



# Deep learning for real-time fruit detection and orchard fruit load estimation: benchmarking of 'MangoYOLO'

A. Koirala<sup>1</sup> · K. B. Walsh<sup>1</sup> · Z. Wang<sup>1</sup> · C. McCarthy<sup>2</sup>

© Springer Science+Business Media, LLC, part of Springer Nature 2019

## Abstract

The performance of six existing deep learning architectures were compared for the task of detection of mango fruit in images of tree canopies. Images of trees ( $n = 1515$ ) from across five orchards were acquired at night using a 5 Mega-pixel RGB digital camera and 720 W of LED flood lighting in a rig mounted on a farm utility vehicle operating at 6 km/h. The two stage deep learning architectures of Faster R-CNN(VGG) and Faster R-CNN(ZF), and the single stage techniques YOLOv3, YOLOv2, YOLOv2(tiny) and SSD were trained both with original resolution and  $512 \times 512$  pixel versions of 1300 training tiles, while YOLOv3 was run only with  $512 \times 512$  pixel images, giving a total of eleven models. A new architecture was also developed, based on features of YOLOv3 and YOLOv2(tiny), on the design criteria of accuracy and speed for the current application. This architecture, termed 'MangoYOLO', was trained using: (i) the 1300 tile training set, (ii) the COCO dataset before training on the mango training set, and (iii) a daytime image training set of a previous publication, to create the MangoYOLO models 's', 'pt' and 'bu', respectively. Average Precision plateaued with use of around 400 training tiles. *MangoYOLO(pt)* achieved a F1 score of 0.968 and Average Precision of 0.983 on a test set independent of the training set, outperforming other algorithms, with a detection speed of 8 ms per  $512 \times 512$  pixel image tile while using just 833 Mb GPU memory per image (on a NVIDIA GeForce GTX 1070 Ti GPU) used for in-field application. The *MangoYOLO* model also outperformed other models in processing of full images, requiring just 70 ms per image ( $2048 \times 2048$  pixels) (i.e., capable of processing ~14 fps) with use of 4417 Mb of GPU memory. The model was robust in use with images of other orchards, cultivars and lighting conditions. *MangoYOLO(bu)* achieved a F1 score of 0.89 on a day-time mango image dataset. With use of a correction factor estimated from the ratio of human count of fruit in images of the two sides of sample trees per orchard and a hand harvest count of all fruit on those trees, *MangoYOLO(pt)* achieved orchard fruit load estimates of between 4.6 and 15.2% of pack-house fruit counts for the five orchards considered. The labelled images (1300 training, 130 validation and 300 test) of this study are available for comparative studies.

**Keywords** Deep learning · Fruit detection · Mango · Yield estimation

---

✉ A. Koirala  
[a.koirala@cqu.edu.au](mailto:a.koirala@cqu.edu.au)

Extended author information available on the last page of the article

## Introduction

### The horticultural issue

Knowledge of tree fruit crop load and timing of maturation can guide agronomic treatments, labour resource management and support market planning. In Australia, mango harvest timing is based on heat sums from flowering and fruit dry matter content, assessed using handheld near infrared spectroscopy (Walsh and Wang 2018). For mango crop load estimation, the appropriate orchard yield metrics are fruit size distribution and fruit number, rather than tonnes per hectare, as fruit are typically sold at wholesale level as trays of uniform sized fruit and at retail level per piece of fruit, rather than on weight of fruit.

In current practice, harvest forecasts of crop load for the Australian mango industry are based on previous yield history and manual counting of fruit on trees. Harvest forecasts are required six weeks before actual harvest to be of practical use in guiding harvest resourcing decisions, i.e., as early as possible to allow for management decision making, but after stone (endocarp) hardening and the period of fruit drop. However, the current manual yield estimation technique is time consuming, labour intensive and inaccurate (Anderson et al. 2018). Indeed, given the level of variability in fruit load per tree even in a well-managed orchard, Anderson et al. (2018) has calculated that typically approximately 200 trees should be assessed per orchard management unit, for a 0.95 probability and a percentage error of 10% on fruit load estimation. This level of human based sampling effort is impractical.

### Machine vision in the orchard

Machine vision can be applied in-field for fruit load estimation, extending the application of this technology from its current use in the fruit pack-house. RGB camera images of tree canopies have been coupled to depth camera images to provide information on fruit size distribution within an orchard. For example, Wang et al. (2017) employed a time-of-flight depth camera on the (6 km/h) moving platform used in the current study to estimate mango fruit lineal dimensions to a RMSE of 5 mm, and applied an allometric relationship to relate lineal dimensions to fruit weight and tray size category. Given knowledge of fruit size distribution, the remaining element for a practical yield estimation technique for mango is orchard fruit load estimation.

Mango fruit hang on long stalks derived from the panicle, making imaging from the orchard inter-row more practical than imaging from above (e.g., by drone). Mounting of imaging gear to a farm vehicle allows for easy deployment on farm, however, imaging of a whole farm is not a trivial task. For example, driving a medium sized farm with 30 000 trees at 4 m spacing involves 120 km, or 240 km if the imaging rig faces one direction only. Imaging in two directions from a farm vehicle that is traversing the farm for other tasks, such as a spray rig, is a possibility. Further, night imaging is not a constraint in the Australian mango industry, with several operations currently conducted at night, from spraying to harvesting, to avoid day time heat.

In orchard scenes, common challenges for machine vision detection of fruit include variation in illumination, fruit occlusion, fruit orientation and similarities in colour and texture of fruit and foliage (Gongal et al. 2015; Jimenez et al. 2000; Kamilaris and Prenafeta-Boldú 2018; Payne and Walsh 2014; Syal et al. 2013). To minimize illumination issues inherent

in daylight photography, a number of researchers have advocated night imaging with artificial illumination (e.g., Qureshi et al. (2017)) or imaging under a shade structure (Gongal et al. 2015). Bargoti and Underwood (2017a) reported imaging of mango orchards in day light hours using high intensity strobe lighting and very short exposure times to avoid issues in direct sun lighting, but this required a camera capable of microsecond exposures and use of high intensity Xenon strobe lights, increasing deployment cost and complexity.

During the past decade there have been many reports of object classification in orchard scenes (Gongal et al. 2015) involving manually defined ('handcrafted') parameters for features such as intensity, colour space, shape and texture, or histogram of oriented gradients (HOG), local binary patterns (LBP) or Haar-like features (Gongal et al. 2015). For in-field detection of mango fruit, Payne et al. (2013) applied a colour based segmentation technique followed by blob detection, while Payne et al. (2014) and Qureshi et al. (2017) placed emphasis on texture analysis to achieve an improved fruit detection rate. Nanaa et al. (2014) detected mango fruit based on elliptical shape fitting and Kadir et al. (2015) applied texture analysis to define the fruit edge, followed by morphological operations on binary image and ellipse fitting using Randomized Hough Transform (RHT) technique for detection of mango fruit in clusters. Detection errors were associated with leaves of similar shape to fruit and occluded fruit. However, these 'handcrafted' approaches often fail to generalize to other conditions (cultivars, growing conditions, lighting conditions), as reported for the mango application by Payne et al. (2014) and Qureshi et al. (2017).

A generalised multi-scale feature (unsupervised) learning approach based on a sparse autoencoder and a backpropagation neural network was applied by Hung et al. (2015) to estimate almond and apple fruit load (with F1 scores of 84.8 and 87.3, respectively). More recently, deep neural networks have been successfully used in a range of machine vision applications, including medical, automotive, aerospace, defence, consumer electronics and natural language processing. Sa et al. (2016) utilised the deep learning framework of faster regional-convolutional neural network (Faster R-CNN) (Ren et al. 2015) with Oxford Visual Geometry Group network (VGGNet) (Simonyan and Zisserman 2014) for detection of various fruits through transfer learning. In the transfer learning technique used, a pre-trained model was further trained using small numbers of images (ranging from 43 to 136), primarily sourced from the internet, with F1 scores of 0.848, 0.948, 0.938, 0.932, 0.942, 0.915 and 0.828 achieved for rockmelon, strawberry, apple, avocado, mango, orange and sweet pepper, respectively. In the following year, Bargoti and Underwood (2017a) used Faster R-CNN to detect fruit in day-time images of real orchard scenes illuminated with high intensity strobe lights, reporting higher F1 scores with the deeper (i.e., more layers) VGGNet than when using the Zeiler and Fergus network (ZFNet) (Zeiler and Fergus 2014) (0.904, 0.908 and 0.775 compared to 0.892, 0.876 and 0.726, for apple, mango and almond fruit detection, respectively). These F1 scores were higher than that achieved using pixel-wise fruit segmentation with a Convolution Neural Network (CNN) followed by Watershed Segmentation (WS) and blob detection (F1 scores 0.861 and 0.836 for apple and mango, respectively) (Bargoti and Underwood 2017b) but lower than those reported by (Sa et al. 2016) for apple and mango detection, presumably as images of real orchard scene present more challenges (such as varying illumination, large number of fruits per image, low pixel count per fruit) for training and testing fruit detection models than for close-up images collected from a greenhouse and Google search. More recently, Wang et al. (2018) reported on the use of the Faster R-CNN(VGG) deep learning architecture for task of segmenting and counting mango panicles in canopy images, and compared the result to a panicle associated pixel count (but not panicle count) achieved using a traditional method based on handcrafted colour based features.

## Aim

Deep learning architectures appear to hold value for fruit load estimation from orchard canopy images. However, in reports to date (e.g., Bargoti and Underwood 2017a, Sa et al. 2016 and Wang et al. 2018), image processing has utilised high performance computing (HPC) resources over a period of days following the actual imaging event. For practical application, an architecture was sought that was sufficiently light (i.e., fewer convolutional and detection layers) to allow deployment on readily available PCs, to allow for real time application in orchard, while maintaining accuracy.

