Computers and Electronics in Agriculture 162 (2019) 219–234

Contents lists available at ScienceDirect

Computers and Electronics in Agriculture

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

Review

# Deep learning – Method overview and review of use for fruit detection and yield estimation

Anand Koirala[a,*], Kerry B. Walsh[a,*], Zhenglin Wang[a], Cheryl McCarthy[b]

[a] Central Queensland University, Rockhampton, Australia
[b] University of Southern Queensland, Toowoomba, Australia

ARTICLE INFO

*Keywords:*
Convolutional neural network
Precision horticulture
Machine vision
Tree crop
Yield estimation
YOLO

ABSTRACT

A review of developments in the rapidly developing field of deep learning is presented. Recommendations are made for original contributions to the literature, as opposed to formulaic applications of established methods to new application areas (e.g., to new crops), including the use of standard metrics (e.g., F1 score, the harmonic mean between Precision and Recall) for model comparison involving binary classification. A recommendation for the provision and use of publically available fruit-in-orchard image sets is made, to allow method comparisons and for implementation of transfer learning for deep learning models trained on the large public generic datasets. Emphasis is placed on practical aspects for application of deep learning models for the task of fruit detection and localisation, in support of tree crop load estimation. Approaches to the extrapolation of tree image counts to orchard yield estimation are also reviewed, dealing with the issue of occluded fruit in imaging. The review is intended to assist new users of deep learning image processing techniques, and to influence the direction of the coming body of application work on fruit detection.

## 1. Introduction

In any given discipline there are periods of rapid advance and periods of incremental progress. Machine vision and machine learning is in a period of rapid advance. Typically, step advances are catalysed by some combination of expertise, resources and application need, and then diffuse into other application areas. Advances can originate in the agricultural area, e.g., the discipline of near infrared spectroscopy was born from the need to assess forage and grain quality (Norris, 1996), but more often advances occur in better resourced sectors (e.g., medical, security) and diffuse into the agricultural sector. The latter is true of deep learning (multiple layer neural networks) in machine vision, which represent a step advance from algorithms based on hand crafted features such as colour, shape and texture.

Application of deep learning techniques to agricultural applications is nascent, with the 2018 review of Kamilaris and Prenafeta-Boldú (2018) reporting just 40 published studies (taking a broad definition of deep learning), with four reports on the topic of in-field fruit counting. Subsequent progress has been rapid, both in the field of deep learning, and in application to the task of fruit counting. In the current review we seek to extend the review of Kamilaris and Prenafeta-Boldú (2018) in context of developments in deep learning, broadening the topic focus to provide background on the techniques, and narrowing the application topic to that of fruit detection and localisation.

Previous reviews of the task of in-field fruit detection, e.g., Gongal et al. (2015), Kamilaris and Prenafeta-Boldú (2018), Naik and Patel (2017) and Syal et al. (2013), have reinforced the choice of the RGB camera as the detector of choice (based on the practicality of cost and ease of implementation) and discussed use of handcrafted features such as colour, texture and shape in fruit detection. However, these techniques require re-design when used outside of a set of conditions particular to the calibration conditions, e.g., variation in fruit or foliage colour, illumination, camera viewing angle and camera to fruit distance. Gongal et al. (2015) noted that supervised methods based on machine learning yield better results than simple image processing techniques, but require greater computational resources and a greater resource of labelled data for training was required. In recent years, however, high performance GPU has become available, and the task of labelling object in images has become easier with the advent of freely available graphical annotation tools (e.g., LabelImg https://github.com/tzutalin/labelImg). Moreover, many state-of-art deep learning frameworks such as Faster Regional Convolutional Neural Network (Faster R-CNN) (Ren et al., 2015), Single Shot multibox Detector (SSD) (Liu et al., 2016), and You Only Look Once (YOLO) (Redmon and Farhadi, 2018) have scripts to parse the commonly used PASCAL-VOC (Everingham et al., 2010) annotation format for training the network.

* Corresponding authors.
*E-mail addresses:* anand.koirala@cqumail.com (A. Koirala), k.walsh@cqu.edu.au (K.B. Walsh).

https://doi.org/10.1016/j.compag.2019.04.017
Received 20 January 2019; Received in revised form 4 March 2019; Accepted 13 April 2019
Available online 17 April 2019

Thus constraints to practical adoption of the deep learning methods have recently fallen away.

Naik and Patel (2017) briefly reviewed some popular feature extraction methods e.g., Speeded Up Robust Features (SURF) (Bay et al., 2006), Histogram of Oriented Gradients (HOG) (Dalal and Triggs, 2005) and Linear Binary Patterns (LBP) (Ojala et al., 1996) and machine learning algorithms e.g., K-Nearest Neighbour (KNN), Artificial Neural Network (ANN) and Convolutional Neural Network (CNN) for use in fruit classification and grading. Liakos et al. (2018) presented an overview of machine learning (including ANN and deep learning) and its application in the agriculture domain including its use in yield prediction and detection/classification of weed, crop quality and disease. It was noted that the majority of published papers target applications of machine learning in crop management with the most popular models being ANNs. In a more extensive review, Kamilaris and Prenafeta-Boldú (2018) reviewed the use of deep learning methods in agricultural applications in general, concluding that the deep learning methods provide improved detection accuracy than previous image processing techniques.

However, other deep learning techniques have become available since the 2018 reviews were undertaken, particularly the so called 'single shot detectors' that offer improvements in speed (and thus potential for real time application). Further reports on the use of deep learning approaches for fruit detection have also appeared, noting the techniques to generalize very well in real orchard scenes and to be robust to issues such as fruit occlusion and variable lighting conditions for the object detection task (Koirala et al., 2019).

Given the success of the deep learning technique in other application areas and the increasing ease of use, there will be a flood of application reports in the agricultural domain. In an attempt to guide such work, this review has three areas of recommendation that shape its outline:

(i) *Applying deep learning:* This paper is intended to provide a background on machine vision concepts and terminology, an insight of object detection framework and a review of deep learning approaches for fruit detection based on deep learning. An evolution of both frameworks and detectors can be traced in which detection speed and accuracy has markedly improved over a few years. Object detection frameworks include Faster R-CNN, SSD and YOLO, while detectors include Oxford Visual Geometry Group Network (VGGNet) (Simonyan and Zisserman, 2014), Residual Network (ResNet) (He et al., 2016), Zeiler and Fergus Network (ZFNet) (Zeiler and Fergus, 2014). Emphasis is placed on practical aspects that require consideration when adopting standard deep learning models for the fruit detection task. Recommendations are made on what is required to make original contributions to the literature, as opposed to formulaic applications of established methods to new application areas (e.g., to a new commodity), including the use of standard metrics for model comparison involving binary classification.

(ii) *Common image sets:* The deep learning community has benefited from publicly available annotated image datasets such as PASCAL Visual Object Classes (PASCAL VOC); Microsoft Common Objects in COntext (COCO) (Lin et al., 2014), and ImageNet (Deng et al., 2009) which contain thousands of common object classes and millions of images, and are available to developers for model training or for benchmarking object recognition algorithms. Unfortunately, these datasets do not contain orchard images. Kamilaris and Prenafeta-Boldú (2018) also commented on the paucity of publicly available data sets for agricultural applications. The deep learning models trained on the large public generic datasets can be fine-tuned for fruit detection with addition of training data through transfer learning. To facilitate such development, and for benchmark comparison, it is recommended that fruit-in-orchard image sets be made publicly available for all major fruit tree commodities.

(iii) *Orchard yield estimation:* Much published work has focused on improving the accuracy of algorithms to accurately predict the number of fruits within images of tree canopies. Less work has been reported that relates image fruit counts to actual yield of an orchard block. Therefore approaches to the issue of occluded fruit are also reviewed.

The following detail is therefore intended to assist new users of deep learning image processing techniques, and hopefully, if in some small way, influence the direction of the coming body of application work, particularly in context of fruit detection.

## 2. CNN and deep learning – A background

Deep learning with convolutional networks (convNets) are widely used for image processing tasks as convNets can learn translational invariant patterns, allowing detection of objects wherever positioned in an image, and can extract complex visual concepts through detection of a hierarchy of increasingly complex patterns (early layers learn simple local patterns, e.g., edges, while later layers capture more semantic representations of the object, e.g., shape).

A deep learning revolution started when AlexNet (Krizhevsky et al., 2012) won the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (http://image-net.org/challenges/LSVRC/) by a large margin (85% accuracy compared to 74% for the runner up model which was based on traditional Support Vector Machine classifiers (SVM) (Cortes and Vapnik, 1995). The winning entries in subsequent years (ZFNet, VGGNet, GoogleLeNet, GBD-Net, SENet; 2013-17) have all involved deep learning. The top-5 classification error rate decreased for 2015, 2016 and 2017 ILSVRC challenges, to 3.6, 3.1 and 2.3%, respectively, while the average human error rate was 5% (He et al., 2015). The improved accuracy is in general associated with increased model depth, tempered by use of connections between layers. However, training and testing error increase with depth, making deeper models more difficult to train (e.g., 'vanishing gradient' problem; He et al., 2016). This issue has been addressed in newer architectures that use skip connections and residual networks (He et al., 2016).

ZFNet used an architecture similar to AlexNet but with more convolution filters of smaller sizes. A smaller filter size allows capture of information that is more locally distributed in the images, which can lead to more accurate classification/detection results. VGGNet, used even smaller filter sizes and more convolution layers (16–19 layers) but a huge memory requirement rendered it computationally expensive. The use of more layers (deeper model) sacrificed computation speed for accuracy. GoogLeNet (Szegedy et al., 2015) introduced inception modules and was deeper (22 layers), but computationally efficient. A basic inception module consists of filters of multiple sizes operating on the same level. ResNet used a residual learning framework to achieve efficient training of even deeper (up to 152 layers) networks. ResNet used residual blocks that featured 'identity shortcut connection' which allowed the information to flow without being lost ('vanishing gradient' problem) in the deeper networks. Gated Bi-Directional Network (GBD-Net) (Zeng et al., 2018) is a CNN architecture which utilizes the relationship among the features of different resolution and candidate support regions to detect objects in image. GBD-Net is an attempt to integrate local and contextual visual information for more accurate object classification. In the Squeeze-and-Excitation Network (SENet) (Hu et al., 2017) a squeeze-and-excitation (SE) block was added to convolution layers to boost representational (classification) power. The SE block dynamically models the interdependencies between convolutional feature maps by exciting relevant features while suppressing irrelevant features.

Object detection frameworks, which combine both classification and localization into a single system to detect and draw boxes around objects in images, have also evolved markedly in recent years. Region-

CNN (R-CNN) (Girshick et al., 2014), released in 2012, combined heuristic region proposal - selective search (Uijlings et al., 2013) with Convolutional Network (ConvNet) feature extractors for object detection. OverFeat (Sermanet et al., 2013) implemented feature extraction from multiple square grid cells over multi-scale input image, without the need of separate region proposal. Thus OverFeat was faster than R-CNN but was less accurate in object localization. Spatial Pyramid Pooling net (SPPNet) (He et al., 2014) introduced adaptively-sized pooling to extract features from a common global feature volume, reporting SPPNet to operate faster than R-CNN. MultiBox (Szegedy et al., 2014) proved that ConvNets are more efficient for region proposals. In Fast R-CNN (Girshick, 2015), the SPP layer was replaced by a fixed size region of interest pooling (ROIPooling) layer, enabling a speed increase over R-CNN. In 2015, Faster R-CNN replaced the selective search (heuristic region proposal) in Fast R-CNN with a region proposal network (RPN) and was end-to-end trainable (i.e., all model parameters were simultaneously trained using a multi-task loss function). In 2017, He et al. (2017) introduced Mask-RCNN as an extension to Faster R-CNN for instance segmentation (i.e., location of exact pixels followed by masks for each object inside the bounding box).

While previous object detection framework were 2-stage methods (region proposal stage followed by classification stage), YOLO was developed as a one stage (single shot) unified object detection model. In YOLO, a single CNN is able to simultaneously predict multiple bounding boxes and their class probabilities. The 'single shot detector' SSD implemented prior boxes (subsets of fixed sized anchor boxes) at different resolutions of feature maps at different levels inside the network for multiscale training, making it a very fast (faster than Faster R-CNN) yet accurate framework for object detection. YOLOv2 (Redmon and Farhadi, 2017) gave a speed and accuracy improvement on YOLOv1 (Redmon et al., 2016) through introduction of a pass-through layer, higher resolution classifier and anchor boxes, and achieved nearly the same mean Average Precision (mAP) as Faster R-CNN and SSD on the PASCAL VOC dataset. In 2017, RetinaNet (Lin et al., 2017), a one-stage detector, outperformed all the previous one-stage and two-stage detectors available at that time in terms of both speed and accuracy. YOLOv3 (Redmon and Farhadi, 2018), released in 2018, is deeper than the previous YOLO variants. This architecture achieves an accuracy similar to SSD and RetinaNet, at three and four times the speed, respectively (Redmon and Farhadi, 2018). The various YOLO architectures offer a trade-off between speed and accuracy. Pre-trained YOLO models are available from the github repository https://github.com/AlexeyAB/darknet.

