance with ethical standards

Conflict of interest The author declares that there is no conflict of interest.

References

1. Yuzhen L, Renfu L (2017) Non-destructive defect detection of apples by spectroscopic and imaging technologies: a review. Trans ASABE 60(5):1765–1790. https://doi.org/10.13031/trans.12431
2. Harker FR, Gunson FA, Jaeger SR (2003) The case for fruit quality: an interpretive review of consumer attitudes, and preferences for apples. Postharvest Biol Technol 28(3):333–347. https://doi.org/10.1016/S0925-5214(02)00215-6
3. Yuzhen L, Renfu L (2018) Detection of surface and subsurface defects of apples using structured illumination reflectance imaging with machine learning algorithms. Trans ASABE 61(6):1831–1842. https://doi.org/10.13031/trans.12930
4. Dubey SR, Jalal AS (2016) Apple disease classification using color, texture and shape features from images. SIViP 10(5):819–826. https://doi.org/10.1007/s11760-015-0821-1
5. Hartman J (2010) Apple fruit diseases appearing at harvest. In: Plant pathology fact sheet. https://plantpathology.ca.uky.edu/files/ppfs-fr-t-02.pdf. Accessed 20 Aug 2019
6. Bhargava A, Bansal A (2018) Fruits and vegetables quality evaluation using computer vision: a review. J King Saud Univ Comput Inf Sci. https://doi.org/10.1016/j.jksuci.2018.06.002
7. Ji Y, Zhao Q, Bi S, Shen T (2018) Apple grading method based on features of color and defect. In: 2018 37th Chinese control conference (CCC). https://doi.org/10.23919/ChiCC.2018.8483825
8. Moallem P, Serajoddin A, Pourghassem H (2017) Computer vision-based apple grading for golden delicious apples based on surface features. Inf Process Agric 4(1):33–40. https://doi.org/10.1016/j.inpa.2016.10.003
9. Zhang B, Huang W, Gong L, Li J, Zhao C, Liu C, Huang D (2015) Computer vision detection of defective apples using automatic lightness correction and weighted RVM classifier. J Food Eng 146:143–151. https://doi.org/10.1016/j.jfoodeng.2014.08.024
10. Sofu MM, Er O, Kayacan MC, Cetişli B (2016) Design of an automatic apple sorting system using machine vision. Comput Electron Agric 127:395–405. https://doi.org/10.1016/j.compag.2016.06.030
11. Bhatt AK, Pant D (2015) Automatic apple grading model development based on back propagation neural network and machine vision, and its performance evaluation. AI Soc 30(1):45–56. https://doi.org/10.1007/s00146-013-0516-5
12. Xiao-bo Z, Jie-wen Z, Yanxiao L, Holmes M (2010) In-line detection of apple defects using three color cameras system. Comput Electron Agric 70(1):129–134. https://doi.org/10.1016/j.compag.2009.09.014
13. Dubey SR, Jalal AS (2012) Detection and Classification of apple fruit diseases using complete local binary patterns. In: 3rd