The ‘need for speed’ is illustrated in the following calculations. For a medium sized farm of 30 000 trees there are 60 000 (dual-view) 5 Mega pixels images that are currently uploaded by satellite link by farm management for image processing on a HPC resource. Local processing would avoid the need for data transfer off-farm. If processing took 1 s per image, the 60 000 images would consume 17 h of computing resources, either in the farm office or during imaging. At the typical speed of 6 km/h of a spray rig (a potential carrier for an in-field tree imaging system) and a tree spacing of 4 m, there is 2.4 s between trees, but only a fraction of this time is available for image processing, especially for systems capturing views in two directions (both rows), or employing multiple images per tree or video for fruit tracking between images (e.g., Stein et al. 2016). Further, real-time detection of fruits is a precursor to use in an autonomous harvesting system.

It is difficult to assess the performance of deep learning architectures across published studies because of difference in datasets, computing hardware, network parameters and performance metrics. The current study undertook a comparison using common datasets of the deep learning architectures employed to date in fruit load estimation (i.e., Faster R-CNN with VGG and ZF, two stage detectors), and added consideration of current ‘state of art’ single stage detector architectures, which have not yet been reported in context of fruit detection to our knowledge. The one-stage detectors offer increased speed of operation. Further, reported work to date has been based on use of ‘off the shelf’ architectures that were trained for fruit detection, but were created for a purpose other than the current application (mango fruit load estimation). Therefore a new deep learning architecture was proposed, based on the single stage architectures, to optimize memory and speed of detection for the task of mango fruit detection, and performance benchmarked to the existing ‘off-the-shelf’ algorithms.

Compared with previous published work, the current work contributes to the development of a solution for fruit detection and yield estimation by examining the hypothesis that a single stage deep learning detector is faster than a two stage detector with similar accuracy, and that a deep learning architecture can be optimized for speed and accuracy for a given application (mango fruit detection in this study). A target processing speed of 500 ms on a field deployable computer was set to allow for in-field image processing. The ultimate goal is provision of a yield estimate to growers to within 15% of fruit harvest count (i.e., information useful to farm management; Walsh and Wang 2018). This aim requires accurate assessment of both fruit seen in images (i.e., a high F1 score) and the proportion of fully occluded fruit. Fruit load estimation of five orchards is estimated to gauge utility of the estimation technique.

## Deep Learning: object detection frameworks

A brief review of deep learning methods is presented to place the current work in context. Deep learning methods have been reported to outperform traditional approaches in the majority of object segmentation and classification tasks in agriculture, with the automatic feature extraction capabilities of deep learning architectures reported to solve complex problems more easily, with greater accuracy and better generalization than traditional approaches (Kamilaris and Prenafeta-Boldú 2018). While a greater number of training image is required for the deep learning methods than those based on hand-crafted features, the need for labour intensive and expert knowledge for feature extraction is avoided (Bargoti and Underwood 2017a; Kamilaris and Prenafeta-Boldú 2018).

The training of the deep learning frameworks involves optimization of a cost/loss/objective function (combination of classification, confidence and box regression losses) for minimum error in object classification and localization. As deep learning architectures have millions of parameters to be optimized, these models are commonly trained on large datasets, such as ImageNet (450 k images in 200 classes, (Deng et al. 2009), PASCAL VOC (12 k images in 20 classes, (Everingham et al. 2010) and COCO (120 k images in 80 classes, (Lin et al. 2014). These online datasets provide labelled data (images and annotation files for different object classes) free for training and benchmarking object detection models. While these sets do not contain mango orchard images, deep-learning models can be reused in new applications, with the learned features transferred through transfer learning. Transfer learning is a method involving use of a relatively small number of images for ‘re-training’ or ‘fine tuning’ the knowledge (convolutional weights) of a model trained on a very large dataset, to create a new model.

Current state of the art object detection frameworks include Faster R-CNN (Ren et al. 2015), You Only Look Once (YOLO) (Redmon and Farhadi 2018), Single Shot Multi-Box Detector (SSD) (Liu et al. 2016), RetinaNet (Lin et al. 2017b) and Mask R-CNN (He et al. 2017). There is a fast pace of developments in this field. For example, the recent review and survey of deep learning in agriculture by Kamilaris and Prenafeta-Boldú (2018) did not report on use of single stage detectors (SSD and YOLO).

Two stage detectors like Faster R-CNN have a region proposal network as a first stage which proposes possible areas likely to contain objects (regions of interest, ROI), followed by a feature extractor (a base CNN such as VGGNet or ZFNet) that extracts visual features from the ROIs to determine the presence of object. A second stage undertakes classification and bounding box regression. This pipeline is generally considered to be too slow for tasks requiring real-time processing, e.g., on a GPU (GeForce GTX Titan X) hardware, Redmon and Farhadi (2017) reported processing at 7, 46 and 91 frames per second using Faster R-CNN with VGG-16, SSD-300 and YOLOv2-288, respectively. In a single stage detector, there is no region proposal stage as the prediction is made directly on the input image in single forward pass through the CNN. A single stage detector like YOLO converts the input image to a vector of scores and produces coordinates for the predicted boxes with a single CNN, making a very fast detector.

A general object detection framework involves: (i) input image pre-processing (resizing, normalizing, color space change etc.); (ii) detection of objects; (iii) placement of a bounding box (usually rectangle) around the detected objects for the task of object localization; (iv) a calculation of class specific confidence score for each detection; and (v) post processing filtering of detections on the basis of a confidence score criterion,

with merging of multiple detections of one object using the non-maximum suppression (NMS) technique to suppress detections based on an overlap threshold.

In the current study, three different state-of-the-art deep learning architectures for object detection (Faster R-CNN, SSD, and YOLO) were trained and tested for mango fruit detection.

### Faster R-CNN

Faster R-CNN (Ren et al. 2015) is a two stage complete object detection framework which replaced the selective search method of R-CNN (Girshick et al. 2014) with a region proposal network (RPN), with the advantage of higher detection speed. RPN slides a  $3 \times 3$  window across the final feature map and at each window location considers  $k$  different anchor boxes centred on the location to generate possible region proposals. Each region proposal consists of an objectness score for that region as well as the co-ordinates of the boxes. The regional proposals are filtered based on objectness threshold and passed to what is essentially a Fast R-CNN (Girshick 2015) for object detection. In general, Faster R-CNN can be regarded as an RPN on the top of Fast R-CNN. Faster R-CNN predicts bounding boxes using handpicked anchors ( $k=9$ ) composed of three scales and three aspect ratios and employs one detection layer. The base CNNs ZFNet and VGGNet (VGG-16) have 7 and 16 layers respectively.

### SSD

SSD (Liu et al. 2016) is a one stage complete object detection framework. Unlike Faster R-CNN which performs region proposal and classification separately, SSD simultaneously predicts the object class and places a bounding box on the image. An input image is fed through a series of layers in the base CNN, generating different sets of feature maps at different scales. For each feature map position, a  $3 \times 3$  convolutional filter is used to assess a set of default anchor boxes referred to as 'priors'. Boxes are then drawn and classified at every single position in the image and at several scales, generating a very large number of negative examples and few positives. To address the class imbalance problem, SSD uses a technique called 'hard negative mining' to balance the positive and negative classes during training, keeping the negatives to positives ratio to 3:1 by using only the subset of negatives with highest training loss.

### YOLO

YOLOv2 (Redmon and Farhadi 2017), a single stage complete object detection framework, is an improvement on YOLO (Redmon et al. 2016) making it better, faster and stronger. Class probabilities, objectness scores, and bounding box coordinates are predicted from the final feature map in one evaluation (forward pass) making it one of the fastest object detection methods. A pass-through layer is used that concatenates features from the higher resolution  $26 \times 26$  layer to a  $13 \times 13$  layer. This process may have an advantage in small object detection involving finer grained features (Redmon and Farhadi 2017). The backbone feature extractor for YOLOv2 is the Darknet-19 framework which has 19 convolution layers, mostly using  $3 \times 3$  filters similar to VGGNet. YOLOv2 has 22 convolutional layers and 1 detection layer. YOLOv2(tiny) has a small architecture, with just 9 convolutional layers, 6 pooling layers, and 1 detection layer, sacrificing accuracy for speed (Redmon 2018).

YOLOv3 (Redmon and Farhadi 2018) is based on Darknet-53 and is an improvement over YOLOv2 (Redmon and Farhadi 2017). Darknet (Redmon 2018) is an open source deep learning framework written in C and CUDA. Darknet-53 is a new feature extractor (53 convolution layers), improving on Darknet-19 by addition of successive  $3 \times 3$  and  $1 \times 1$  convolutional layers with skip connections similar to ResNet (He et al. 2016). Like SSD, YOLOv3 uses the concept of feature pyramids (Lin et al. 2017a) to extract features at three different scales for box predictions. In order to capture more meaningful and fine-grained information, the feature maps from lower layers are merged with up-sampled feature map from higher layers and processed further. Like Faster R-CNN, YOLOv3 uses logistic regression to predict objectness score of bounding boxes but only one bounding box anchor is assigned per ground truth object. YOLOv3 employs 75 convolution layers and 3 detection layers.

## Materials and methods

### Orchard sites

Images were acquired of all trees in each of five mango (*Mangifera indica*) orchards (farm management units) located in Queensland, Australia, on December 7 and 8, 2017 (Table 1). The orchard E set represented only a part of an orchard. Imaging occurred at least 6 weeks before harvest, but after the ‘stone hardening’ stage (after which stage fruit drop is typically small). The orchards varied in cultivar and canopy structure.