## 3. Object (fruit) detection in images

Tree fruit yield estimation by machine vision typically involves the steps of classification of possible regions as fruit objects (using, e.g., colour thresholding, key-point extraction or convolutional filtering) followed by location of individual object (e.g., blob segmentation, shape fitting and bounding box regression). The segmented blob or bounding boxes can then be counted to provide the fruit number in images. Object detection represents one of the most important step towards yield estimation of fruits. The following section provides the information on what constitutes an object detection framework and details the feature extraction methods.

### 3.1. Framework steps

An object detection framework (object classification plus localization) generally follows the following steps:

- Image pre-processing (image resizing, change of colour space, image data normalization etc.)
- Generation of hypotheses (generate possible regions containing objects e.g. test patch at each location of sliding window, voting from patches or key-points, and region based proposals using selective search algorithms)
- Score hypotheses (assign level of probability/confidence for an object to belong to a particular class/category using classifiers)
- Resolve detection (remove low scoring hypotheses, e.g., using class confidence threshold, and suppress multiple redundant detection, e.g., using Non-Maximal Suppression (NMS), with a goal to assign one box per object).

### 3.2. Feature extraction

In traditional feature extraction, 'handcrafted' features (e.g., colour, shape, texture, intensity) are used to compute class membership. As colour of fruit can vary, segmentation should be based on a wide colour window, and additional features such as shape and texture should be used (Gongal et al., 2015).

The modelling of object class is made difficult by variation in illumination, viewing angle, pose of the object (orientation) and position of object (tight clustering, occlusions). Traditional image segmentation techniques that rely on morphological operations in binary images are affected by strong shadows and occlusion from stem and leaves which split the fruit image into smaller segments (e.g., Payne et al., 2013; Annamalai et al., 2004). These techniques also tend to count fruit clusters as a single blob (fruit) because of pixel connectivity (Annamalai et al., 2004). Algorithms such as Circular Hough Transform (CHT) for apple (e.g., Bargoti and Underwood, 2017b; Sengupta and Lee, 2014; Stajnko et al., 2009) and citrus (e.g., Choi et al., 2015) and Random Hough Transform (RHT) for mango (e.g., Kadir et al., 2015; Nanaa et al., 2014), have been widely used to fit circular and oval shapes around possible regions in an image to segment fruits.

Features such as HOG (Dalal and Triggs, 2005), LBP (Ojala et al., 1996), Scale-Invariant Feature Transform (SIFT) (Lowe, 1999), SURF (Bay et al., 2006) and Haar-like features (Viola and Jones, 2001) have been widely used in traditional object detection/classification methods in computer vision.

HOG features (Fig. 1) have been used in fruit detection by a number

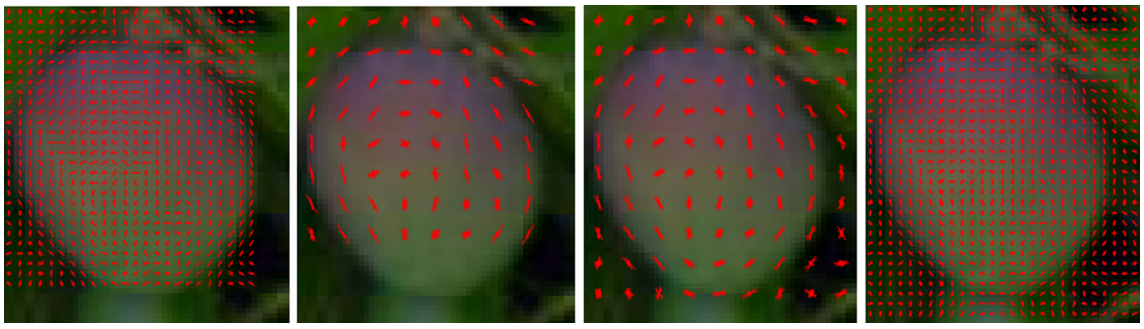

**Fig. 1.** HOG features, using different block and cell sizes, overlaid on the fruit image.

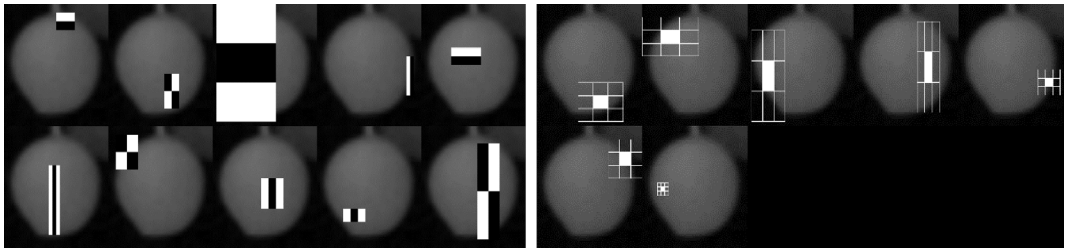

**Fig. 2.** Visualization of some Haar features (left panel) and some LBP features (right panel) used by a cascade classifier for mango fruit detection.

of authors. For example, Pothen and Nuske (2016) used a modified HOG feature extractor for detecting apple and grape fruit in images. Sa et al. (2015) compared HOG and LBP texture features for capsicum fruit detection.

Cascade classifiers with Haar-like features (Fig. 2) have also been adopted for fruit detection tasks. For example, Zhao et al. (2016) attempted detection of ripe tomatoes in a greenhouse setting with a cascade classifier using Haar-like features followed by average pixel value (APV) based colour analysis. Similarly, Wachs et al. (2009) implemented the Viola Jones cascade classifier with Haar-like features for real time detection of green apples within a tree canopy using RGB-IR images.

Other feature extractors are also available. For example, for green citrus fruit detection, Kurtulmus et al. (2011) used the eigenface (Turk and Pentland, 1991) ('eigenfruit') approach in combination with color and a circular Gabor filter.

### 3.3. CNN as a feature extractor

With deep learning methods there is no requirement for manual hand-crafting of features, although such features can be used as a pre-processing input. During training, the CNN automatically undertakes the task of feature selection and classification, and models can be developed to detect many object classes in images. For example, Redmon and Farhadi (2017) reported a YOLOv2 model recognizing 9000 common object categories. The tasks involved in use of the traditional segmentation (handcrafting) method to detect such a large number of classes would be daunting.

A basic CNN consists of an input and an output layer, with intervening convolution and pooling/sub-sampling layers (Fig. 3). Image data fed to the input layer passes through the intermediate layers to produce a vector of distinct features at the output layer. During training, a CNN develops filters in the convolution layers that extract useful information for different object classes. Feature extraction occurs in the convolution layers, the output of which can be visualized in an attempt to understand the features utilised in the model. Each layer down-samples the image data. Thus, in general, more complex and semantic features such as shape and patterns are learned in deeper layer while basic features such as colour, edges and lines are learned in the early layers.

The class activation map from the final convolutional layer can be visualized (Fig. 4) using methods such as Gradient-weighted Class Activation Mapping (Grad-CAM, Selvaraju et al., 2017). Visualisation allows for some interpretation of the model function. In the case of Fig. 4, the model visualisation indicates weighting of the fruit regions of the images.

The CNN feature extractor is used by object detection frameworks for classification tasks, using box regression for object localization. For example, AlexNet has been implemented in RCNN and Overfeat and ZFNet is used in SPPNet, VGG-16 is used in Fast R-CNN, Faster R-CNN and SSD, Darknet-19 is used in YOLOv2, and Darknet-53 is used in YOLOv3. However, depending upon the application, feature extractors can be used interchangeably. For example, ZFNet could be used in a Faster R-CNN framework.

## 4. Deep learning object detection framework

The short developmental history of deep learning frameworks for image object detection are reviewed in the following sections, finishing with consideration of the 'state of art' one stage detectors.

### 4.1. CNN sliding window detectors

Early CNN based object detection frameworks such as Overfeat used

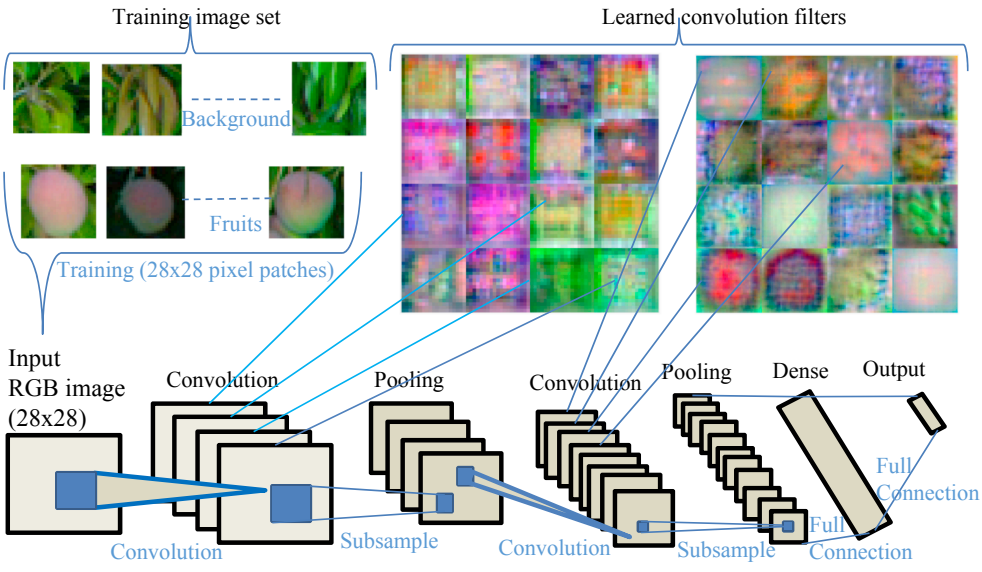


**Fig. 3.** The LeNet (LeCun et al., 1998) CNN classifier accepts 28 × 28 pixel images and processes through five layers. Multiple feature maps are created in each convolution layer. Pooling layers subsample data from the convolution layers. Example input training images are provided in the top left of the graphic, and visualization of the output of example convolution filters (feature extraction) after training is provided in the top right.

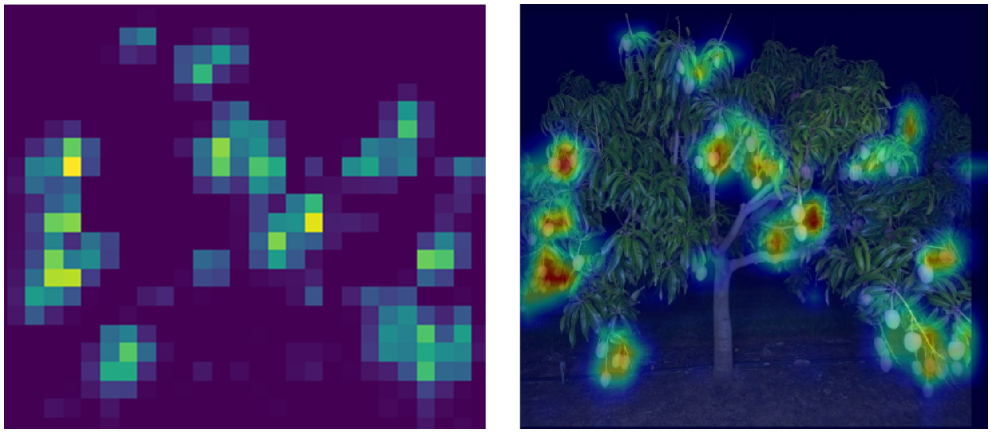

**Fig. 4.** Grad-CAM visualization of activation map of the final convolutional layer of a deep learning 'Xception' (Chollet, 2017) model (trained to directly predict fruit count from input images) as heat-map (left) and a heat map superimposed on input image (right).

a sliding window approach where the classifier is run at evenly spaced locations over the image. A large number of patches are generated at each position of the sliding window, with each patch classified as containing an object or not. For multi-scale detection, patches are generated at each scale. Feeding all available patches to a CNN slowed down the object detection framework.