international conference on computer and communication technology. https://doi.org/10.1109/ICCCT.2012.76
14. Pham VH, Lee BR (2015) An image segmentation approach for fruit defect detection using k-means clustering and graph-based algorithm. Vietnam J Comput Sci 2(1):25–33. https://doi.org/10.1007/s40595-014-0028-3
15. Lu Y, Li R, Lu R (2016) Structured-illumination reflectance imaging (SIRI) for enhanced detection of fresh bruises in apples. Postharvest Biol Technol 117:89–93. https://doi.org/10.1016/j.postharvbio.2016.02.005
16. Lu Y, Lu R (2017) Using composite sinusoidal patterns in structured-illumination reflectance imaging (SIRI) for enhanced detection of apple bruise. J Food Eng 199:54–64. https://doi.org/10.1016/j.jfoodeng.2016.12.008
17. Zhang B, Liu L, Gu B, Zhou J, Huang J, Tian G (2018) From hyperspectral imaging to multispectral imaging: portability and stability of HIS–MIS algorithms for common defect detection. Postharvest Biol Technol 137:95–105. https://doi.org/10.1016/j.postharvbio.2017.11.004
18. Zhang C, Zhao C, Huang W, Wang Q, Liu S, Li J, Guo Z (2017) Automatic detection of defective apples using NIR coded structured light and fast lightness correction. J Food Eng 203:69–82. https://doi.org/10.1016/j.jfoodeng.2017.02.008
19. Jarolmasjed S, Espinoza CZ, Sankaran S (2017) Near infrared spectroscopy to predict bitter pit development in different varieties of apples. J Food Meas Charact 11(3):987–993. https://doi.org/10.1007/s11694-017-9473-x
20. Si Y, Sankaran S (2016) Computed tomography imaging-based bitter pit evaluation in apples. Biosyst Eng 151:9–16. https://doi.org/10.1016/j.biosystemseng.2016.08.008
21. Keresztes JC, Goodarzi M, Saeys W (2016) Real-time pixel based early apple bruise detection using short wave infrared hyperspectral imaging in combination with calibration and glare correction techniques. Food Control 66:215–226. https://doi.org/10.1016/j.foodcont.2016.02.007
22. Herremans E, Melado-Herreros A, Defraeye T, Verlinden B, Hertog M, Verboven P, Val J, Encarnación M, Valle F, Bongaers E, Estrade P, Wevers M, Barreiro P, Nicolaï BM (2014) Comparison of X-ray CT and MRI of watercore disorder of different apple cultivars. Postharvest Biol Technol 87:42–50. https://doi.org/10.1016/j.postharvbio.2013.08.008
23. Doosti-Irani O, Golzarian MR, Aghkhani MH, Sadrnia H, Doosti-Irani M (2016) Development of multiple regression model to estimate the apple's bruise depth using thermal maps. Postharvest Biol Technol 116:75–79. https://doi.org/10.1016/j.postharvbio.2015.12.024
24. Tian Y, Yang G, Wang Z, Li E, Liang Z (2019) Detection of apple lesions in orchards based on deep learning methods of CycleGAN and YOLOV3-Dense. J Sensors. Article ID 7630926. https://doi.org/10.1155/2019/7630926
25. Rehkugler GE, Throopmann JA (1989) Image processing algorithm for apple defect detection. Trans ASAE 32(1):267–272. https://doi.org/10.13031/2013.30994
26. Li JB, Huang WQ, Zhao CJ (2015) Machine vision technology for detecting the external defects of fruits—a review. Imaging Sci J 63(5):241–251. https://doi.org/10.1179/1743131X14Y.0000000088
27. Steinmetz V, Roger JM, Moltó E, Blasco J (1999) On-line fusion of colour camera and spectrophotometer for sugar content prediction of apples. J Agric Eng Res 73(2):207–216. https://doi.org/10.1006/jaer.1999.0407
28. Redmon J, Farhadi A (2018) YOLOv3: an incremental improvement. arXiv:1804.02767
29. Everingham M, Gool LV, Williams CKI, Winn J, Zisserman A (2010) The PASCAL visual object classes (VOC) challenge. Int J Comput Vis 88:303–338. https://doi.org/10.1007/s11263-009-0275-4
30. Everingham M, Eslami SMA, Gool LV, Williams CKI, Winn J, Zisserman A (2015) The PASCAL visual object classes challenge: a retrospective. Int J Comput Vision 111(1):98–136. https://doi.org/10.1007/s11263-014-0733-5
31. Tzutalin (2015) LabelImg. Git code. https://github.com/tzutalin/labelImg Accessed 12 Mar 2019
32. Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, Fischer I, Wojna Z, Song Y, Guadarrama S, Murphy K (2017) Speed/accuracy trade-offs for modern convolutional object detectors. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR). https://doi.org/10.1109/CVPR.2017.351
33. Chen T, Lu S, Fan J (2018) S-CNN: subcategory-aware convolutional networks for object detection. IEEE Trans Pattern Anal Mach Intell 40(10):2522–2528. https://doi.org/10.1109/TPAMI.2017.2756936
34. Li Z, Chen Y, Yu G, Deng Y (2018) R-FCN++: towards accurate region-based fully convolutional networks for object detection. In: Thirty-second AAAI conference on artificial intelligence, pp 7073–7080
35. Alabachi S, Sukthankar G, Sukthankar R (2019) Customizing object detectors for indoor robots. arXiv:1902.10671
36. Kang K, Li H, Yan J, Zeng X, Yang B, Xiao T, Zhang C, Wang Z, Wang R, Wang X, Ouyang W (2018) T-CNN: tubelets with convolutional neural networks for object detection from videos. IEEE Trans Circuits Syst Video Technol 28(10):2896–2907. https://doi.org/10.1109/TCSVT.2017.2736553
37. Jung H, Kim B, Lee I, Yoo M, Lee J, Ham S, Woo O, Kang J (2018) Detection of masses in mammograms using a one-stage object detector based on a deep convolutional neural network. PLoS ONE. https://doi.org/10.1371/journal.pone.0203355
38. Ren S, He K, Girshick R, Sun J (2017) Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell 39:1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031
39. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: unified, real-time object detection. In: 29th IEEE conference on computer vision and pattern recognition (CVPR), pp 779–788. https://doi.org/10.1109/CVPR.2016.91
40. Liu W et al (2016) SSD: single shot multibox detector. In: Leibe B, Matas J, Sebe N, Welling M (eds) Computer vision—ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9905. Springer, Cham. https://doi.org/10.1007/978-3-319-46448-0_2
41. Dai J, Li Y, He K, Sun J (2016) R-FCN: Object detection via region-based fully convolutional networks. In: 30th international conference on neural information processing systems, pp 379–387
42. Redmon J, Farhadi A (2017) YOLO9000: better, faster, stronger. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR), pp 7263–7271
43. Redmon J, Farhadi A (2016) YOLO: real-time object detection. https://pjreddie.com/darknet/yolov2/. Accessed 20 Aug 2019
44. Ginosar S, Haas D, Brown T, Malik J (2015) Detecting people in cubist art. In: Agapito L, Bronstein M, Rother C (eds) Computer vision—ECCV 2014 Workshops. ECCV 2014. Lecture Notes in Computer Science, vol 8925. Springer, Cham. https://doi.org/10.1007/978-3-319-16178-5_7
45. Cai H, Wu Q, Corradi T, Hall P (2015) The cross-depiction problem: computer vision algorithms for recognising objects in artwork and in photographs. arXiv:1505.00110
46. Huang J, Rathod V, Votel R, Chow D, Sun C, Zhu M, Fathi A, Lu Z (2019) TensorFlow object detection API. https://github.com/tensorflow/models/tree/master/research/object_detection. Accessed 20 Aug 2019
47. Lin TY et al (2014) Microsoft COCO: common objects in context. In: Fleet D, Pajdla T, Schiele B, Tuytelaars T (eds) Computer vision—ECCV 2014. ECCV 2014. Lecture Notes in Computer

Science, vol 8693. Springer, Cham. https://doi.org/10.1007/978-3-319-10602-1_48
48. Jay M (2017) Image detection with YOLO-v2 (pt.1) render video. Video. https://www.youtube.com/watch?v=PyjBd7IDYZs&list=PLX-LrBk6h3wSGvuTnxB2Kj358XfctL4BM. Accessed 20 Aug 2019