### Field imaging hardware

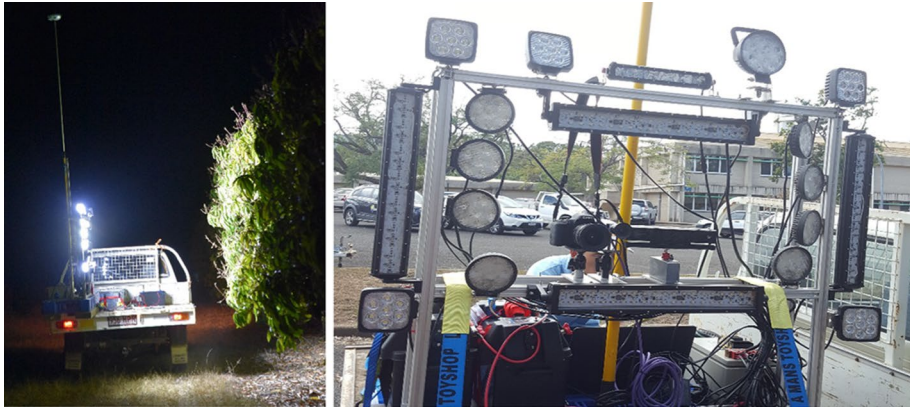
The field imaging rig (Fig. 1) consisted of an aluminium frame ( $1 \times 1$  m) with combination of LED bars (SCA, 126 W, 5 400 Lumens) and LED flood lamps (Calibre, 35 W, 2 500 Lumens) totalling 720 W, powered with a 24 V DC lead acid battery. Reflectors were removed from the LED bars to improve light spread. A Basler acA2440-75um machine vision camera (75 fps, 5 Mp,  $2\,448 \times 2\,048$  pixels) with a Goyo (GM10HR30518MCN) lens (5 mm focal length) was operated at FStop 3, gain 5 and exposure 2.5 ms for RGB imaging. The rig also carried a Canon EOS750D with 10–22 mm lens and a Kinect RGB-D camera, as described in (Wang et al. 2018). Camera colour calibration was not performed. Images of tree canopies were captured at night with a typical camera to canopy distance of 2 m. The rig was mounted to the tray of a farm utility vehicle and operated after sunset, with the vehicle driven at a speed of about 6 km/h. The use of night imaging allows for consistent illumination conditions compared to the variation experienced in daylight hours due to sun angle and weather conditions (Payne et al. 2013). The camera was triggered automatically, with reference to the pre-acquired GNSS position of trees. As a one-off task, every tree was geo-located to an accuracy of within 2 cm using a Leica GS-14 unit operating on a Continuously Operating Reference Stations (CORS) network. The receiver was mounted to a mast above average canopy height. Imaging of an orchard consisted of collection of two images per tree, imaged from opposite sides of each tree (termed ‘dual-view’ images). The system has been previously reported in (Wang et al. 2018) and evolved from the systems used by Payne et al. (2013), Qureshi et al. (2017) and Underwood et al. (2018).

**Table 1** Description of orchards used for imaging

Location lat, long	Orchard	Cultivar	Row spacing (m)	Tree spacing (m)	Tree height (m)	Canopy width (m)	# Trees imaged	# Sample trees	# Harvested fruit
- 25.144, 152.377	A	Calypso	9.5	4	3	4	494	17	3 519
- 25.144, 152.377	B	Calypso	10.5	4	4.5	4	121	6	1 676
- 25.144, 152.377	C	Calypso	12	4	4	4	256	12	1 784
- 25.162 152.351	D	Honey Gold	8	4	3	2.8	566	18	1 302
- 25.205, 152.284	E	R2E2	9	5	3.3	2.8	78	18	729

Sample trees refers to trees that were manually harvested for a ground truth fruit load estimate (# Harvested fruit)





**Fig. 1** Lighting and imaging camera rig mounted on a farm utility vehicle, operated at 6 km/h (left panel), with components of LED floodlights, RGB camera and Time of Flight camera (right panel)

## Computing hardware

Model training and testing was implemented on the CQUniversity High Performance Computing (HPC) facility graphics node with following specifications: Intel® Xeon® Gold 6126 (12 cores, 2 600 MHz base clock) CPU, NVIDIA® Tesla® P100 (16 GB Memory, 1 328 MHz base clock, 3584 CUDA cores) GPU. Red Hat Enterprise Linux Server 7.4 (Maipo) and 384 GB RAM. CUDA v9.0, cuDNN v7.1.1, OpenCV v3.4.0, Python v2.7.14, GCC v4.8.5.

For in-field use, a Nuvo-6108GC industrial-grade computer with following specifications was used: Intel® Core™ i7-6700TE CPU @ 2.40 GHz, 32 GB RAM, NVIDIA GeForce GTX 1070 Ti GPU (1 607 MHz GPU clock) with 8 GB dedicated memory (2 002 MHz memory clock), 64 bit Windows 10 Pro, CUDA v9.1, cuDNN v7.0.5, OpenCV v3.4.0.

## Image pre-processing

All object detection methods re-size the input image to a specific resolution ('network resolution') for training. The feature extractors (base CNN) used by deep learning frameworks usually require a square input resolution. In its original configuration, Faster R-CNN will resize the input image such that the shorter dimension of height or width is resized to 600 pixels, while keeping the image aspect ratio unchanged. SSD-300 and YOLO resize input images to  $300 \times 300$  pixels and  $416 \times 416$  pixels, respectively. Network resolution can be increased to accept larger input images (e.g.,  $2\,048 \times 2\,048$  pixels), but at the cost of increased memory and computation requirement. The resizing of Basler  $2\,448 \times 2\,048$  pixel images to  $717 \times 600$  pixels for use in Faster R-CNN training resulted in a decrease in typical fruit image size from  $\sim 16 \times 16$  pixels to  $\sim 7 \times 7$  pixels.

Network stride, the factor by which network scales down the input image to its final feature map, is 16 and 32 in Faster R-CNN (for ZF and VGG) and YOLO, respectively. With a stride of 16, a  $16 \times 16$  pixel image corresponds to one pixel in the final feature map. To have object size greater than the stride size on the final feature map,  $2\,448 \times 2\,048$  pixel

images were split into tiles (sub-images) of  $612 \times 512$  pixels with a  $4 \times 4$  grid, maintaining the original image aspect ratio. This allowed for training of different detection frameworks with similar network resolution (around  $512 \times 512$  pixels) and object size, while reducing memory requirement in training.

Training object detection models requires labelled data, i.e., the class-label and the position (co-ordinates) of all ground truth bounding boxes in training images. While labelling is a manual and labour intensive process, annotation (drawing of ground truth bounding boxes) was easier on tiles than on the full image, as the lower number of fruits on a tile compared to a full image reduces chances of human error. The graphical image annotation tool *labelImg* (<https://github.com/tzutalin/labelImg>; accessed 15/08/2017) was used to hand label all the ground truth bounding boxes, with annotation files saved in PASCAL VOC format.

## Image sets

The different object detection frameworks, i.e., Faster R-CNN, SSD and YOLO, require different data formats for processing image data during training and testing. Datasets and directories were structured similar to the PASCAL VOC dataset (Everingham et al. 2010), avoiding the need to change scripts, with the detection frameworks parsing PASCAL VOC annotations into their format. For each processed image, an XML annotation file (file-name= image name) was created containing the image attributes (name, width, height) and the object attributes (class name, object bounding box co-ordinates). All image data is available at <http://hdl.cqu.edu.au/10018/1261224>.

Tiles of  $612 \times 512$  pixels were randomly drawn from images from three rows of orchard A. Tiles with no fruit were excluded. Training was undertaken using a set of  $1 \times 300$  tiles, with validation undertaken on a further 130 tiles from the same set of trees (Table 2). Test set 1 consisted of 300 tiles from three different rows of the same orchard, and was split into three subsets containing 100 tiles each. Subset L (Low complexity) contained tiles with well separated fruits. Subset M (Medium complexity) contained tiles with some clustered (partially occluded) fruits. Subset H (High complexity) contained tiles with many fruits in clusters. Test set 2 consisted of images 6–18 trees in each of five orchards (A to E, Table 2). The fruit load of these trees was manually assessed at harvest. There was no overlap of the orchard A prediction set with images used in the training and validation sets. The trees of test set 2-A were imaged with Canon and Kinect cameras, creating two further image test sets (2A-can and 2A-kin, respectively). Finally, the training and validation set images used by Bargoti and Underwood (2017a), accessed from <https://data.acfr.usyd.edu.au/ag/treecrops/2016-multifruit/>, were employed (Train set 2-bu and Test set 3-bu, respectively).

Trained models were also compared in terms of prediction results for sets of 6–18 trees in each of five orchards (Test set 2, Table 2), for which fruit load was manually assessed at harvest. Models trained using images from Orchard A (Train set 1) were compared in terms of prediction results for the image test sets from different orchards (differing in growing conditions and cultivars, Test set 2 A-E) and from different cameras (Test set 2A, 2A-can and 2A-kin).

Further benchmarking of the model developed in the current study was undertaken by comparison to the Faster R-CNN(VGG) and Faster R-CNN(ZF) models used by Bargoti and Underwood (2017a), employing their training and test sets (Train set 2-bu and Test set 3-bu; Table 2). These images were acquired of trees in orchard A in the season preceding imaging undertaken for the current study, and were captured in daytime using a 20 Mp

**Table 2** Description of image tile sets used in training, validation (model tuning) and testing

Dataset names	Number of images or tiles	Image size (pixels)	Total # fruits in images	Cultivar
Train set 1	1 300	612×512	11,820	Calypso
Validation set 1	130	612×512	861	Calypso
Test set 1 All	300	612×512	2 600	Calypso
Test set 1 Low	100	612×512	341	Calypso
Test set 1 Medium	100	612×512	636	Calypso
Test set 1 High	100	612×512	1 623	Calypso
Test set 2A	34	2 448×2 048	2 151	Calypso
Test set 2B	12	2 448×2 048	964	Calypso
Test set 2C	24	2 448×2 048	842	Calypso
Test set 2D	36	2 448×2 048	1 229	Honey Gold
Test set 2E	36	2 448×2 048	552	R2E2
Test set 2A-can	34	6 000×4 000	2 137	Calypso
Test set 2A-kin	34	1 920×1 080	1 746	Calypso
Train set 2-bu	1 154	500×500	7 065	Calypso
Test set 3-bu	270	500×500	947	Calypso

The training, validation and test set 1 were from images of trees of two rows from within orchard A (cultivar Calypso). Suffixes 'can' and 'kin' refers to Canon and Kinect camera images, while 'bu' refers to an image set from Bargoti and Underwood (2017a)

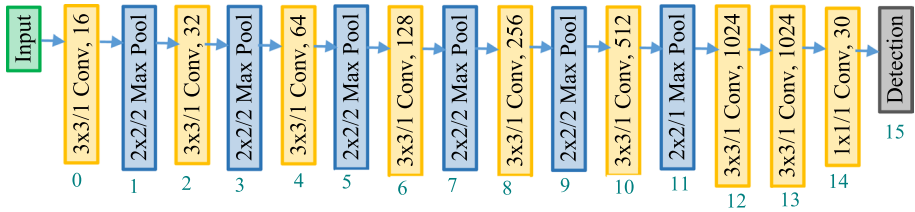
Prosilica GT3300c camera and Xenon strobe lamps with a very short exposure time to reduce the effect of sunlight. The images are visually more complex and challenging compared to the night acquired images of the current study, because the images are relatively under-exposed, and the background is illuminated.