R-CNN replaced the sliding window method with relatively faster heuristic Selective Search algorithm (Uijlings et al., 2013) to filter out some regions, feeding only the potential region proposals into the CNN. Fine-tuned CNN was used to extract features from each region proposal for a SVM classifier. However, the feeding of nearly 2000 region proposals per image through a CNN rendered this method slow.

### 4.2. Two stage detection

To decrease detection time, it is desirable to feed the whole image to the CNN model in one pass to generate a final feature map. A two stage approach (region proposal stage followed by classification/detection stage) can deliver this goal, as achieved in Fast R-CNN and Faster R-CNN.

**Fast R-CNN** was developed to improve on the detection speed of R-CNN. Feature extraction was performed on the whole image, and region proposals were generated on the final feature map (Fig. 5). Region of Interest (RoI) pooling was implemented to obtain a fixed sized feature vector from all cropped feature maps for classification. Detection speed was increased by use of a single Softmax layer, which was sufficient for prediction instead of training several SVMs. However, the generation of region proposals using Selective Search remained as a bottleneck to further speed improvement.

**Faster R-CNN** replaced the heuristic selective search method of Fast R-CNN with a Region Proposal Network (RPN), which uses CNN and anchor boxes for speed improvement. RPN slides a $3 \times 3$ convolution window on the final feature map and at each window location considers 9 different anchor boxes (composed of 3 scales and 3 aspect ratios) to generate region proposals (Fig. 6). These proposed regions are filtered based on an objectness score (probability the proposed region contains object) and passed to next stage (essentially a Fast R-CNN) for object detection. However, despite the improved detection speed achieved in Faster R-CNN compared to Fast R-CNN, the pipeline was still too slow to apply on real-time videos/streaming.

### 4.3. One stage detection (Single shot detectors)

To achieve further improvement in speed, SSD and YOLO removed the region proposal stage and the CNN was designed to consider dense sampling of possible object locations for object detection.

**Single Shot MultiBox Detector (SSD)** simultaneously predicts the object class and bounding box on the image making it faster than Faster R-CNN. SSD feeds the input image through series of convolution and pooling layers to generate feature maps at different scales (Fig. 7). A $3 \times 3$ convolutional window at each location of feature maps evaluates a small set of default anchor boxes (separate set of boxes of different aspect ratios at different resolution of feature maps) for which SSD simultaneously predicts the class probabilities and bounding box offsets for different scales. There can be different number of detections at each scale. Due to the multiscale detection approach, SSD is very effective in detecting objects of different sizes in the image.

**YOLO** (You Only Look Once) or YOLOv1 is a single shot detector in which a fully convolution neural network converts the input image to a tensor of scores for object detection. The prediction of class probabilities and bounding box coordinates from the final feature map in a single forward pass through the CNN makes YOLO one of the fastest object detection methods. YOLO divides the input image into grid cells (each grid cell is 1/32 times the network input resolution), and each grid cell is responsible for detecting object.

YOLOv1 does not use anchor boxes like Faster R-CNN and SSD, but instead directly predicts two bounding boxes and one class per grid cell. Since YOLOv1 learns to predict boxes directly from the image data it

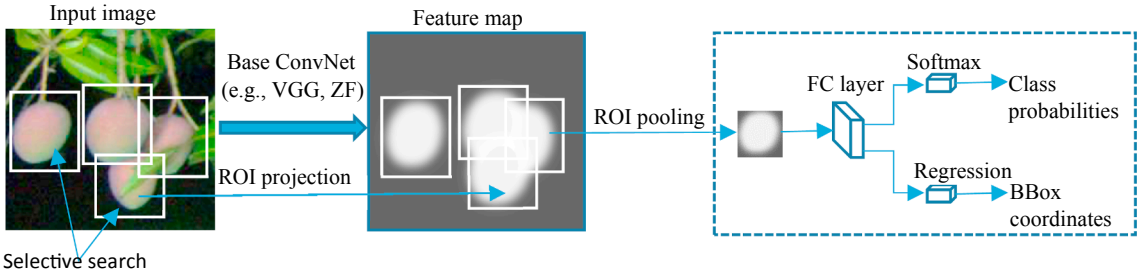


**Fig. 5.** The fast R-CNN object detection framework.

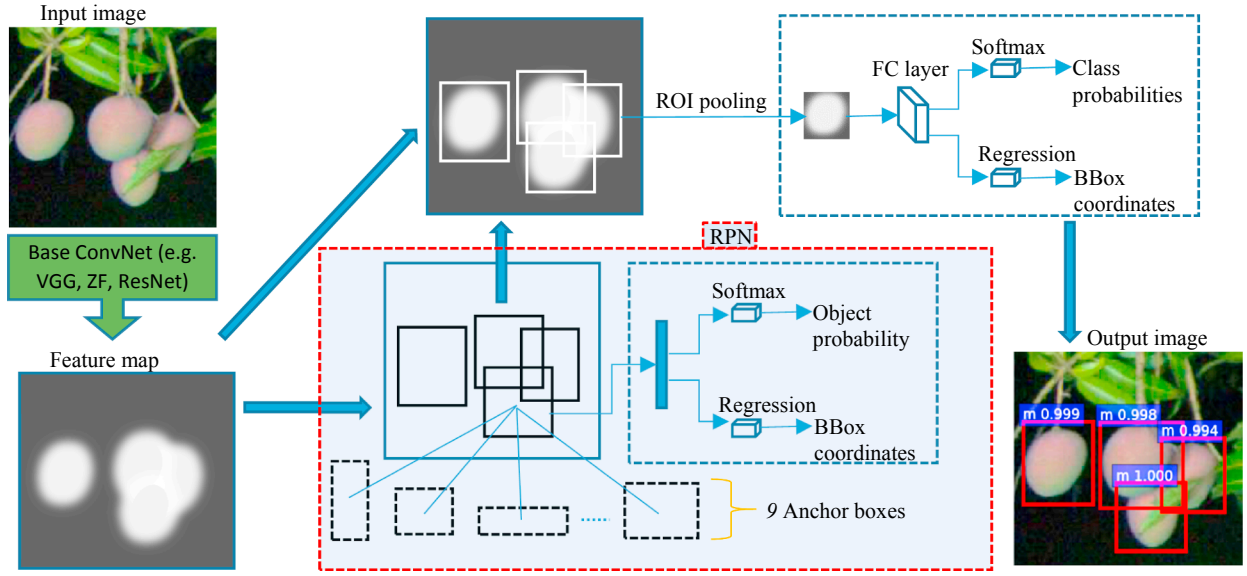


**Fig. 6.** The Faster R-CNN object detection framework (RPN plus Fast R-CNN). Output image shows bounding boxes (Bboxes) on detected objects of class 'm' (mango), with associated probability of correct classification.

produced many localization errors (for small objects and objects in groups). YOLOv2 provided an accuracy and speed improvement on YOLOv1. YOLOv2 achieved nearly the same mean Average Precision (mAP) as Faster R-CNN and SSD on PASCAL VOC dataset and was still the fastest detector. Redmon and Farhadi (2017) reported the object detection speed of Faster R-CNN (VGG-16), SSD-300 and YOLOv2-288 at 7 fps, 46 fps and 91 fps respectively. YOLOv2 introduced anchor boxes, with prediction of more than 1000 boxes per image, compared to just 98 for YOLOv1. YOLOv2 also implemented a pass-through layer that concatenates the features from a higher resolution layer to lower resolution layer, providing the advantage of small object detection from the finer grained features (Redmon and Farhadi, 2017) (Fig. 8). The backbone feature extractor for YOLOv1/v2 was the Darknet-19 framework which has 19 convolution layers, mostly 3 × 3 filters similar to VGG net.

***YOLOv3*** was an improvement over YOLOv2 in terms of detection accuracy. According to Redmon and Farhadi (2018), YOLOv3 is as accurate as SSD and RetinaNet, but 3.0 and 3.8 times faster, respectively. YOLOv3 implements similar concept to feature pyramids (Lin et al., 2017) for multiscale box prediction. The backbone feature extractor for YOLOv3 is Darknet-53, which has 53 convolution layers (successive 3 × 3 and 1 × 1 convolutional layers with skip connections similar to ResNet) (Fig. 9).

## 5. Network and model training

The following sections cover the typical steps involved in developing and testing a deep learning model. Emphasis is placed on appropriate data set structure, determination of the appropriate size of the training set, the use of pre-existing models and the image size.

### 5.1. Training and testing requirements

The creation and validation of a model requires (i) training, (ii) validation (tuning) and (iii) test sets of images and associated annotation files containing bounding box co-ordinates and the class of the object. Manual bounding box annotation is tedious and prone to user bias in ground truth labelling, especially for images having a large number of objects and clusters with a high degree of occlusion (Bargoti and Underwood, 2017a). The quality of annotated training image sets can thus be an issue, a factor which should drive the use of shared image sets.

The validation set is used for tuning of model parameters such as confidence and overlap thresholds before the model is applied on the test set, e.g., Bargoti and Underwood (2017a) and Koirala et al. (2019). The test set should be "independent" of the training and validation sets, where independence refers to different orchards or seasons for the

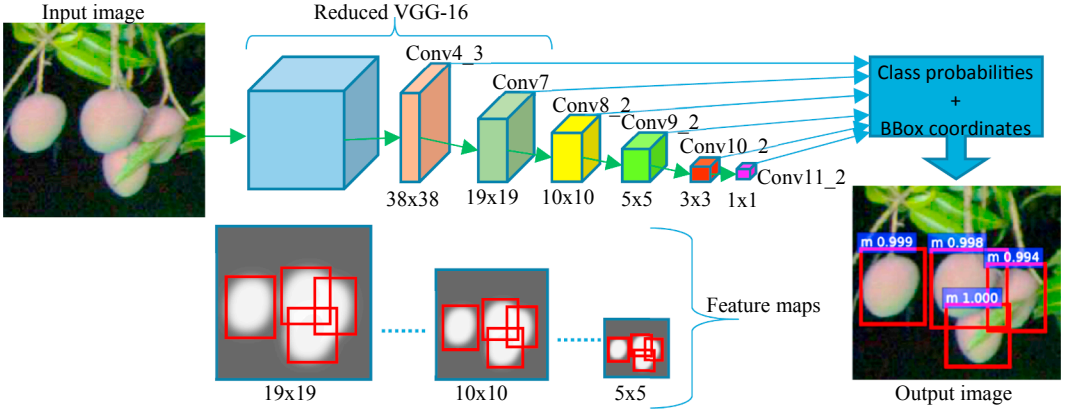


**Fig. 7.** SSD object detection framework. 'Conv' refers to 'convolution layer'. Output image displays bounding box on ROI, classification result ('m' for mango) and probability score.

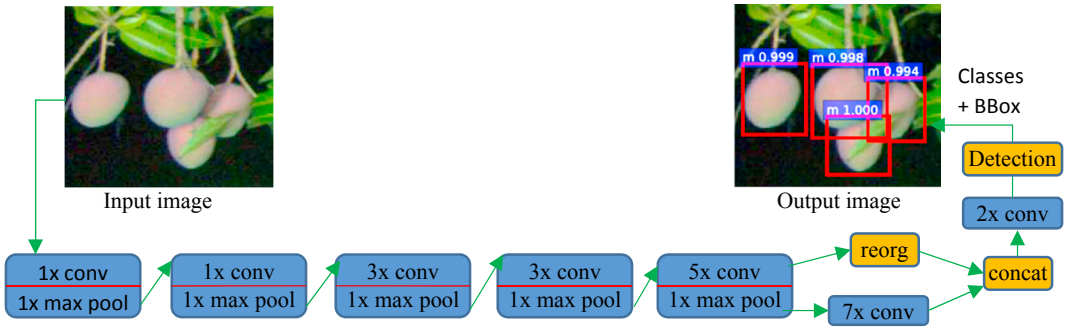


**Fig. 8.** YOLOv2 object detection framework. 'Conv' refers to 'convolution layer', 'concat' to 'concatenation layer' and 'reorg' refers to 'reorganize/route'. Output image displays bounding box on ROI, classification result ('m' for mango) and probability score.

machine vision task of detection of fruit on canopy. Unfortunately, many published papers used an image set collected in one event for these three functions. It is a common practice to reserve a randomly selected 10 to 20% of an initial set for use as a validation set. The reservation of images from one row of trees as the test set is better practice, but ideally the test set should involve images of another orchard or season, to emulate the real world condition for model use.