## Deep learning architectures

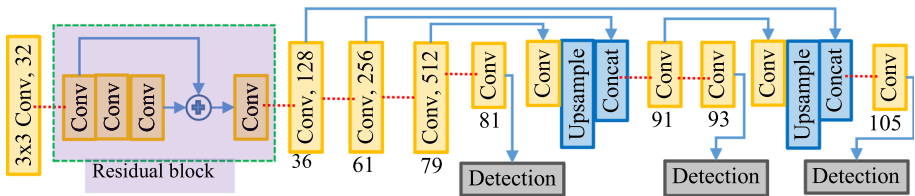
### Architectures used

Open source codes of the deep learning algorithms were downloaded from their official repositories through the links provided by Redmon et al. (2016) for YOLO, by Liu et al. (2016) for SSD, and by Ren et al. (2015) for Faster R-CNN. As a general naming convention, deep learning model names are suffixed with a number representing the input image resolution of the network. In the current paper, any model names with suffix *-original* refers to a model used with the default network resolution, as available in the official repositories. The original architectures require input of 416×416 and 300×300 pixels for YOLO and SSD respectively. Faster R-CNN on other hand resizes and rescales the input image to make the shorter side equal to 600 pixels (e.g., 612×512 pixel input image gets resized to 600×717 pixels). The suffix *-512* on a model name refers to the models whose network resolution was changed to 512×512.

The algorithms implemented in this study are sourced from open source code available from online repositories. The YOLO repository is constantly maintained, with more features added (e.g., object tracking) and frequent code optimizations for better speed and



**Fig. 2** Block diagram of architecture YOLOv2(tiny)



**Fig. 3** Block diagram of architecture YOLOv3

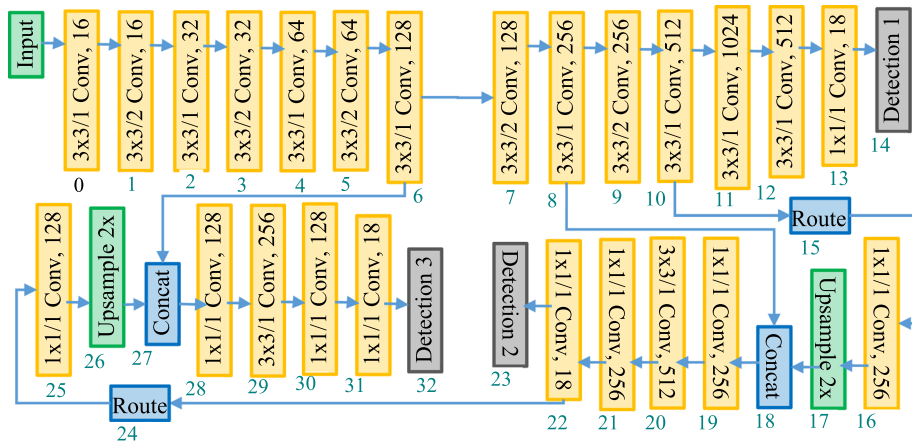
accuracy. Performance (speed and memory consumption) results for YOLO variants in HPC and Nuvo hardware are presented for the code implementation of 12/2018.

Redmon and Farhadi (2018) note that the YOLO object detection framework allows for a trade-off between speed and accuracy. Network resolution can be changed to detect objects on different input image resolutions without the need for further training, and the heavy data augmentation (modification of training image hue, saturation, rotation, jitter, multiscale etc.) available with YOLO allows for the training of a robust model with a lower number of training images. YOLOv2(tiny) (Fig. 2) is reported to have the highest speed and lowest accuracy of the YOLO variants. According to Redmon and Farhadi (2018), YOLOv3 is deeper (i.e., has more convolutional layers; Fig. 3) than the previous YOLO versions, but is as accurate and 3.0 and 3.8 times faster than SSD and RetinaNet, respectively.

## YOLO re-design

An attempt was made to optimize a model for speed, memory requirement and accuracy through a re-design of the YOLO framework, to take advantages of both YOLOv2(tiny) (fewer layers and higher speed) (Fig. 2) and YOLOv3 (multiple detection layers and high accuracy) (Fig. 3). The design of *MangoYOLO-512* incorporated 33 layers (Fig. 4), in comparison to 106 layers in YOLOv3 (Fig. 3) and 16 in YOLOv2(tiny) (Fig. 2).

The main rationale behind the modifications was to allow detection on multiple feature maps from different layers of the network on the premise that this would allow for accurate detection of mango fruit even with a reduced number of layers in the network. Similarly, it was reasoned that features from the early stages (layers) would assist detection of smaller and darker fruit whose pixel information could be lost when passed through a large number of layers in a deeper network. Moreover, the decrease in the number of layers was expected to result in decreased computation/detection time. *MangoYOLO* (Fig. 4) (33 layers) was created by modification of YOLOv2(tiny) as follows:



**Fig. 4** Block diagram of architecture of *MangoYOLO*

- All 6 max-pooling layers in YOLOv2(tiny) were replaced with convolution layers as implemented by YOLOv3. Convolution layers are considered to have the same effect as pooling layers but are considered computationally more efficient (Springenberg et al. 2014).
- In Tiny YOLOv2, layer 13 (with 1,024 filters) is a replica of layer 12 (Fig. 2). This layer was removed, reducing computation time with no noticeable change in performance.
- To capture fine-grained information, feature maps from intermediate layers (6, 8 and 10) were further processed by up-sampling to merge with feature maps of different resolutions followed by four convolution layers, before detection (Fig. 4).
- The first detection was implemented at layer 42 in *MangoYOLO*, intermediate between the positions implemented in YOLOv3, YOLOv2(tiny) (at layer 82 and 15, respectively).
- Total depth was 33 layers, less than one-third of YOLOv3 (107 layers), and so involves less computation resulting in an expected improved detection speed.
- Unlike YOLOv3 that uses residual block (successive  $3 \times 3$  and  $1 \times 1$  convolutional layers with skip connections similar to ResNet) (Fig. 3), the *MangoYOLO* architecture involves convolution layers as in YOLOv2(tiny) (Fig. 4). A residual block is useful for resolving the ‘vanishing gradient’ problem in training of deeper networks (He et al. 2016). This implementation was not necessary for shallower networks like *MangoYOLO*.
- As in YOLOv3 implementation, detections in *MangoYOLO* were made at three scales on feature maps of  $16 \times 16$ ,  $32 \times 32$ ,  $64 \times 64$  pixels for input images size  $512 \times 512$  pixels, which were appropriate for the mango image dataset (average fruit size  $43 \times 48$  pixels and minimum fruit size  $16 \times 15$  pixels in the Train set 1) under current study (Table 2).

## Model training

### Number of training images

The number of training images required by deep learning models largely depends on the visual complexity of the images in training and test datasets, the network architecture,

image augmentation techniques and machine learning parameters of the network. Similar to (Sa et al. 2016) and (Bargoti and Underwood 2017a), an experiment was performed to provide a guide to the number of training images required for training of a deep learning architectures, with default network parameters. The training set of 1 300 tiles was sampled randomly for subsets of 10, 50, 100, 200, 400, 600, 800, and 1 300 tiles, with selection of 100 and 600 subsets repeated three times for an estimate of the impact of sampling variation. *MangoYOLO(s)* models were created based on training with each subset, with optimisation on the number of iterations in each case based on average training loss following <https://github.com/AlexeyAB/darknet#when-should-i-stop-training>.

## Pretraining

All models but *MangoYOLO(s)* and (*sbu*) were initialized with pre-trained models/weights (referred to as transfer learning) for training. *MangoYOLO(s)* was trained from scratch (random weight initialization of convolution filters) as a custom designed CNN architecture with no existing pre-trained weights. Pre-trained weights for initializing *MangoYOLO(pt)* and (*ptbu*) were obtained by training *MangoYOLO(s)* and (*sbu*), respectively, on the COCO dataset (following the instruction from <https://pjreddie.com/darknet/yolo>) for 120 K iterations. VOC 2007 pre-trained Caffe models (ZF and VGG16) were downloaded for Faster R-CNN by running the `fetch_fsater_rcnn_models.sh` script provided on the repository (<https://github.com/rbgirshick/py-faster-rcnn>, accessed 21/02/2018). The ImageNet Large Scale Visual Recognition Competition (ILSVRC) pre-trained VGG16 model was downloaded for SSD from the link provided in the official repository (<https://github.com/weiliu89/caffe/tree/ssd>, accessed 16/03/2018). ImageNet pre-trained convolutional weights for YOLOv2 (Darknet-19\_448.conv.23) and YOLOv3 (darknet53.conv.74) were downloaded from the links provided in the YOLO official website (<https://pjreddie.com/darknet/yolo>, accessed 15/03/2018) (Redmon 2018).

## Training with fruit images

Models were trained on the training set (Train set 1; Table 2) using default parameters of the networks (YOLO, SSD and Faster R-CNN). The trained models were then tuned for the highest F1-score using the validation set. F1 score is the weighted average (harmonic mean) of precision and recall, and varies between 0 and 1. To tune the model on the validation set, non-maximal suppression (NMS) was varied across the range 0.1 to 0.6, in 0.1 steps, but including 0.35 and 0.45, while the confidence threshold was kept to the default minimum values for each detection framework. The NMS threshold and class confidence threshold associated with the highest F1 score were used as the ‘tuned’ parameters for the model for fruit detection on test sets.

Detection files were generated for the test set using the scripts: `test_net.py` for Faster R-CNN and `score_ssd_pascal.py` for SSD from their official repositories. Generation of detection files in YOLOv2/v3 required passing ‘*valid*’ as a command line argument to Darknet’s ‘*detector*’ function. To generate precision and recall scores as well as the average precision, the script ‘`voc_eval_py3.py`’ included in the Faster R-CNN official repository was used. This script was also modified to output the highest F1 score and associated confidence cut-off thresholds for model tuning.

Two models were created using the *MangoYOLO* architecture, one involving training from scratch (no transfer learning used) on the mango fruit training image set



(*MangoYOLO(s)*-512), while the second model involved training on the COCO dataset (following <https://pjreddie.com/darknet/yolo/>) for 120 K iterations before training on the mango fruit image training dataset (*MangoYOLO(pt)*-512).

## Model comparisons

Seven models were compared, relative to human labelling of images, using: test sets 1 (orchard A test set tiles) and 2 (images from trees from each of five orchards) (Table 2). In the latter case, models trained on  $512 \times 512$  pixel tiles were applied directly on the full ( $2\,448 \times 2\,048$  pixels) images by setting the model input resolution to  $2\,048 \times 2\,048$  pixels in configuration files. The increase in network resolution allowed processing of large images without need for tiling, i.e., the model weights remained the same but the network used larger feature maps for fruit detection. Fruit pixel size remained the same whether it was from either tiled image or a full image.

*MangoYOLO* was also compared to the Faster R-CNN(VGG) and Faster R-CNN(ZF) results of Bargoti and Underwood (2017a). For the latter case, a *MangoYOLO* model (*MangoYOLO(sbu)*-512) was trained from scratch with the training set of Bargoti and Underwood (2017a) (Table 2) for comparison to a model trained with the night imaging training set used in other exercises in the current study (*MangoYOLO(s)*-512). These exercises were undertaken to gauge model robustness to plant phenotype and to lighting conditions.

The *MangoYOLO(s)*-512 model, trained on orchard A training tiles extracted from images acquired with a Basler camera, was also used in detection of fruit in images from the Canon and Kinect cameras for the 17 test trees of orchard A. This exercise was undertaken to demonstrate model robustness across camera platforms.

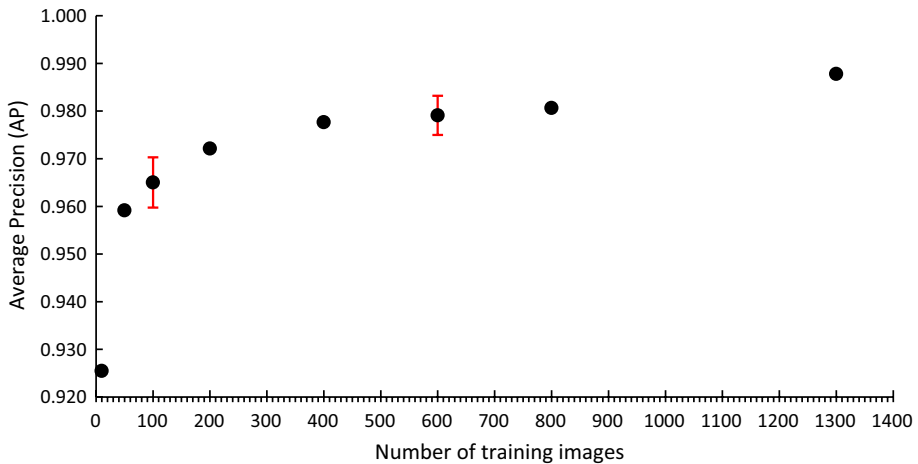
## Orchard fruit load estimation

To estimate the fruit load of an orchard from a machine vision count of two sides of all trees, a correction factor was calculated for ‘hidden’ fruit (i.e., fruit not seen in either side view of the ‘dual-view’ per tree). The correction factor per orchard was calculated as a ratio of the sum of the number of fruit (human count) in images of each side of the canopy to the hand harvest count per sample tree (approximately 18 trees per orchard, Table 1). The total number of fruit detected on the block by machine vision was multiplied by the correction factor to estimate the total fruit count per block/orchard. Three estimates of fruit load for each of the five orchards were made, based on use of Faster R-CNN with VGG, YOLOv3 and *MangoYOLO* models, with comparison to the pack-house count of harvested fruit. Percentage error on the machine vision estimate was calculated as  $\frac{SE_{A \times B}}{A \times B} \times 100$ , where  $SE_{A \times B} = A \times B \times \sqrt{\left(\frac{a}{A}\right)^2 + \left(\frac{b}{B}\right)^2}$  and A is the hidden fruit correction factor and a is its associated standard error, B is the machine vision count per tree and b is its associated RMSE.

## Results

### Number of training images

The AP of *MangoYOLO(s)* models in prediction of validation set 1 improved with increasing number of training tiles, increasing from a score of 0.925 with just 10 training images,



**Fig. 5** Average Precision for *MangoYOLO(s)*-512 on validation set 1 (Table 2) plotted against the number of training images from Training set 1. Error bar represents standard deviation on three repeated assessments

to a plateau around 0.98 after about 400 training images (Fig. 5). The *MangoYOLO(s)* model benchmarked to other models in this study was created using the full set of 1,300 training tiles.

### Model performance

Eight models using an image resolution of  $512 \times 512$  pixels were evaluated in terms of speed in training and inference, and model size. Image batch size was set at 1 (i.e., only one image loaded into memory during training). *MangoYOLO(s)*/*MangoYOLO(pt)* had the smallest size and a similar GPU memory usage as YOLOv2 (tiny)-512, i.e., the least of the seven models (Table 3).

**Table 3** GPU memory consumption in training and testing time for models using  $512 \times 512$  pixel images and final size of the trained model weight files

Models	GPU memory consumption		Trained weights
	Training (Mb)	Inference (Mb)	Size (Mb)
Faster R-CNN(VGG)-512	2 739	1 759	533.948
Faster R-CNN(ZF)-512	1 719	1 123	230.105
SSD-512	2 119	1 259	92.759
YOLOv3-512	2 101	2 097	240.533
YOLOv2-512	1 643	1 639	261.957
YOLOv2(tiny)-512	849	845	61.604
<i>MangoYOLO(s)</i> -512	877	873	53.776
<i>MangoYOLO(pt)</i> -512	877	873	53.776

Values assessed on HPC hardware and using Darknet repository (<https://github.com/AlexeyAB/darknet>, accessed on 15/03/2018) in the YOLO variants



**Table 4** Model performance (F1-score and AP) for fruit count on Validation set 1 (512×512) images and average detection/inference times for different models

Models	F1-score	Average precision	Inference time (ms)
Faster R-CNN(VGG)-original	0.936	0.918	67
Faster R-CNN(VGG)-512	0.945	0.953	67
Faster R-CNN(ZF)-original	0.929	0.933	37
Faster R-CNN(ZF)-512	0.939	0.950	37
SSD-300-original	0.950	0.983	46
SSD-512	0.959	0.973	70
YOLOv3-512	0.951	0.967	25
YOLOv2-original	0.916	0.945	20
YOLOv2-512	0.933	0.959	20
YOLOv2(tiny)-original	0.900	0.938	10
YOLOv2(tiny)-512	0.917	0.953	10
<i>MangoYOLO(s)</i> -512	0.967	0.986	15
<i>MangoYOLO(pt)</i> -512	0.968	0.983	15

Values assessed on HPC hardware and using Darknet repository (<https://github.com/AlexeyAB/darknet>, accessed on 15/03/2018) in the YOLO variants

A further five models trained using images at original resolution were evaluated. The required memory allocation more than five-fold, e.g., YOLOv3 GPU memory requirement on the HPC increased from 2 101 Mb (Table 3) to more than 16 GB (data not shown) for inference using 512×512 and 2 048×2 048 pixel image resolution, respectively. This memory requirement exceeded capacity so resolution was set to 1 888×1 888 pixels, for which 16 GB was required (Table 5).

**Table 5** Detection speed and memory usage for three models used with full canopy images (2048×2048)

Model	Network input resolution (pixels)	HPC detection time (ms)	HPC GPU memory (Mb)	Nuvo detection time (ms)	Nuvo GPU memory (Mb)
MangoYOLO(s)/ MangoYOLO(pt)	512×512	<1	879	8	833
YOLOv3	512×512	<1	2 103	30	2 055
YOLOv2(tiny)	512×512	<1	851	5	834
MangoYOLO(s)/ MangoYOLO(pt)	2 048×2 048	20	4 513	70	4 417
YOLOv3	1 184×1 184	–	–	123	6 805
YOLOv3	1 888×1 888	20	16 273	–	–
YOLOv2(tiny)	2 048×2 048	10	3 279	51	3 270

Values assessed on HPC hardware and using Darknet repository (<https://github.com/AlexeyAB/darknet>, accessed on 21/11/2018) in the YOLO variants. Within the GPU memory constraint the maximum network resolutions for YOLOv3 were 1 888×1 888 and 1 184×1 184 for HPC and Nuvo computing platforms respectively (note: YOLO requires the network input resolution to be multiple of 32)

All models achieved good detection results on Validation set 1 (F1 scores > 0.9, Fig. 7a), with the highest F1 scores associated with *MangoYOLO(pt)-512* (0.968), *MangoYOLO(s)* (0.967), SSD-512 (0.959) and YOLOv3-512 (0.951, Table 4). Tiny YOLO-original had the lowest F1 score (0.90). *MangoYOLO(s)-512* achieved the highest average precision (AP=0.986), with similar results achieved for *MangoYOLO(pt)* (AP=0.983), and SSD-300 (AP=0.982), while the lowest AP was associated with Faster R-CNN-VGG-original (AP=0.917) (Table 4).

The detection/inference times on  $512 \times 512$  pixels images was lower for YOLO models than other models (Table 4). The shortest inference time was achieved by YOLOv2(tiny) (10 ms) and *MangoYOLO(pt)-512/MangoYOLO(s)-512* (15 ms), while SSD-512 model required 70 ms (on a HPC resource). For full canopy images ( $2048 \times 2048$  pixels), *MangoYOLO(s)/MangoYOLO(pt)* achieved a good detection speed of 70 ms per image (~14 fps), and consumed the least memory of the three models compared (Table 5). For the test set-Overall, the lowest Root Mean Square Error (RMSE) and highest  $R^2$  (correlation coefficient of determination) was associated with the *MangoYOLO* models (Table 6). For the same network resolution, Faster R-CNN(VGG) out-performed Faster R-CNN(ZF) (Table 6).