Various object detection methods require different data formats for processing image data during training. Many object detection systems include the scripts to parse the annotation files in PASCAL VOC format to their own formats. For example, the online code repository of Faster R-CNN, YOLO and SSD have scripts to parse PASCAL VOC annotation files. Therefore, structuring dataset and directories similar to PASCAL VOC dataset format can avoid the need to change the scripts that are ready to parse annotations to framework specific format. The following recommendations are offered as practical advice in the creation of models:

Object detection frameworks train object classes against background while trying to match the detected box to the ground truth box for maximum overlap. Therefore, ground truth boxes should be tight enough to cover the object and some background around the object perimeter (Fig. 10). For overlapping (clustered) fruit, emphasis should be put on reducing the box overlaps among neighbouring fruits as much as possible to avoid merging into single detection from NMS (Koirala et al., 2019).

To enable another user to replicate the development of a deep learning model, it is necessary to detail the process followed in organizing data and in pre-processing of data (e.g., colour space conversion, histogram normalization, image resizing/cropping, etc.). This includes consideration of the number of training/test/validation images used.

### 5.2. Number of training images

The minimum number of images required for training a deep learning model depends on the visual complexity of the image for object detection and the deep learning models used for learning. An assessment of minimum number of training images sufficient to capture the variation in the test set can be done as described by Bargoti and Underwood (2017a) and Koirala et al. (2019) for mango detection in day-time and night-time images, respectively. In this approach, AP is calculated for models developed using an increasing number of images sampled from the training set. Bargoti and Underwood (2017a) report that for the image set used, AP asymptoted with use of ~729 apple fruit images, while AP for models for detection of almond and mango did not stabilise with full available set of 385 and 1154 training images, respectively. Koirala et al. (2019) imaged the same mango orchard as Bargoti and Underwood (2017a), but used different imaging hardware, with images collected at night under artificial light. An AP plateau was achieved with fewer (approximately 400) training images, a result attributed to a lower level of variation and background noise in the images. Data augmentation techniques (e.g., random crop, flips, zoom, change in hue, saturation etc.) can be used to expand the variation in training images. This augmentation helps to increase generalizability of the model and to reduce the chance of data overfitting. Bargoti and Underwood (2017a) reported that an AP score of 0.86 was achieved for apple detection in images with just 100 training images when data augmentation was used, compared to 300 images without data augmentation. Large numbers of artificial synthetic images can be generated for training neural networks, as demonstrated for a fruit detection application by Rahnemoonfar and Sheppard (2017). All deep learning object detection frameworks provide some level of data augmentation which can be selectively turned on or off.

In summary, different numbers of training images are required for

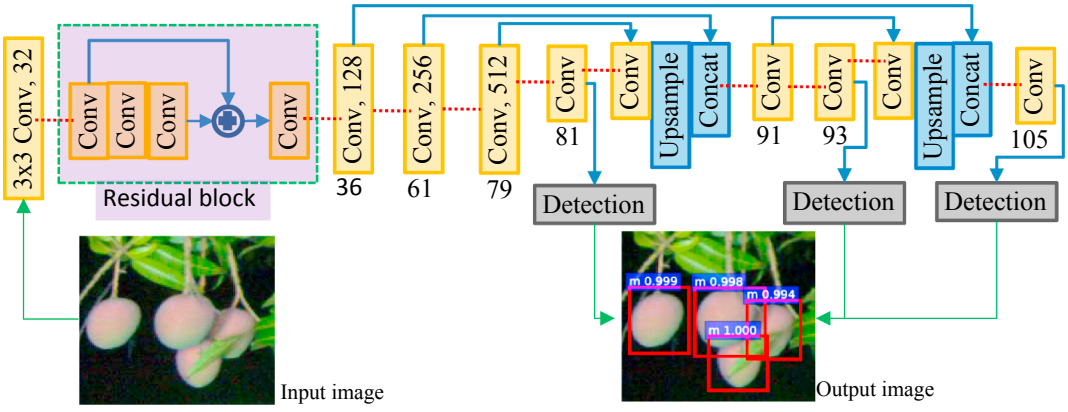


**Fig. 9.** Block diagram of YOLOv3 architecture. 'Conv' refers to 'convolution layer' and'concat' to 'concatenation layer'. Output image displays bounding box on ROI, classification result ('m' for mango) and probability score.

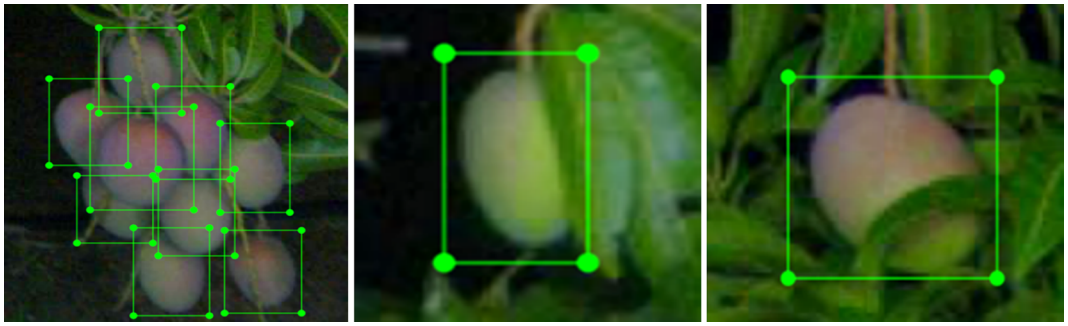

**Fig. 10.** Examples of ground truth labelling of individual fruit using LabelImg software in scenes with varying levels of occlusions by other fruits or leaves.

different crops and different lighting conditions, in consequence of the amount of image variation in each case. Therefore, training data should be carefully chosen such that it is relevant to the problem space and has enough variation in relation to the production scope or context of deployment. The availability of labelled training data can limit the choice of network architecture to be used. Larger networks can be more accurate, but will require more training data. However, lack of data should not be a defence for the lower performance of a deeper model.

Publicly available annotated fruit image datasets would facilitate comparison of models, with use of the same training and validation data. Some annotated fruit datasets are publicly available. For example, the 575 images at various resolutions of avocado, orange, apple, mango, strawberry, rockmelon and capsicum fruits acquired in glasshouse and through Google image search is available at http://enddl22.net/wordpress/datasets/deepcrops-datasets-and-annotation-tool (accessed on 3/01/2019), as reported by Sa et al. (2016). The image data used by Bargoti and Underwood (2017a) (1120 apple fruit images at 308 × 202 pixels, 1964 mango fruit images at 500 × 500 pixels and 620 almond fruit images at 308 × 202 pixels) is available at https://data.acfr.usyd.edu.au/ag/treecrops/2016-multifruit (accessed on 3/01/2019). Koirala et al. (2019) (http://hdl.cqu.edu.au/10018/1261224) provides 1730 mango fruit on canopy images (at 612 × 512 pixels). These images were collected at night. The Fruit-360 dataset by Mureşan and Oltean (2018) (https://github.com/Horea94/Fruit-Images-Dataset: accessed on 1/03/2019) contains 65,429 images (100 × 100 pixels) of popular fruits (95 fruit classes) and is intended for fruit recognition applications. The MangoNet semantic dataset by Kestur et al. (2019) (https://github.com/avadesh02/MangoNet-Semantic-Dataset: accessed on 28/02/2019) contains 49 daytime mango fruit on canopy images (at 4000 × 3000 pixels).

### 5.3. Transfer learning

For efficient and more stable training of deep learning models, it is common practice to employ transfer learning (also referred to as or fine-tuning) on a network pre-trained on a large dataset (e.g., ImageNet). Transfer learning allows the re-use of existing parameters (convolution weights) from a model trained on large datasets for training new models from relatively less number of training images. Publically available datasets such as ImageNet (450k images, 200 classes), PASCAL VOC (12k images, 20 classes) and COCO (120k images, 80 classes) provide labelled data of common objects in images, free for training and benchmarking object detection models. During transfer learning, the weights from the earlier layers of pre-trained model are copied to a new model. These weights contain information about the basic features found in common objects, such as colour, shape, edges, and lines. The final classification layer, which is responsible for high level classification of the object into class categories, is not transferred. The model is then trained further on new classes.

If there is a sufficient number of training images available for a particular application, the transfer-learning process holds no advantage. Bargoti and Underwood (2017a) reported that there was no significant performance difference between weights initialized from ImageNet and from other orchards (almond and mango) for training a Faster R-CNN model to detect apple on images. Koirala et al. (2019) also reported no significant performance gain for a mango fruit detection model (MangoYOLO) trained with over 1000 image tiles when initialized with the COCO pre-trained weights.

### 5.4. Image and object sizes

The optimum image/tile size should be established for the imaging conditions (camera resolution, lens, camera to canopy distance, fruit size, etc.) and the machine vision model architecture used. A larger object size (pixel number) in images is desirable for proper object detection and localization, but higher resolution incurs greater computational time and memory. Zhao et al. (2016), Cheng et al. (2017), Hung et al. (2015) and Sengupta and Lee (2014) down-sized images for computational efficiency in fruit detection applications.

Image resizing (subsampling) is inherent in most deep learning architectures. The CNNs have a predefined input size (network input resolution). All input images will be resized to the network input resolution before feeding through the network. For example, SSD-300 resizes all input images to 300 × 300 pixels before further processing inside the network. This step reduces the size of the objects in the image, which can impact the performance of object detection models. Moreover, there are several subsampling layers inside a CNN which cause further resizing of the input image. For example, CNN feature extractors such as ZF and VGG, as used in the object detection frameworks of Faster-RCNN and SSD, use a subsampling factor of 16. Thus, object pixel size is further decreased in the final feature map, providing little pixel information for detection. Network architecture can be changed to accept images of higher resolution, but at the cost of higher computation and training memory requirement.

Close-up views of fruit from Google Images or from imaging of fruit in glasshouses, e.g., as used by Sa et al. (2016), provide high resolution detail of the fruit. Such images can tolerate a subsampling factor of 16 and still have objects (fruit) of a reasonable pixel dimension. In comparison, canopy orchard imagery provides relatively less resolution for fruit in the image, unless a very high resolution camera is used. Use of tiles from the image rather than the whole image is useful in decreasing the effect of downsizing. Splitting of large images into sub-images also decreases the fruit count per image, helping to reduce human labelling errors. Moreover, object detection frameworks trained on tiles (e.g., 500 × 500 pixels) can be used for inference on whole images (e.g., 2048 × 2048 pixels) without need for further training (Koirala et al., 2019). Alternatively, to reduce memory requirement, a tiling approach can also be followed during detection, as used by Bargoti and Underwood (2017a). In this process, detections are performed using a sliding windows on sub-sections of the image, with thresholding and NMS applied over fused output on large images.

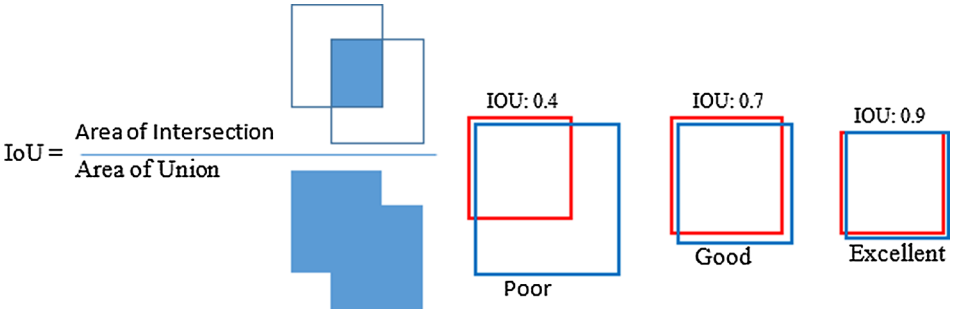


**Fig. 11.** Definition of IoU in the object detection task using bounding box annotations.

## 6. Performance assessment

The algorithm of a new deep learning based machine vision system should be appropriately documented to enable a skilled practitioner to replicate the work, and the performance of the model must be evaluated relative to existing best practice. There are a set of parameters and metrics that are commonly used in machine vision evaluation that should be reported in any published study.

### 6.1. Parameters

#### 6.1.1. IoU

For the object detection task, the Intersection over Union (IoU, also known as the 'Jacard index' is the ratio of the area of overlap to the area of union between detected and the ground-truth bounding boxes (Fig. 11). IoU can vary from 0 (no overlap) to 1 (full overlap). This metric is set for use in model training, with use during testing by the NMS to suppress multiple redundant detections. The object detection challenges of PASCAL-VOC and ImageNet both set an IoU of 0.5 of detection on ground-truth bounding as valid detections. Any detections with an IoU between detected box and ground truth annotation greater than the set IoU threshold is considered as a positive detection. Detections with an IoU less than threshold value are false detections.