## Model robustness

### Camera comparison and daylight imaging

For images acquired of the same trees under the same lighting, Canon images were the brightest and Kinect images were the darkest (Fig. 6). *MangoYOLO(pt)* and *MangoYOLO(s)* models were employed in estimation of fruit number per image for images of Test set 2A (17 trees in orchard A) collected using three cameras (Table 7). The *MangoYOLO(pt)* model produced higher fruit counts than the *MangoYOLO(s)* model, but with higher false positive rates and a lower  $R^2$ . The highest false positive rate (ratio of counts of false positive detections to total detections expressed as percentage) was associated with detection using *MangoYOLO(pt)* on Canon images. It was observed that in resizing the Canon images from  $6000 \times 4000$  pixels to  $2048 \times 2048$  pixels that image of some leaves took a curved shape similar to fruit. Moreover, false detections were also registered for parts of branches and tree trunks where the light intensity was high. Less fruit detection occurred for Kinect images using both *MangoYOLO(s)* and *MangoYOLO(pt)* models.

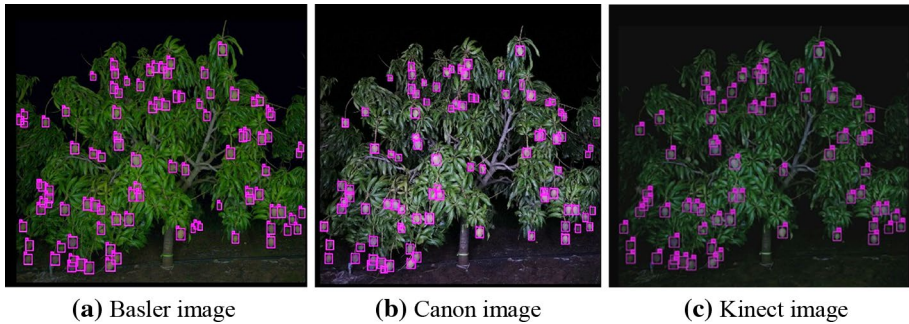
### Bargoti and underwood images

*MangoYOLO-512* was further benchmarked on the mango dataset created by Bargoti and Underwood (2017a) (Table 8). Poor results were obtained for the models trained on the night images acquired with the Basler camera when applied on the daytime/Prosilica camera images. In contrast, when a *MangoYOLO* model was trained using the Bargoti and Underwood (2017a) daytime images, it performed reasonably well on the night-Basler images. *MangoYOLO* trained using the Bargoti and Underwood (2017a) training set (from scratch or pretrained on the COCO data set) outperformed YOLOv3.

**Table 6** Model performance ( $R^2$ , root mean square error (RMSE) of prediction and bias) for fruit counts using the overall test set (Test set 1 All) and subsets of low, medium and high frequency of fruit occlusion ( $n = 100$  tiles in each set)

Model	Test set 1 All			Test set 1 Low			Test set 1 Medium			Test set 1 High		
	$R^2$	RMSE	Bias	$R^2$	RMSE	Bias	$R^2$	RMSE	Bias	$R^2$	RMSE	Bias
Faster R-CNN(VGG)-original	0.97	2.25	- 1.24	0.96	0.68	- 0.14	0.92	1.67	- 1.02	0.90	3.46	- 2.6
Faster R-CNN(VGG)-512	<b>0.98</b>	1.22	- 0.43	<b>0.98</b>	<b>0.44</b>	0.01	<b>0.96</b>	0.97	- 0.33	0.95	1.83	- 1
Faster R-CNN(ZF)-original	0.95	2.19	- 0.97	0.92	0.85	0.05	0.87	1.75	- 0.74	0.89	3.25	- 2.2
Faster R-CNN(ZF)-512	0.96	2.06	- 0.99	0.95	0.68	- 0.10	0.94	1.28	- 0.63	0.90	3.25	- 2.2
YOLOv2(tiny)-original	0.95	1.66	<b>0.09</b>	0.92	0.90	0.31	0.92	1.38	0.43	0.89	2.35	- 0.5
YOLOv2(tiny)-512	0.97	1.46	0.49	0.91	1.00	0.45	0.93	1.35	0.55	0.93	1.89	0.46
YOLOv2-original	0.96	1.85	- 0.90	0.94	0.76	- 0.02	0.93	1.28	- 0.63	0.93	2.83	- 2.1
YOLOv2-512	0.97	1.33	0.13	0.95	0.70	0.27	0.92	1.21	0.05	0.93	1.82	<b>0.08</b>
YOLOv3-512	<b>0.98</b>	1.18	- 0.18	0.96	0.60	0.02	0.95	0.95	- 0.18	0.95	1.72	- 0.4
SSD-300	<b>0.98</b>	1.31	- 0.52	<b>0.98</b>	<b>0.44</b>	- <b>0.01</b>	0.94	1.19	- 0.47	0.95	1.88	- 1.1
SSD-512	<b>0.98</b>	1.48	- 0.65	0.96	0.60	0.02	0.95	1.26	- 0.62	0.95	2.14	- 1.4
<i>Mango</i> YOLO(s)-512	<b>0.98</b>	1.09	- 0.39	<b>0.98</b>	<b>0.44</b>	- 0.05	0.95	1.04	- 0.39	0.96	1.51	- 0.7
<i>Mango</i> YOLO(pt)-512	<b>0.98</b>	<b>0.99</b>	0.16	0.96	0.64	0.19	<b>0.96</b>	<b>0.81</b>	<b>0.03</b>	<b>0.97</b>	<b>1.37</b>	0.25

Units for RMSE and bias are fruit number per tile. Best result within a column is indicated in bold. Input resolution was 416×416 pixels for YOLO original variants. Images were rescaled for the shorter side to be 600 pixels for Faster R-CNN original variants



**Fig. 6** Example of fruit detection on images of same tree (Test set 2A) for different cameras (Basler, Canon and Kinect), using a *MangoYOLO(s)* model trained on Train set 1

### Orchard and cultivar

Models trained using a set of images from one orchard and one cultivar (Calypso) only (i.e., Train set 1) were employed in evaluation of fruit load of orchards varying in location, growing condition and cultivar. Despite variations in fruit shape and leaf and fruit colour between orchards and cultivars, all models performed well when applied to images of other orchards, without further training or fine tuning (Fig. 7, Table 9). The best performance metrics (highest  $R^2$  and lowest RMSE) on the relationship between machine vision counts of each sides of a tree and human count was achieved using the *MangoYOLO(pt)* model in four orchards and using the *MangoYOLO(s)* model in one orchard, relative to Faster R-CNN(VGG) and YOLOv3 models (Table 9). The poorest result for all models was associated with orchard B which contained trees with larger canopies than the other orchards.

### Orchard fruit load estimation

A correction factor for the occluded fruit on the trees was calculated as the average ratio of ground truth human fruit count per tree image to the harvest count per tree. This value was estimated from the data of the sample trees in each orchard, following the approach of Anderson et al. (2018). The correction factors and their standard deviations were  $1.69 \pm 0.5$ ,  $1.63 \pm 0.46$ ,  $2.43 \pm 1.21$ ,  $1.05 \pm 0.17$  and  $1.32 \pm 0.46$  for orchards A, B, C, D and E respectively. The total number of fruit detected in the images for an orchard was multiplied by the common correction factor to estimate the fruit counts (fruit load) per orchard (Table 10). The percentage error on the orchard estimates combined the error of the correction factor estimate and the machine vision count estimate, varying between 8 and 24% across orchards and models.

The machine vision estimates based on *MangoYOLO(s)* and Faster R-CNN(VGG) were closest to the packhouse counts in two and three orchards, respectively. The higher error for the orchard D estimate was associated with small trees in which the same fruit could be seen from both sides of the tree.

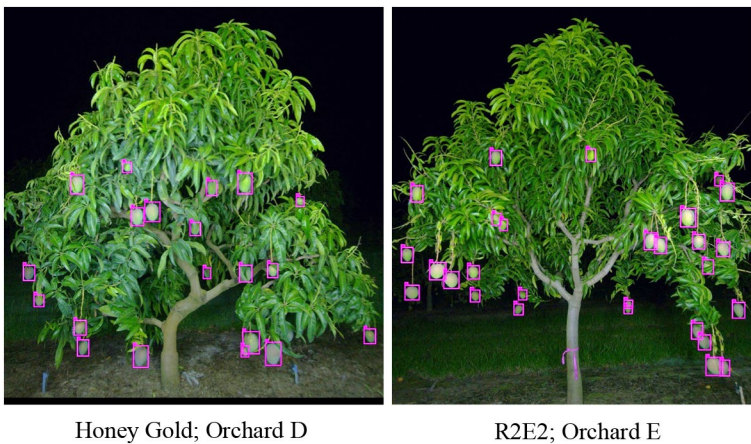
**Table 7** Regression statistics for fruit detection by *MangoYOLO(s)* and *MangoYOLO(pt)* on sets Test set 2 A-can and Test set 2 A-kin captured using Basler, Canon and Kinect cameras respectively

Test sets	Original image resolution	Network input resolution	Model	Ground truth fruit #	Total detected fruit #	R <sup>2</sup>	RMSE	Bias	False positive rate %
Test set 2 A	2 448 × 2 048	2 048 × 2 048	<i>MangoYOLO(s)</i>	2 163	2 103	0.996	2.24	1.77	0.00
Test set 2 A	2 448 × 2 048	2 048 × 2 048	<i>MangoYOLO(pt)</i>	2 163	2 201	0.987	3.25	− 1.12	0.40
Test set 2 A-can	6 000 × 4 000	2 048 × 2 048	<i>MangoYOLO(s)</i>	2 137	2 014	0.987	4.40	3.62	0.44
Test set 2 A-can	6 000 × 4 000	2 048 × 2 048	<i>MangoYOLO(pt)</i>	2 137	2 455	0.964	10.90	− 9.35	6.31
Test set 2 A-kin	1 920 × 1 080	1024 × 1 024	<i>MangoYOLO(s)</i>	1 746	1 512	0.981	7.39	6.88	0.00
Test set 2 A-kin	1 920 × 1 080	1 024 × 1 024	<i>MangoYOLO(pt)</i>	1 746	1 560	0.965	6.52	5.47	0.32

**Table 8** Prediction results for models trained and tested on the image sets used in (Bargoti and Underwood 2017a)

Train Set	Test set	Model	AP	F1	R <sup>2</sup>	RMSE	Bias
Train set 1	Test set 1-Overall	<i>MangoYOLO(s)</i>	0.991	0.959	0.98	1.09	− 0.39
Train set 1	Test set 1-Overall	<i>MangoYOLO(pt)</i>	0.989	0.956	0.98	0.99	0.16
Train set 2-bu	Test set 3-bu	<i>MangoYOLO(sbu)</i>	0.927	0.890	0.93	1.12	0.17
Train set 2-bu	Test set 3-bu	<i>MangoYOLO(ptbu)</i>	0.920	0.893	0.91	1.33	0.42
Train set 2-bu	Test set 3-bu	YOLOv3(sbu)	0.889	0.875	0.92	0.18	0.01
Train set 1	Test set 3-bu	<i>MangoYOLO(s)</i>	0.420	0.482	0.36	3.96	2.46
Train set 1	Test set 3-bu	<i>MangoYOLO(pt)</i>	0.463	0.535	0.39	0.39	2.38
Train set 2-bu	Test set 1-Overall	<i>MangoYOLO(sbu)</i>	0.834	0.868	0.86	4.60	3.13
Train set 2-bu	Test set 1-Overall	<i>MangoYOLO(ptbu)</i>	0.931	0.913	0.95	2.58	1.65
Train set 2-bu	Test set 3-bu	Faster R-CNN (VGG)	–	0.908	–	–	–
Train set 2-bu	Test set 3-bu	Faster R-CNN (ZF)	–	0.876	–	–	–

Results for the Faster R-CNN(ZF) and Faster R-CNN(VGG) models are from Bargoti and Underwood (2017a). YOLO models sbu, and ptbu refer to models s and pt trained on bu (Bargoti and Underwood 2017a) dataset. Network resolution for YOLO variants was set to 512×512 pixels for training and testing

**Fig. 7** Example of fruit detection on images of cultivars Honey Gold and R2E2, using a *MangoYOLO(s)*-512 model trained on cultivar Calypso images (orchard A)

## Discussion

### Architecture

In the current study, features of two YOLO variants (YOLOv3 and YOLOv2(tiny)) were combined to create a new architecture (*MangoYOLO*). This architecture provided a processing speed appropriate to real time operations, with use of only 33 layers for *MangoYOLO* compared to 107 layers in YOLOv3, while the accuracy was on par to (or better than) other detectors for the task of mango fruit detection in orchard.

**Table 9** Regression statistics of machine vision count against human count of fruit on images of sample trees (Table 1) for five orchards

Orchard	Mean fruit/ tree image	Faster R-CNN(VGG)		YOLOv3		<i>MangoYOLO(s)</i>		<i>MangoYOLO(pt)</i>	
		R <sup>2</sup>	RMSE	R <sup>2</sup>	RMSE	R <sup>2</sup>	RMSE	R <sup>2</sup>	RMSE
A	63.3	0.966	6.88	0.980	3.48	0.984	4.09	<b>0.988</b>	<b>2.52</b>
B	80.3	0.955	14.53	0.957	10.33	0.951	11.79	<b>0.964</b>	<b>8.71</b>
C	35.1	0.981	6.76	0.983	5.05	0.981	5.90	<b>0.990</b>	<b>3.20</b>
D	34.1	0.968	6.61	0.988	4.77	<b>0.988</b>	4.91	0.977	<b>3.91</b>
E	15.3	0.940	2.93	0.873	2.42	0.931	2.74	<b>0.946</b>	<b>2.23</b>

Best result within a row is indicated in bold. Network resolution for Faster R-CNN and *MangoYOLO* models was set to 2 048×2 048 but for YOLOv3 it was 1 888×1 888 (maximum possible within available GPU memory of 16 Gb)

**Table 10** Pack-house fruit count and number of fruits detected for each orchard using dual view imaging 6 weeks before harvest, for each of five orchards

Orchard		A	B	C	D	E
Packhouse fruit #		97 382	26 273	40 837	36 490	2 110
<i>Machine vision model</i>						
Faster R-CNN(VGG)	Fruit # estimate	93 694	25 162	41 337	40 151	2 343
	% Error estimate	13	21	24	20	21
	% Difference to packhouse count	− 4	− 4	1	10	11
YOLOv3	Fruit # estimate	100 098	27 102	43 379	41 689	2 409
	% Error estimate	9	17	20	14	18
	% Difference to packhouse count	3	3	6	14	14
<i>MangoYOLO(s)</i>	Fruit # estimate	98 029	26 453	42 179	41 146	2 383
	% Error estimate	10	19	22	15	20
	% Difference to packhouse count	1	1	3	13	13
<i>MangoYOLO(pt)</i>	Fruit # estimate	101 857	27 742	44 501	42 020	2 400
	% Error estimate	8	16	17	12	17
	% Difference to packhouse count	5	6	9	15	14

Best result within a row is indicated in bold. Network resolution was set to 2048×2048 for Faster R-CNN and *MangoYOLO* models, and to 1 888 × 1 888 (maximum possible within available GPU memory of 16 Gb) for YOLOv3

The convolutional and pooling layers used in YOLO gradually decrease the spatial dimension with increasing depth of the network, such that detection of some objects (e.g., small and non-obvious or dark fruits) may become difficult at lower resolutions. Predicting layers were implemented at three different resolutions of the feature maps, as in YOLOv3, concatenating meaningful semantic information from the deeper layers with fine grained information from the earlier layers for better detection of objects. The first detection layer was implemented at layer 42 in *MangoYOLO*, compared to layer 82 in YOLOv3, in an attempt to capture information from earlier layers of the network. The superior performance

of *MangoYOLO* compared to YOLOv3 on the mango test set indicates that this design was successful. This result demonstrates that use of very deep networks may not yield better results, depending on the context of deployment, and is consistent with advice that architecture should be customised to the application.

There is a chemometric adage that there should be ‘no prediction without interpretation and no interpretation without prediction’ (e.g., Herold et al. 2009). However, it is difficult to interpret the multidimensional learning capacity of the deep-learning models. The output of convolution filters can be visualized for interpretation of the features learned and saliency/heat maps can be generated to determine the parts of the images that were more important to the model for object detection and classification tasks (Zeiler and Fergus 2014), however in the current application this merely indicates that regions of the image associated with the whole fruit were utilised in the model. As the deep learning models performed well (high F1 score and AP) in cross cultivar generalization and detection of green fruit against a green background it can be inferred that models weighted object edge and texture features rather than object colours. An infrequent error was associated with fruit covered by foliage on all side of its perimeter, thus lacking defined edges. This error is also consistent with the interpretation that the deep learning models weighted edge or shape features of fruit.

## Training

### Pre-training

*Mango-YOLO(s)* was trained on the mango orchard training set only, while *MangoYOLO(pt)* involved pre-training on a COCO dataset for 120 K iterations before training on the mango training set. The transfer learning of the features learned on the larger dataset was of minor value as the validation results of *MangoYOLO(pt)* was similar to *MangoYOLO(s)*.

Pre-training *MangoYOLO* on the COCO dataset (*MangoYOLO(pt)*) did not significantly improve performance compared to a model that was not initialized with COCO trained weights (*MangoYOLO(s)*). Possibly the transfer learning was not helpful as the COCO dataset does not contain images of mango fruit. Perhaps a more likely explanation is that the detection task is relatively simple, and the shallower CNN architecture and heavy data augmentation techniques used in *MangoYOLO* allow for training of a robust model using a modest number of images.

### Number of training images

The training image set should encompass the variation expected in prediction images. With data augmentation (hue, saturation, jitter and multiscale), the performance (AP) of *MangoYOLO(s)* models plateaued with use of > 400 tiles in the training set. The use of at least 1 000 images is recommended to provide some safety margin. *MangoYOLO* models trained with the full training set of 1 300 tiles were robust in prediction of images from other orchards and from other cameras.

In contrast, Bargoti and Underwood (2017a) reported that for images acquired of the same orchard (orchard A) in a previous year, a performance asymptote was not reached for a Faster R-CNN within the available set of 1 154 training images. The difference between the two studies may result from the differences in lighting conditions. Night imaging results



in suppression of background issues and often highlights fruit through specular reflection from the curved fruit surface (Payne et al. 2014). Presumably night imaging allows for the quicker convergence, using a lower number of images, with consequent saving in the labour cost for labelling.

The standard deviation (SD) on AP scores (Fig. 5) for three repeated training exercises using a randomly selected 100 tiles (SD=0.0053) was only slightly higher than that for training with 600 tiles (SD=0.0041), indicative of a low level of variance in the night imaged training set. Indeed, an AP score of 0.925 was achieved for the validation set with use of just 10 training images. This result is ascribed to the heavy data augmentation used by YOLO.

## Model performance

### Comparison to previous reports

Fruit and leaf colour, shape and texture can vary with cultivar and growing condition/stage. The prediction results of handcrafted algorithms based on colour, texture and shape reported by Payne et al. (2014) and Qureshi et al. (2017) for mango fruit detection were poor relative to the results of the deep learning architectures reported in the current study, when used with images or canopies from new orchards and cultivars. For example, Qureshi et al. (2017) reported a RMSE of 11.0 fruit/tree for fruit load on prediction set images, in comparison to RMSE values <8.7 fruit/image in the current study (Table 9). The result confirms previous observations of the applicability of deep learning architectures for the fruit detection task.

### Performance characterisation

Application of deep learning models directly on higher resolution images required larger memory allocation, with 2 048×2 048 images requiring more than 16 GB of memory for use of YOLOv3-512 (Table 5). If GPU memory becomes a bottleneck for detection on large images, the tiling approach used by Bargoti and Underwood (2017a) can be adopted, wherein detection is done by sliding smaller windows over the image and finally NMS is applied collectively on the detections. However, this route is not required with the *MangoYOLO* model, given its low memory requirement.