Research reports should report the IoU used in their work, and may seek to optimise the value used. For example, for pepper fruit detection, Song et al. (2014) considered a detection to be a true positive when IoU > 0.5 (between predicted and ground truth boxes). In contrast, Bargoti and Underwood (2017a) used an IOU > 0.2 to better detect small fruit. Sa et al. (2016) also justified use of an IOU > 0.4 as the threshold for detection of relatively small fruit size in their image dataset. Similarly, the non-maximal suppression (NMS) threshold value (the IoU used to remove the redundant overlapping detections in output images) should be reported for replication of the work.

### 6.2. Evaluation metrics

#### 6.2.1. Useful metrics

There are a range of metrics useful for the characterisation of model performance in the application of tree fruit detection in canopy images. For example, the review by Kamilaris and Prenafeta-Boldú (2018) tabulates values for classification accuracy, Precision, Recall, F1 score, L2 error, IoU, ratio of estimated to actual fruit count per image and root mean square of error (RMSE) and mean residual error (MRE) on estimated compared to actual fruit count per image.

#### 6.2.2. Precision and Recall

The fruit detection task is a binary classification problem with the following results:

- True positive (TP): Fruit detected as fruit
- True negative (TN): Background detected as background. This is not applicable for deep learning object detection frameworks like (Faster R-CNN, SSD and YOLO) which do not require labelling of the background class.
- False positive (FP): Background detected as fruit
- False negative (FN): Fruit not detected

$$Precision(P) = \frac{TP}{TP + FP} \quad (1)$$

$$Recall(R) = \frac{TP}{TP + FN} \quad (2)$$

The term *Precision* gives the number of true detections out of total detections, while *Recall* gives the number of true detections out of total ground truth annotations (Fig. 12).

Both *Precision* and *Recall* measures are essential to characterisation of the performance of an algorithm on an image set. A high *Precision* is sought when there is high cost associated with false positives. A high *Recall* is sought when there is high cost associated with false negatives. *Precision* and *Recall* values are affected by the IoU threshold between the predicted and ground truth box (Fig. 13). *Precision* and *Recall* values can be optimized by changing the IoU threshold or limiting number of detections (changing confidence threshold and NMS threshold). For a good model, *Precision* remains high as *Recall* increases, i.e., the model is able to detect a high proportion of true positives before it starts collecting false positives.

#### 6.2.3. Average precision

The area under the P-R curve (known as Average Precision, AP) can be used as a single metric to summarize the performance of the object detection model (Fig. 13). For example, Bargoti and Underwood (2017a) and Koirala et al. (2019) used AP in assessment of performance of deep learning models in context of the number of training images. A model with high Precision at all levels of Recall will have a high AP score, while methods that return a high Precision with a subset of detections will not.

Multi class object detection challenges (like PASCAL VOC) commonly use mean Average Precision (mAP) as metric to evaluate model performance. mAP is the mean of AP computed over all classes.

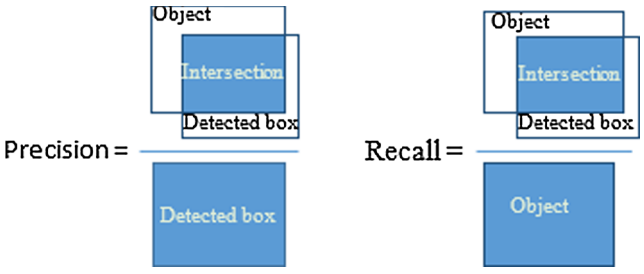


**Fig. 12.** Interpretation of *Precision* and *Recall* for object detection task using bounding box annotation. *Precision* is the proportion of detected boxes that matched the ground truth boxes. *Recall* is the proportion of detected boxes that matched the ground truth boxes on all ground truth boxes.

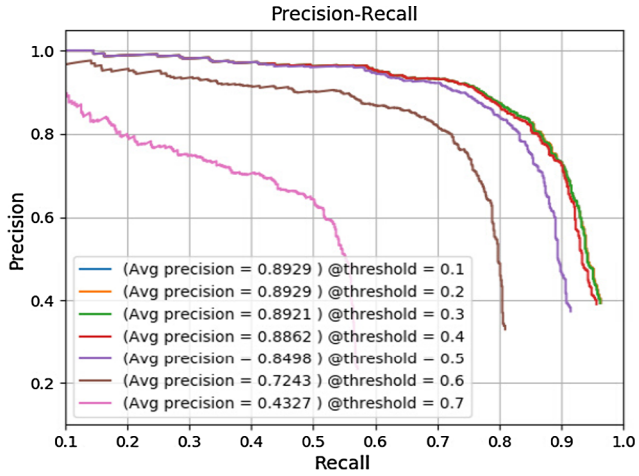


**Fig. 13.** Precision-Recall curves for a Faster R-CNN with ZFNet model used to detect mango fruit in whole tree images, obtained for varying IoU thresholds (0.1–0.7) at a NMS threshold of 0.5.

#### 6.2.4. F1 score

For any classifier (object detection model), there is a trade-off between Precision and Recall, and a model can be optimised (e.g., through change in class confidence threshold and NMS threshold) for either higher Precision or Recall, depending on the detection task.

F1 is the weighted average (harmonic mean) of Precision and Recall (Eq. (3)), calculated from the point on the P-R curve where P and R have higher and identical values. The F1 score varies between 0 (worst) and 1 (best). If the model is tuned in favour of either Precision or Recall, the F1 score will decrease.

$$F_1 = 2 * \frac{precision * recall}{precision + recall} \quad (3)$$

#### 6.2.5. Fruit detection evaluation

The similarity of fruit and foliage features can cause many false positives detections, while false negatives can result from partial occlusion and poor illumination. The success of a model for fruit detection in orchard scenes can thus be evaluated in terms of Precision, Recall, AP and F1 score.

A recommendation is made for use of F1 score as an overall performance measure, allowing fruit detection models/methods to be compared and benchmarked with a single metric. The metric is already in common use to compare fruit detection models, e.g., Bargoti and Underwood (2017a), Sa et al. (2016), Pothen and Nuske (2016) and Koirala et al. (2019).

## 7. Architecture and model optimization

A 'network architecture' is defined by its structure in terms of number of layers, filters and connections. The network also contains several hyper-parameters (e.g., learning rate, momentum, weight initialization and activation function) which determines the performance of a trained model. A model is created through the process of training a network using a training image set. Both the network and the model can be optimised for a given application.

### 7.1. Architecture optimization

Many published works in precision agriculture compare established network architectures for a particular application. In general, a deeper network architecture with more layers in CNN will improve prediction accuracy and achieve higher accuracy compared to a shallower network. For example, VGG is a deeper model with higher accuracy but requires greater memory and computation time than ZF-Net, as demonstrated for a fruit counting application by Koirala et al. (2019) and Bargoti and Underwood (2017a). Such comparisons have practical merit, but to advance the discipline some optimisation of the model architecture should be attempted. Of course, new architectures should be benchmarked to an existing, well reported one, with Faster R-CNN (VGG) acting as the current 'standard'.

Existing CNN architectures can be modified depending upon the application task to optimise performance in context of speed, accuracy and memory requirement. For example, Sa et al. (2016) altered the VGG structure to accept four channels (RGB and NIR). Rahnemoonfar and Sheppard (2017) modified Inception-ResNet model, achieving an accuracy of 91% compared to 76% with the original inception-ResNet. Koirala et al. (2019) modified the YOLO architecture to create an architecture deeper than YOLOv1 but shallower than YOLOv3, for memory and speed optimization in the task of mango fruit detection.

To advance the common scientific base, a published report on performance of a new architecture should be accompanied by a clear description of the algorithm, and a performance comparison to an existing standard architecture. If an existing architecture is being used, the report should detail the source of the open-source code used for compiling the deep learning models. Ideally, the officially released algorithm codes from published papers should be implemented, to allow for replication of work.

### 7.2. Model optimization

In the process of training a given architecture, a model is created with weightings unique to the training set used. Parameters such as learning rate and momentum of the network and the number of filters in each layer can be varied, depending on the visual complexity of the object class to be modelled, while NMS and class confidence thresholds can be varied to obtain the desired detection output (Koirala et al., 2019).

#### 7.2.1. Class confidence scores

The class confidence (or probability) score is a numeric value (0–1) assigned to each detection describing the confidence or probability of a detected object belonging to a particular class (Fig. 14). If the confidence score threshold is relaxed (set low) many detections will be accepted (increasing TP and FP) (Fig. 15). The confidence cut-off (threshold) must be selected for the application.

#### 7.2.2. Non-maximum suppression

NMS is a common technique used by various object detection frameworks to suppress multiple redundant (low scoring) detections with the goal of one detection per object in the final image (Fig. 16). All detected boxes with an overlap greater than the NMS threshold are merged to the box with the highest confidence score. NMS accepts IoU values between 0 (no overlap) to 1 (complete overlap).

Fruit detection models should therefore be tuned for both confidence threshold and NMS values to achieve the highest F1-score on the validation set as illustrated in Koirala et al. (2019).

### 7.3. Training and test time

Few papers report on training and detection time. Even when such information is reported, comparison is difficult given use of different CPUs and image resolution. Model training time is generally not critical as it represents a one off exercise. Training time is a function of batch size (number of images processed in parallel), which depends on available memory and, thus, on the computing resource employed. Institutional researchers often access High Performance Computing (HPC) clusters. Deep learning models can also be trained on cloud based high performance computers (e.g., Amazon's AWS, Microsoft's Azure and Google's GCP), alleviating the need for a local computing resource.


**Fig. 14.** An example display of class label 'm' (mango fruit) and associated confidence scores for an image of mango fruit on tree, produced by Faster-RCNN using ZFNet (left panel) and VGGNet (right panel).

In contrast, test or detection time is of importance to the use of a detection system if real or near real time detection is required, e.g., at a processing time of 1 s per image, an orchard yield estimation for a large orchard of 100,000 trees, using two images per tree, requires 55 h. Future fruit harvesting operations will require near real time detection. In general, a processing rate of >30 frames (images) per second (<30 ms per image) is regarded as a real-time speed for video processing, but slower rates may be acceptable for particular applications.

Detection speed will be a function of algorithm complexity and computing capacity. For example, good accuracy on sweet pepper detection was noted for a Conditional Random Field (CRF) framework relative to a deep learning framework (Faster R-CNN), but at the cost of processing time (842 times greater) and a more tedious ground truthing for the pixel-level approach, compared to bounding box annotation (Sa et al., 2016). System design will therefore depend on the importance of accuracy relative to speed.

Sa et al. (2016) employed a GeForce GTX980M 8 GB GPU, reporting processing of 1297 × 964 images within ~341 ms, and 1920 × 1080 images within ~393 ms for a multimodal Faster R-CNN model used in detection of sweet peeper fruit. Rahnemoonfar and Sheppard (2017) reported processing time using a NVidia 980Ti 6 GB GPU for fruit count on tomato plant (128 × 128) images at 6 and 50 ms, using deep learning CNN and area-based methods, respectively. On a NVDIA 980 Ti GPU, Bargoti and Underwood (2017a) reported average detection times of 130 ms per (500 × 500) image using a Faster R-CNN framework VGG-16 model, 40 ms per image using a ZFNet model and 2300 ms per image using a pixel-wise CNN model. Koirala et al. (2019) reported on performance, memory requirement and inference time for several deep learning models (VGG, ZFNet and DarkNet) and object detection frameworks (Faster R-CNN, SSD and YOLO) in context of a mango fruit image dataset. For an image size of 512 × 512 pixels and using a HPC platform (NVIDIA Tesla P100 GPU), SSD and YOLO variants YOLOv3, YOLOv2, YOLOv2(tiny) and MangoYOLO took 70, 25, 20, 10 and 15 ms per image, respectively, for fruit detection, while Faster R-CNN(ZF) and Faster R-CNN(VGG) took 37 and 67 ms, respectively. For 2048 × 2048 input images, the detection times for MangoYOLO and YOLOv2 (tiny) were 20 and 10 ms and 70 and 51 ms on the HPC and on a NVIDIA GeForce GTX 1070 Ti GPU (local computer) platforms, respectively.

## 8. Fruit detection using deep learning

Prior to 2016, published work to discriminate fruits from background in images of tree fruit canopies was based on hand-engineered features to encode visual attributes (Gongal et al., 2015). Subsequently there have been 12 papers published on use of deep learning models in estimation of tree fruit number per image (Scopus data base, www.scopus.com, keywords:'deep' + 'learning' + 'fruit' + 'detection', accessed 19/01/2019, revealing 9 publications, with an additional four publications located 'by hand') (Table 1).

The first report on use of deep learning with CNN for fruit detection was by Sa et al. (2016). These authors implemented a multimodal faster R-CNN (using both RGB + NIR), reporting an improved F1 score (increased from 0.81 to 0.84) for sweet pepper detection compared to their prior work (McCool et al., 2016), which was based on a pixel-wise segmentation technique. A F1 score > 0.8 was obtained for all 6 fruit

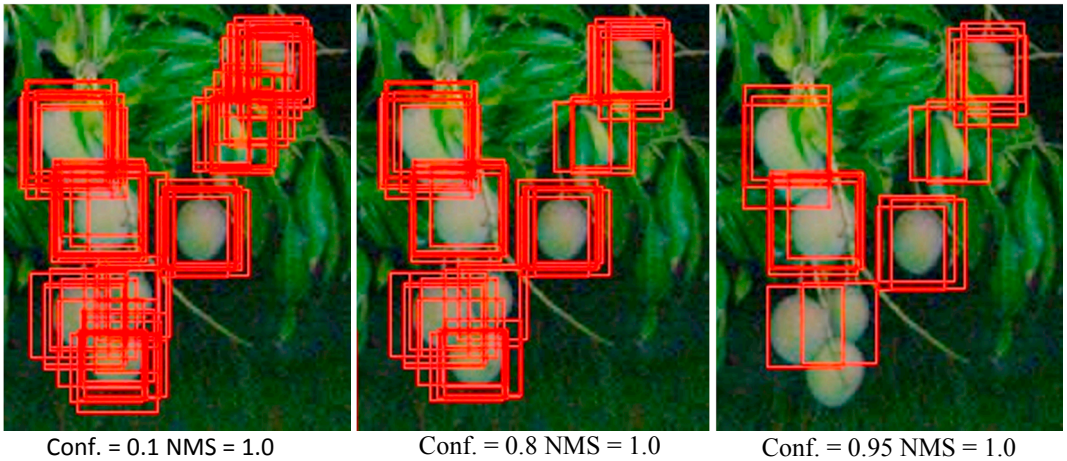


**Fig. 15.** Object detection with no suppression (NMS = 1.0) and an increasing level of confidence threshold values (0.1, 0.8, 0.95, for left to right panels) resulting in fewer multiple detections per fruit (lower FP), but failure to detect some fruit (higher FN).

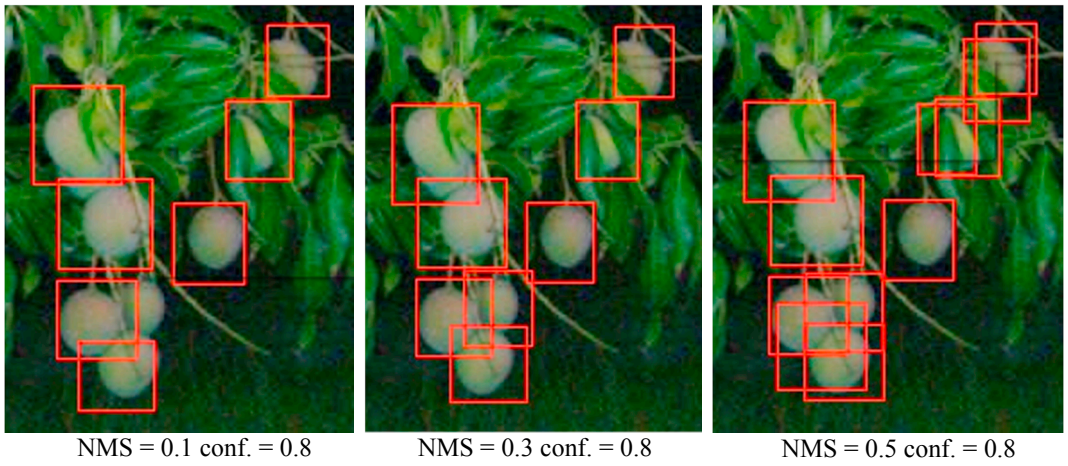


**Fig. 16.** Effect of NMS setting: left to right panels: NMS = 0.1, one detection for each fruit but some FN; NMS 0.3, one detection for each fruit with no FN; NMS 0.5, but not all detections merged.

categories (apple, orange, avocado, mango, strawberry) considered.

Subsequent studies have implemented the latest architectures from the deep learning – CNN community, with some studies attempting to modify architecture for the fruit classification/detection application. F1 scores of >0.9 are being routinely achieved. A summary of representative studies follows:

Chen et al. (2017) proposed a deep learning method to directly estimate total fruit count from input images, with integration of fruit segmentation and a count regression network into a single pipeline. A Fully Convolutional Network (FCN) (Long et al., 2015) was used to extract candidate regions in the images as blobs, while another CNN was used to count the number of fruits (oranges and apples) inside each blob of fruit region. A regression model related CNN derived fruit count to human fruit-in-image count.

Other studies have used deep learning with CNN to detect fruit, then counted the detected fruit. For example, Bargoti and Underwood (2017a) used the original Faster R-CNN framework, achieving higher F1 scores using VGG-16 than ZFNet (Table 1). The results were superior to work using pixel-wise CNN (a multiscale Multi Layered Perceptron with three hidden layers and a CNN of two convolution, pooling and fully connected layers respectively) (Bargoti and Underwood, 2017b). The F1 score for fruit detection was thus improved with use of deeper networks (pixel-wise CNN vs ZFNet vs VGG-16) (Table 1). Similarly, (Kestur et al., 2019), used a full convolutional deep CNN– 'MangoNet' to segment mango fruit in the images followed by connected object detection for fruit counting in images.

Rahnemoonfar and Sheppard (2017) took the innovative step of deep simulated learning within the task of tomato fruit counting. Inception-ResNet architecture was modified, with the number of fruit objects estimated without detection and localization of objects, acting on a view of the entire image. The model was modified and trained entirely on synthetic (tomato) fruit images and tested on natural images, with a 91% average test accuracy, $R^2$ of 0.90 and RMSE of 2.52 fruit/image on real images (downloaded from Google images; only 100 images used) and 93% on synthetic images. The algorithm was robust to varying degree of lighting conditions, occlusions and fruit overlaps. The modified Inception-ResNet architecture achieved a higher fruit detection accuracy than the standard Inception-ResNet architecture, while a shallow CNN of four layers (2 convolution and 2 fully connected layers) performed poorly (91.0, 70.0 and 11.6% accuracy, respectively).

Mureşan and Oltean (2018) used deep learning (a CNN with four convolutional layers) for fruit classification (recognition) of 60 fruit categories, reporting a classification accuracy of 96.3% on a test set. In this application fruit localisation is not required. The datasets ('Fruit-360') were released by the authors, allowing comparative work to be undertaken by other researchers.

Koirala et al. (2019) compared the detection results for a number of

**Table 1**
Scientific reports on use of deep learning models in estimation of tree fruit number per image. The best result of each paper is shown. When available, the F1 score is recorded, otherwise the validation metric used by the authors is included.

| Author | Commodity | Method | Validation statistics |
|---|---|---|---|
| Sa et al (2016) | Sweet pepper | Faster R-CNN | F1 0.84 |
| Bargoti and Underwood (2017a) | Apple, mango, almond | Faster R-CNN ZF | F1 0.89, 0.88, 0.73 |
| | Apple, mango, almond | Faster R-CNN VGG | F1 0.90, 0.91, 0.76 |
| Bargoti and Underwood (2017b) | Apple, mango | Pixel wise CNN | F1 0.86, 0.84 |
| Chen et al. (2017) | Apple, orange | FCN + CNN regression | Ratio counted 0.91, 0.97 |
| Choi et al. (2015) | Citrus (classification) | AlexNet | TP rate 96% |
| Habaragamuwa et al. (2018) | Strawberry | CNN | AP 88.0% |
| | | | Bounding Box Overlap 0.74 |
| Liang et al. (2018) | Mango | SSD VGG | F1 0.91 |
| Peng et al. (2018) | Apple, litchi, navel orange, Huangdi gan | SSD ResNet | F1 0.96 |
| Tao et al. (2018) | Peach, apple, orange | Faster R-CNN | AP 91.5, 94.2, 90.2% |
| Xiong et al. (2018) | Green citrus | Faster R-CNN | F value 77.5% |
| | | | mAP 85.5 |
| Koirala et al. (2019) | Mango | Faster-RCNN ZF | F1 0.94 |
| | | Faster-RCNN VGG | F1 0.95 |
| | | SSD VGG | F1 0.96 |
| | | MangoYOLO | F1 0.97 |
| Kestur et al. (2019) | Mango | CNN | F1 0.84 |

detection frameworks (Faster-RCNN, SSD and YOLO) for mango fruit detection, and re-designed the YOLO architecture (to 'MangoYOLO') for speed and memory optimization for the mango fruit application. All models achieved F1 > 0.90 on an independent validation set of 512 × 512 pixels images with the highest score (F1 of 0.97) from the MangoYOLO model. The superior result for MangoYOLO compared to the deeper architecture of YOLOv3 demonstrates that it is not number of layers alone that defines the performance, but also the design of the architecture. In MangoYOLO, Koirala et al. (2019) merged the information from early detection layers to that of the later detection layers.

## 9. Tree fruit yield estimation

### 9.1. Can't see the forest for the trees?

Uses for machine vision detection of fruit in images of tree canopies include estimation of fruit number per tree ('load'), in-field fruit sizing, and automated harvest. Estimation of fruit size together with fruit number allows estimation of fruit weight ('yield') per orchard.

Fruit weight can be correlated to fruit lineal dimensions in many fruit. Such a relationship allows fruit weight to be estimated using machine vision, given a measure of camera to fruit distance, e.g. using a time of flight camera. For whole canopy images, as used in fruit load estimation, only a fraction of fruit in the image have entire outlines (i.e., not partially occluded), but that is sufficient to provide a size class distribution, assuming visible, outer canopy fruit are representative of all fruit on a tree (as achieved by Wang et al., 2017).

Of course, a tree fruit load estimate relies on assessment of the total number of fruit per tree, not the number of fruit visible in an image. With high accuracies reported for fruit detection in an image using deep learning methods, as discussed earlier in Section 8, research attention should now shift to approaches to estimate total fruit per tree and per orchard.

### 9.2. Seeing all fruit

One approach to the estimation of fruit number per tree and orchard involves increasing the number of viewpoints of images of the tree, in an attempt to visualise all fruit on the tree (akin to a human estimate of fruit on tree, which involves walking around the tree). Payne et al. (2013) reported a higher ($R^2$) between image counts and human counts of total fruit on tree based on a summation of counts from images taken on four sides of each canopy, compared to counts from images of two or one side.

Multiple imaging of one canopy can result in over-estimation of fruit load by multiple counting of the same fruit. This issue has been addressed by background removal or masking of fruits based on depth or localizing fruit in 3D space (Wang et al., 2013), or use of stereo imaging with fruit registration between frames from multiple viewpoints (Moonrinta et al., 2010). Wang et al. (2013) and Song et al. (2014) implemented an object (fruit) tracking algorithm with multiple viewpoint images of apple and pepper plants, respectively.

Stein et al. (2016) described a 'multiview' method in which 37 images of each side of mango tree canopies were captured from a passing platform. Fruit detection was done using deep learning (Faster R-CNN), inferring the instances of detected bounding-box as fruit counts as in (Bargoti and Underwood, 2017a). Fruit number per tree was estimated based on an epipolar projection approach with fruit tracking using trajectory data (camera pose) provided by the navigation system and association of fruit to individual trees achieved using a LiDAR mask of orchard trees. A $R^2 = 0.90$ for total fruit numbers for 16 trees (harvest count) with an error rate of only 1.36% for individual tree. A slope ~1 was reported on actual fruit number per tree (for 18 'calibration' trees, as determined by harvest). In this case, any double counting of fruit was balanced by non-detection (hidden fruit).

Similarly, Liu et al. (2018) used a FCN to segment fruit pixels followed by fruit tracking across sequence of video frames. Fruit were then localized in 3D (to minimize errors of double counting fruit on same tree and form other rows) using a Structure from Motion (SfM) algorithm. For fruit counts against human count on images, L1 error of 203 and 322 and error mean of −0.2% and 3.3% and error standard deviation of 7.8% and 4.1% were reported for orange (daytime) and apple (night-time) images respectively

However, these multiple view approaches are computationally complex and require precise object tracking algorithms. Tracking can be challenging in scenes with large number of similar objects (fruit), as present in a canopy scene. This can be an issue particularly for fruit clusters. Also, double counting of fruit will occur if the tracking of a fruit is lost due to fruit occlusion or sudden movements in the camera platform. Extra hardware (e.g., an inertial navigation system) is used to minimise such issues.

### 9.3. Occluded fruit correction factor

Another approach to provide a tree fruit load estimate from a count of fruit in a tree image involves calculation of a correction factor for occluded fruit. The simplest approach is to use a single factor per orchard. For example, Payne et al. (2013), Wang et al. (2013), Stein et al. (2016), Chen et al. (2017) and Koirala et al. (2019) report use of the slope of the linear regression between the machine vision image count and harvest count for a set of calibration trees for fruit load estimation. This method relies on consistent tree architecture across the orchard (i.e., a single correction factor is applied to all trees). The factor must be established empirically for each orchard and may have to be repeated each season if canopy structure/fruit position in canopy changes. For example, for the five mango orchards considered, Koirala et al. (2019) report that an orchard wide occluded fruit correction factor used in conjunction with MangoYOLO based estimates of fruit number per canopy image achieved fruit yield estimates between 4.6 and 15.2% of packhouse fruit count. The correction factor varied from 1.05 to 2.43 across the five orchards.

Ideally, a correction factor could be established on an individual tree basis rather than for the whole orchard block. For example, the number of totally occluded fruit may be proportional to the number of partly occluded fruit.

A per tree correction factor can be built into the model used to estimate fruit count per image. For example, Črtomir et al. (2012) trained an ANN using several input parameters (fruit area, canopy area and number of visible fruits on images) against actual harvest count for apple trees. The ANN model provided an improved yield estimation, at $R^2 = 0.69$ (Golden Delicious) and $R^2 = 0.61$ (Braeburn), compared to a simple regression model (fruit count on images as the only input), at $R^2 = 0.53$ (Golden delicious) and $R^2 = 0.26$ (Braeburn). Cheng et al. (2017) used fruit and canopy features (number and area of fruit, area of fruit clusters, and leaf area per tree image) to train a back-propagation neural network (BPNN) model for prediction of apple yield (fruit weight per tree), reporting a $R^2$ of 0.75, RMSE of 2.5 kg/tree, for a crop near harvest maturity.

### 9.4. Estimating fruit weight

As noted earlier, allometric relationships exist between fruit lineal dimensions and weight. Stajnko et al. (2009) employed traditional machine vision segmentation techniques in estimation of fruit number per canopy image using two images per tree and fruit size, using a size marker placed in the tree. Apple yield per tree ($Y_t$) was calculated based on Mitchell's (Mitchell, 1986) equation, involving number and diameter of fruits.

$$Y_t = N \times a \times D^b,$$

where: Yt = yield per tree, N = number of fruits per tree, D = average diameter of fruit, a,b = constants depending on apple variety. For average yield per tree (kg) an $R^2$ of 0.96 between manual measurement and machine vision estimation was reported.

Črtomir et al. (2012) used an ANN (4-6-1 and 5-14-1 network architectures for varieties Golden Delicious and Braeburn respectively) to estimate apple fruit yield (kg per tree) from a single image per canopy. The ANN regressed the inputs of fruit segmentation, as described in Stajnko and Čmelik (2005) against weighted yield per tree. An $R^2$ of 0.69 and 0.61 and standard deviation of 2.83 and 2.55 kg/tree between forecasted and actual (harvest) yield per tree was achieved for Golden and Braeburn varieties respectively. This result was superior to that obtained using the method of Stajnko et al. (2009).

Cheng et al. (2017) used fruit and canopy features from images (fruit number estimated using a traditional colour segmentation method as described by Zhou et al. (2012), fruit area, fruit cluster area, and foliage leaf area) as input parameters to an ANN (Backpropagation Neural Network, BPNN) for yield (fruit weight per tree) prediction. Two separate ANN architectures were applied: a 4-12-1 architecture for the early period of fruiting and a 4-11-1 architecture for the ripening period. Both models performed well for apple prediction with the later period model slightly better than the early period model. The later period model achieved an $R^2$ of 0.82 and RMSE of 2.31 kg/tree on a Gala apple test set from the season of model development, and an $R^2$ of 0.75 and RMSE of 2.54 kg/tree for a next season test set. A BPNN model (4-10-1 architecture) on Pinova variety achieved an $R^2$ of 0.88 and RMSE of 2.53 kg/tree on the test set.

Other workers have reported relationships between yield (fruit number or fruit weight per tree and canopy characters based on remote (satellite) imagery such as spectral indices and canopy area. For example, Rahman et al. (2018) report use of an ANN regression model (10-5-1 architecture) between spectral indices and crown area against mango fruit yield. However, while canopy area and plant health will be linked to the potential to flower and for fruit set (i.e., the number of vegetative terminals on a mango canopy sets the upper limit for the number of panicles that can form), there can be high levels of flower and fruit drop. Therefore such models will require annual 'calibration' to actual yield, to accommodation changes in flowering and fruit set conditions.

Deep learning has yet to be applied to such data sets for estimation of fruit load per tree and per orchard. There are examples with other crops where deep learning models have been applied with more complex spatio-temporal data and remotely sensed multi-spectral images for yield (t/ha) prediction. Kuwata and Shibasaki (2015) used a Caffe based deep learning regression model (Gaussian Radial Basis Function) trained with remotely sensed data (satellite data, climate data and environmental metadata) to model corn crop yield estimation at a county level. Jiang et al. (2018) used weather and soil data as inputs to a Long Short-Term Memory (LSTM) model for corn yield at a county level. Similarly, You et al. (2017) used deep learning (CNN and LSTM) models trained on yearly yield and multi-spectral remote sensing data for real-time forecasting of soybean yield at a county level. The advantage of deep learning is the automatic extraction of meaningful information from the real world raw data. There is a need to explore different deep learning architectures for their possible implementation in the precision agriculture field.

## 10. Conclusion and recommendations

The need for handcrafting of features and designing of feature descriptors is removed in an ANN through automated learning. In general, ANN model accuracy improves with number of model layers, but at the cost of increased computational complexity. Object detection frameworks have evolved under the selective pressure of a requirement for higher speed, moving from two-staged region based detectors to single shot dense object detectors. Deep learning models are reported to outperform pixel-wise segmentation techniques involving traditional machine learning and shallower CNN and Neural networks in the task of fruit-on-plant detection. The availability of publicly-available detection frameworks with pre-trained weights and large public datasets of annotated images for training have made adopting deep learning relatively easy. Typically, with use of some hundreds of training images, the models can be fine-tuned for use in a particular image object detection task. Public datasets of annotated images of fruit on tree (and other agricultural applications) should be created to facilitate algorithm comparison using the same image training, validation and test sets.

There will be many reports generated on the adoption of a deep learning framework for specific precision agricultural applications. Published research studies must present information on number of training samples and values of network parameters used for model optimization. For replication of a given study and for comparison/benchmarking, the source of the open-source codes/algorithms and the model parameters (IoU and NMS thresholds) must be reported. A recommendation is made for use of the F1 score as a common evaluation metric for performance measure of the prediction models. Studies should also include several test sets independent to training set for the evaluation of model robustness.

However, while reports of performance of a given model have technical merit to achieve an application end, the research frontier in this field has now moved to creation of existing deep learning architecture to suit a particular application, in terms of either increased speed or accuracy. Recent application of deep learning methods to directly predict fruit numbers on images without the need for labour intensive data labelling (pixel-wise or bounding box method) is of practical importance. Given the accuracy of current frameworks in fruit detection, research focus should also shift towards accurate estimation of orchard fruit number, i.e., to a consideration of systems to avoid or adjust for occluded fruit. Reports of fruit load estimation should be validated using test sets from more than one season, one cultivar and one orchard (i.e., one canopy architecture). It is recommended that fruit load estimation be attempted using several deep learning approaches (e.g., CNN detectors, deep regression and LSTM) involving several data sources (e.g., tree images, orchard metadata, weather and yield history).

Precise fruit detection allows generation of yield maps, providing information on spatial variation on which to base agronomic decisions. Fruit load estimation is also useful to inform harvest resourcing and management, and marketing. Light weight deep learning models will also support robotic harvesting.

Deep learning has been discussed in context of machine vision in this review, but the technique has relevance with other data types. The ability of deep learning methods to automatically extract useful information from multi-dimensional raw data (e.g., digital images, LIDAR data, multi spectral, remote sensing and satellite data, weather, climate and environmental data from several sensors) and its scalability with big data holds great promise for applications such as real-time yield estimation and forecasting, crop growth modelling and plant disease and pest modelling. Application development will depend on quality data encompassing a range of environmental conditions, cultivars and crop types and modalities (sensors and data types).

## Acknowledgement

This work received funding support from the Australian Federal Department of Agriculture and Water (R&D4Profit program), administered through Hort Innovation (project ST15005, Multiscale monitoring of tropical fruit production). Anand Koirala and Zhenglin Wang acknowledge receipt of an Australian Regional Universities Network scholarship and a CQUniversity Early Career Fellowship, respectively.

## References


Annamalai, P., Lee, W.S., 2004. Burks TF Color vision system for estimating citrus yield in real-time. In: ASAE Annual Meeting. American Society of Agricultural and Biological Engineers, pp. 1. https://doi.org/10.13031/2013.16714.

Bargoti, S., Underwood, J., 2017a. Deep fruit detection in orchards. In: Proceedings – IEEE International Conference on Robotics and Automation, pp. 3626–3633. https://doi.org/10.1109/ICRA.2017.7989417.

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

Bay, H., Tuytelaars, T., Van Gool, L., 2006. Surf: speeded up robust features. In: European Conference on Computer Vision. Springer, pp. 404–417. https://doi.org/10.1007/11744023_32.

Chen, S.W., Shivakumar, S.S., Dcunha, S., Das, J., Okon, E., Qu, C., Taylor, C.J., Kumar, V., 2017. Counting apples and oranges with deep learning: a data-driven approach. IEEE Rob. Autom. Lett. 2, 781–788. https://doi.org/10.1109/LRA.2017.2651944.

Cheng, H., Damerow, L., Sun, Y., Blanke, M., 2017. Early yield prediction using image analysis of apple fruit and tree canopy features with neural networks. J. Imaging 3, 6.

Choi, D., Lee, W.S., Ehsani, R., Schueller, J.K., Roka, F., 2015. Machine vision system for early yield estimation of citrus in a site-specific manner. In: ASABE Annual International Meeting. American Society of Agricultural and Biological Engineers, pp. 1. https://doi.org/10.13031/aim.20152181863.

Chollet, F., 2017. Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1251–1258.

Cortes, C., Vapnik, V., 1995. Support-vector networks. Mach. Learn. 20, 273–297. https://doi.org/10.1007/BF00994018.

Črtomir, R., Urška, C., Stanislav, T., Denis, S., Karmen, P., Pavlovič, M., Marjan, V., 2012. Application of neural networks and image visualization for early forecast of apple yield. Erwerbs-Obstbau 54, 69–76. https://doi.org/10.1007/s10341-012-0162-y.

Dalal, N., Triggs, B., 2005. Histograms of oriented gradients for human detection. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2005. pp. 886–893. https://doi.org/10.1109/CVPR.2005.177.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L., 2009. Imagenet: A large-scale hierarchical image database. In: Proceedings – IEE conference on Computer Vision and Pattern Recognition, pp. 248–255. https://doi.org/10.1109/CVPR.2009.5206848.

Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A., 2010. The pascal visual object classes (voc) challenge. Int. J. Comput. Vision 88, 303–338. https://doi.org/10.1007/s11263-009-0275-4.

Girshick, R., 2015. Fast R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1440-1448. https://doi.org/10.1109/ICCV.2015.169.

Girshick, R., Donahue, J., Darrell, T., Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 580–587. https://doi.org/10.1109/CVPR.2014.81.

Gongal, A., Amatya, S., Karkee, M., Zhang, Q., Lewis, K., 2015. Sensors and systems for fruit detection and localization: a review. Comput. Electron. Agric. 116, 8–19. https://doi.org/10.1016/j.compag.2015.05.021.

Habaragamuwa, H., Ogawa, Y., Suzuki, T., Shiigi, T., Ono, M., Kondo, N., 2018. Detecting greenhouse strawberries (mature and immature), using deep convolutional neural network. Eng. Agric. Environ. Food 11, 127–138. https://doi.org/10.1016/j.eaef.2018.03.001.

He, K., Gkioxari, G., Dollar, P., Girshick, R., 2017. Mask R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2980–2988. https://doi.org/10.1109/ICCV.2017.322.

He, K., Zhang, X., Ren, S., Sun, J., 2014. Spatial pyramid pooling in deep convolutional networks for visual recognition. In: European Conference on Computer Vision. Springer, pp. 346–361. https://doi.org/10.1007/978-3-319-10578-9_23.

He, K., Zhang, X., Ren, S., Sun, J., 2015. Delving deep into rectifiers: surpassing human-level performance on ImageNet classification. In: Proceedings – IEEE International Conference on Computer Vision, pp. 1026–1034.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 770–778.

Hu, J., Shen, L., Sun, G., 2017. Squeeze-and-excitation networks. arXiv preprint 7.

Hung, C., Underwood, J., Nieto, J., Sukkarieh, S., 2015. A feature learning based approach for automated fruit yield estimation. In: Field and Service Robotics. Springer, pp. 485–498. https://doi.org/10.1007/978-3-319-07488-7_33.

Jiang, Z., Liu, C., Hendricks, N.P., Ganapathysubramanian, B., Hayes, D.J., Sarkar, S., 2018. Predicting County Level Corn Yields Using Deep Long Short Term Memory Models. arXiv preprint .

Kadir, M.F.A., Yusri, N.A.N., Rizon, M., Bin Mamat, A.R., Makhtar, M., Jamal, A.A., 2015. Automatic mango detection using texture analysis and randomised hough transform. Appl. Math. Sci. 9, 6427–6436. https://doi.org/10.12988/ams.2015.53290.

Kamilaris, A., Prenafeta-Boldú, F.X., 2018. Deep learning in agriculture: a survey. Comput. Electron. Agric. 147, 70–90. https://doi.org/10.1016/j.compag.2018.02.016.

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

Koirala, A., Wang, Z., Walsh, K., McCarthy, C., 2019. Deep learning for real-time fruit detection and orchard fruit load estimation: benchmarking of *'MangoYOLO'*. Precis. Agric. https://doi.org/10.1007/s11119-019-09642-0.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105.

Kurtulmus, F., Lee, W.S., Vardar, A., 2011. Green citrus detection using 'eigenfruit', color and circular Gabor texture features under natural outdoor conditions. Comput. Electron. Agric. 78, 140–149. https://doi.org/10.1016/j.compag.2011.07.001.

Kuwata, K., Shibasaki, R., 2015. Estimating crop yields with deep learning and remotely sensed data. In: IEEE International Geoscience and Remote Sensing Symposium (IGARSS), pp. 858–861. https://doi.org/10.1109/IGARSS.2015.7325900.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proc. IEEE 86, 2278–2323. https://doi.org/10.1109/5.726791.

Liakos, K., Busato, P., Moshou, D., Pearson, S., Bochtis, D., 2018. Machine learning in agriculture: a review. Sensors 18, 2674. https://doi.org/10.3390/s18082674.

Liang, Q., Zhu, W., Long, J., Wang, Y., Sun, W., Wu, W., 2018. A real-time detection framework for on-tree mango based on SSD network. In: International Conference on Intelligent Robotics and Applications. Springer, pp. 423–436. https://doi.org/10.1007/978-3-319-97589-4_36.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L., 2014. Microsoft coco: common objects in context. In: European Conference on Computer Vision. Springer, pp. 740–755.

Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollar, P., 2017. Focal loss for dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2999–3007. https://doi.org/10.1109/ICCV.2017.324.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C., 2016. Ssd: Single shot multibox detector. In: European Conference on Computer Vision. Springer, pp. 21–37. https://doi.org/10.1007/978-3-319-46448-0_2.

Liu, X., Chen, S.W., Aditya, S., Sivakumar, N., Dcunha, S., Qu, C., Taylor, C.J., Das, J., Kumar, V., 2018. Robust Fruit Counting: Combining Deep Learning, Tracking, and Structure from Motion. arXiv preprint .

Long, J., Shelhamer, E., Darrell, T., 2015. Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 3431–3440.

Lowe, D.G., 1999. Object recognition from local scale-invariant features. In: Proceedings of the Seventh IEEE International Conference on Computer Vision, pp. 1150–1157. https://doi.org/10.1109/ICCV.1999.790410.

McCool, C., Sa, I., Dayoub, F., Lehnert, C., Perez, T., 2016. Upcroft B Visual detection of occluded crop: for automated harvesting. In: IEEE International Conference on Robotics and Automation, pp. 2506–2512. https://doi.org/10.1109/ICRA.2016.7487405.

Mitchell, P., 1986. Pear fruit growth and the use of diameter to estimate fruit volume and weight. HortScience 21, 1003–1005.

Moonrinta, J., Chaivivatrakul, S., Dailey, M.N., Ekpanyapong, M., 2010. Fruit detection, tracking, and 3D reconstruction for crop mapping and yield estimation. In: 11th International Conference on Control Automation Robotics & Vision, pp. 1181–1186. https://doi.org/10.1109/ICARCV.2010.5707436.

Mureşan, H., Oltean, M., 2018. Fruit recognition from images using deep learning Acta Universitatis Sapientiae. Informatica 10, 26–42.

Naik, S., Patel, B., 2017. Machine vision based fruit classification and grading – a review. Int. J. Comput. Appl. 170, 22–34 (0975-8887).

Nanaa, K., Rizon, M., Rahman, M.N.A., Ibrahim, Y., 2014. Aziz AZA Detecting mango fruits by using randomized hough transform and backpropagation neural network. In: Proceedings of the International Conference on Information Visualisation, pp. 388–391. https://doi.org/10.1109/IV.2014.54.

Norris, K., 1996. History of NIR. J. Near Infrared Spectrosc. 4, 31–37. https://doi.org/10.1255/jnirs.941.

Ojala, T., Pietikäinen, M., Harwood, D., 1996. A comparative study of texture measures with classification based on featured distributions. Pattern Recogn. 29, 51–59. https://doi.org/10.1016/0031-3203(95)00067-4.

Payne, A.B., Walsh, K.B., Subedi, P., Jarvis, D., 2013. Estimation of mango crop yield using image analysis–segmentation method. Comput. Electron. Agric. 91, 57–64. https://doi.org/10.1016/j.compag.2012.11.009.

Peng, H., Huang, B., Shao, Y., Li, Z., Zhang, C., Chen, Y., Xiong, J., 2018. General improved SSD model for picking object recognition of multiple fruits in natural environment. Trans. Chin. Soc. Agric. Eng. 34, 155–162. https://doi.org/10.11975/j.issn.1002-6819.2018.16.020.

Pothen, Z.S., Nuske, S., 2016. Texture-based fruit detection via images using the smooth patterns on the fruit. In: IEEE International Conference on Robotics and Automation, pp. 5171–5176. https://doi.org/10.1109/ICRA.2016.7487722.

Rahman, M., Robson, A., Bristow, M., 2018. Exploring the potential of high resolution worldview-3 imagery for estimating yield of mango. Remote Sens. 10, 1866. https://doi.org/10.3390/rs10121866.

Rahnemoonfar, M., Sheppard, C., 2017. Deep count: fruit counting based on deep simulated learning. Sensors 17. https://doi.org/10.3390/s17040905.

Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You only look once: unified, real-time object detection. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 779–788. https://doi.org/10.1109/CVPR.2016.91.

Redmon, J., Farhadi, A., 2017. YOLO9000: Better, Faster, Stronger. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7263–7271. https://doi.org/10.1109/CVPR.2017.690.

Redmon, J., Farhadi, A., 2018. YOLOv3: An Incremental Improvement. arXiv preprint .

Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, pp. 91–99.

Sa, I., Ge, Z., Dayoub, F., Upcroft, B., Perez, T., McCool, C., 2016. Deepfruits: a fruit

detection system using deep neural networks. Sensors 16. https://doi.org/10.3390/s16081222.
Sa, I., McCool, C., Lehnert, C., 2015. Perez T On visual detection of highly-occluded objects for harvesting automation in horticulture. IEEE International Conference on Robotics and Automation.
Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D., 2017. Grad-CAM: visual explanations from deep networks via gradient-based localization. In: IEEE International Conference on Computer Vision (ICCV), 22–29 Oct. 2017, pp. 618–626. https://doi.org/10.1109/ICCV.2017.74.
Sengupta, S., Lee, W.S., 2014. Identification and determination of the number of immature green citrus fruit in a canopy under different ambient light conditions. Biosyst. Eng. 117, 51–61. https://doi.org/10.1016/j.biosystemseng.2013.07.007.
Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y., 2013. Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint .
Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint .
Song, Y., Glasbey, C., Horgan, G., Polder, G., Dieleman, J., Van der Heijden, G., 2014. Automatic fruit recognition and counting from multiple images. Biosyst. Eng. 118, 203–215. https://doi.org/10.1016/j.biosystemseng.2013.12.008.
Stajnko, D., Čmelik, Z., 2005. Modelling of apple fruit growth by application of image analysis. Agric. Conspectus Sci. 70, 59–64.
Stajnko, D., Rakun, J., Blanke, M., 2009. Modelling apple fruit yield using image analysis for fruit colour, shape and texture. Eur. J. Horticult. Sci. 74, 260–267.
Stein, M., Bargoti, S., Underwood, J., 2016. Image based mango fruit detection, localisation and yield estimation using multiple view geometry. Sensors 16. https://doi.org/10.3390/s16111915.
Syal, A., Garg, D., Sharma, S., 2013. A survey of computer vision methods for counting fruits and yield prediction. Int. J. Comput. Sci. Eng. 2, 346–350.
Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9. https://doi.org/10.1109/CVPR.2015.7298594.
Szegedy, C., Reed, S., Erhan, D., Anguelov, D., Ioffe, S., 2014. Scalable, high-quality object detection. arXiv preprint .
Tao, Y., Zhou, J., Wang, K., Shen, W., 2018. Rapid detection of fruits in orchard scene based on deep neural network. In: ASABE 2018 Annual International Meeting. https://doi.org/10.13031/aim.201801055.
Turk, M., Pentland, A., 1991. Eigenfaces for recognition. J. Cognit. Neurosci. 3, 71–86. https://doi.org/10.1162/jocn.1991.3.1.71.
Uijlings, J.R., Van De Sande, K.E., Gevers, T., Smeulders, A.W., 2013. Selective search for object recognition. Int. J. Comput. Vision 104, 154–171. https://doi.org/10.1007/s11263-013-0620-5.
Viola, P., Jones, M., 2001. Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. . https://doi.org/10.1109/CVPR.2001.990517.
Wachs, J., Stern, H., Burks, T., Alchanatis, V., 2009. Bet-Dagan I Apple detection in natural tree canopies from multimodal images. In: Proceedings of the 7th European Conference on Precision Agriculture, pp. 293–302.
Wang, Q., Nuske, S., Bergerman, M., Singh, S., 2013. Automated crop yield estimation for apple orchards. In: Experimental Robotics, vol. 88. Springer, pp. 745–758. https://doi.org/10.1007/978-3-319-00065-7_50.
Wang, Z., Walsh, K.B., Verma, B., 2017. On-tree mango fruit size estimation using RGB-D images. Sensors 17. https://doi.org/10.3390/s17122738.
Xiong, J., Liu, Z., Tang, L., Lin, R., Bu, R., Peng, H., 2018. Visual detection technology of green citrus under natural environment. Nongye Jixie Xuebao/Trans. Chin. Soc. Agric. Mach. 49, 45–52. https://doi.org/10.6041/j.issn.1000-1298.2018.04.005.
You, J., Li, X., Low, M., Lobell, D., Ermon, S., 2017,. Deep gaussian process for crop yield prediction based on remote sensing data. In: 31st AAAI Conference on Artificial Intelligence, AAAI 2017, pp. 4559–4565.
Zeiler, M.D., Fergus, R., 2014. Visualizing and Understanding Convolutional Networks, vol. 8689 LNCS. https://doi.org/10.1007/978-3-319-10590-1_53.
Zeng, X., Ouyang, W., Yan, J., Li, H., Xiao, T., Wang, K., Liu, Y., Zhou, Y., Yang, B., Wang, Z., 2018. Crafting gbd-net for object detection. IEEE Trans. Pattern Anal. Mach. Intell. 40, 2109–2123. https://doi.org/10.1109/TPAMI.2017.2745563.
Zhao, Y., Gong, L., Zhou, B., Huang, Y., Liu, C., 2016. Detecting tomatoes in greenhouse scenes by combining AdaBoost classifier and colour analysis. Biosyst. Eng. 148, 127–137. https://doi.org/10.1016/j.biosystemseng.2016.05.001.
Zhou, R., Damerow, L., Sun, Y., Blanke, M.M., 2012. Using colour features of cv. 'Gala'apple fruits in an orchard in image processing to predict yield. Precis. Agric. 13, 568–580. https://doi.org/10.1007/s11119-012-9269-2.