Given data augmentation, all of the ‘off-the-shelf’ deep learning architecture performed well in fruit detection, with *MangoYOLO* achieving the highest F1 score and YOLOv2(tiny) the greatest speed. The detection accuracy results (F1 scores) of the current study were slightly higher than those reported by Bargoti and Underwood (2017a) who used Faster R-CNN(VGG/ZF) with daytime images of orchard A of the current study. For the same network resolution, Faster R-CNN(VGG) out-performed Faster R-CNN(ZF) (Table 4), consistent with the report of Bargoti and Underwood (2017a).

The *MangoYOLO(s)* and (*pt*) models were equally robust (i.e., similar AP and F1 scores) in prediction of fruit load per image across diverse datasets (orchard, camera), although *MangoYOLO(pt)* achieved the lowest RMSE of all models tested for fruit detection across different orchards (Table 7). *MangoYOLO* models trained on the Bargoti and Underwood (2017a) daytime images achieved similar performance (F1 scores) compared to the results reported by Bargoti and Underwood (2017a) using a Faster R-CNN framework with VGGNet

and ZFNet (Table 8). Of practical significance, the *MangoYOLO* model trained on daytime ProSilica camera images performed well in prediction of night Basler camera images.

Bargoti and Underwood (2017a) reported that clustered fruits represented the greatest detection error source in the mango fruit-on-tree detection task. In the current study, the bounding box approach was adequate for detection of fruits in clusters. Therefore the labour intensive method of instance segmentation for data labelling was not required. False detections on the Basler images were associated with curved leaves with contour similar to the fruit and some over-exposed areas of the branches and tree trunk. There were very few cases where detections for two fruit were merged into one when one fruit was heavily occluded by the other fruit.

The false detection rate was high (at 6.3%) for *MangoYOLO(pt)* when applied to images from the Canon camera (Table 7). The false detection mostly occurred from resizing of Canon images, resulting in image distortion, with leaves taking on a curved shape similar to fruit. Compared to the number of fruit detections on the Basler images, there were fewer detections for Kinect images presumably due to the lower luminosity (under-exposure) of those images, and more detections on Canon images (higher luminosity). There was also more false detections on Canon images, resulting from over exposed regions on branches, trunks and leaves.

### Tree and orchard fruit load estimation

The deep learning methods performed well in detection of fruit in images. However, dual view images may not reveal all fruit on a tree with a denser canopy, or may result in double counting of fruits if the same fruit is visible on both images of the same tree. An attempt was made to make an estimation of the proportion of hidden fruits as a correction factor for yield prediction, based on image and total counts of the number of fruits on the sample trees of Table 1. *MangoYOLO(pt)* yield estimations between 4.6 and 15.2% of packhouse counts were achieved, however errors on these measurements were estimated to be large (Table 10). The application of a single correction factor based on a few sample trees across an entire orchard constitutes a source of error for orchard yield estimation as the correction factor may vary with tree canopy density and fruit distribution within the tree canopy, even within an orchard of consistent management practice, as evident in the error term of this factor.

Future studies should consider system features to improve the estimate of fruit per tree. Image masking based on the depth, e.g., using a Microsoft Kinect- time of flight sensor (Wang et al. 2017), could be used to avoid double fruit counting by limiting the count from a given image to the near side of the imaged canopy. Stein et al. (2016) addressed the issue of hidden fruit in dual view imaging by using a multi-view approach, involving spatial localisation of fruit from approximately 25 images per tree side, given input of inertial navigation system and geolocation data. Ideally a tracking algorithm would be used without input of inertial navigation system and geolocation data, to minimize operational complexity and cost.

### Conclusion

This work should encourage future researches to customise deep learning models for a given application task. The multiple image sets of fruit on canopy of the current study have been made available, for use by other researchers for benchmarking performance of new algorithms.

A deep learning architecture, *MangoYOLO*, was constructed, based on YOLOv3 and YOLOv2(tiny). A F1 score of 0.97 for fruit detection in images was achieved, which, combined with a correction factor for hidden fruits, gave orchard yield estimates within 15% of packhouse tallies. With detection of fruit in the image now possible in real-time, the limiting factor for accurate prediction of the block yield using dual view imagery of trees is canopy occlusion of fruit.

Pre-training of the *MangoYOLO* model on larger datasets like ImageNet, COCO and PascalVOC is recommended but not essential, given training with data augmentation on around 1 000 tiles of fruit on tree. The architecture is recommended for the task of estimation of mango fruit in tree images in comparison to YOLOv3, YOLOv2(tiny), Faster R-CNN(VGG) and Faster R-CNN(ZF) in terms of memory use, speed and accuracy. This model can be used with images acquired from a farm vehicle operated at about 6 km/hr for real time fruit detection, and can be deployed for different image resolutions without need for re-training. The model should be tested with other fruit types and other imaging conditions. Increasing image resolution and use of higher illumination levels may further improve the fruit detection rate, and further training of the models with images from other cultivars could improve performance in use with those cultivars.

**Acknowledgement** This work received funding support from the Australian Federal Department of Agriculture and Water, and from Horticulture Innovation Australia (Project ST15005, Multiscale monitoring of tropical fruit production). AK acknowledges receipt of an Australian Regional Universities Network scholarship, and ZW was funded by a CQU Early Career Fellowship. Farm support from Chad Simpson and Ivan Philpott is appreciated. The assistance of Jason Bell of the CQUniversity High Performance Computing cluster is acknowledged. The work also benefitted from discussion with AlexeyAB (<https://github.com/AlexeyAB>).

## References


- Anderson, N., Underwood, J., Rahman, M., Robson, A., & Walsh, K. (2018). Estimation of fruit load in mango orchards: tree sampling considerations and use of machine vision and satellite imagery. *Precision Agric.* <https://doi.org/10.1007/s11119-018-9614-1>.
- 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 Robot*, 34, 1039–1060. <https://doi.org/10.1002/rob.21699>.
- Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L (2009) Imagenet: a large-scale hierarchical image database. In: Proceedings—IEEE 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 Vis*, 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>.
- He K, Gkioxari G, Dollár 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 (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition, pp 770–778

- Herold, B., Kawano, S., Sumpf, B., Tillmann, P., & Walsh, K. B. (2009). Chapter 3. VIS/NIR spectroscopy. In M. Zude (Ed.), *Optical monitoring of fresh and processed agricultural crops* (pp. 141–249). Boca Raton, USA: CRC Press.
- Hung, C., Underwood, J., Nieto, J., & Sukkarieh, S. (2015). A feature learning based approach for automated fruit yield estimation. In A. Zelinsky (Ed.), *Field and service robotics* (pp. 485–498). Cham: Springer. [https://doi.org/10.1007/978-3-319-07488-7\\_33](https://doi.org/10.1007/978-3-319-07488-7_33).
- Jimenez, A., Ceres, R., & Pons, J. (2000). A survey of computer vision methods for locating fruit on trees. *Trans ASAE*, 43, 1911–1920. <https://doi.org/10.13031/2013.3096>.
- Kadir, M. F. A., Yusri, N. A. N., Rizon, M., Bin Mamat, A. R., Jamal, A. A., & Makhtar, M. (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>.
- Lin T-Y, Dollár P, Girshick R, He K, Hariharan B, Belongie S (2017a) Feature pyramid networks for object detection. In: IEEE conference on computer vision and pattern recognition, pp 936–944. <https://doi.org/10.1109/cvpr.2017.106>
- Lin T-Y et al (2014) Microsoft coco: common objects in context. In: European conference on computer vision. Springer, pp 740–755
- Lin TY, Goyal P, Girshick R, He K, Dollar P (2017b) 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 AC (2016) Ssd: single shot multi box detector. In: European conference on computer vision. Springer, pp 21–37. [https://doi.org/10.1007/978-3-319-46448-0\\_2](https://doi.org/10.1007/978-3-319-46448-0_2)
- Nanaa K, Rizon M, Rahman MNA, Ibrahim Y, Aziz AZA (2014) 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>
- Payne, A., & Walsh, K. (2014). Chapter 16. Machine vision in estimation of fruit crop yield. In Y. Ibaraki & S. D. Gupta (Eds.), *Plant image analysis: fundamentals and applications* (pp. 329–374). Boca Raton, FL, USA: CRC Press.
- Payne, A., Walsh, K., Subedi, P., & Jarvis, D. (2014). Estimating mango crop yield using image analysis using fruit at ‘stone hardening’ stage and night time imaging. *Comput Electron Agric*, 100, 160–167. <https://doi.org/10.1016/j.compag.2013.11.011>.
- 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>.
- Qureshi, W. S., Payne, A., Walsh, K. B., Linker, R., Cohen, O., & Dailey, M. N. (2017). Machine vision for counting fruit on mango tree canopies. *Precis Agric*, 18, 224–244. <https://doi.org/10.1007/s11119-016-9458-5>.
- Redmon J (2018) Darknet: open source neural networks in C. <https://pjreddie.com/darknet/>. Accessed 23/03/2018
- 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 [arXiv:180402767](https://arxiv.org/abs/1804.02767)
- 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 1–99
- Sa, I., Ge, Z., Dayoub, F., Upcroft, B., Perez, T., & McCool, C. (2016). Deep fruits: a fruit detection system using deep neural networks. *Sensors*. <https://doi.org/10.3390/s16081222>.
- Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint [arXiv:14091556](https://arxiv.org/abs/1409.1556)
- Springenberg JT, Dosovitskiy A, Brox T, Riedmiller M (2014) Striving for simplicity: the all convolutional net. arXiv preprint [arXiv:14126806](https://arxiv.org/abs/1412.6806)
- Stein, M., Bargoti, S., & Underwood, J. (2016). Image based mango fruit detection, localisation and yield estimation using multiple view geometry. *Sensors*. <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.

- Underwood JP, Rahman MM, Robson A, Walsh KB, Koirala A, Wang Z (2018) Fruit load estimation in mango orchards—a method comparison. Paper presented at the ICRA 2018 workshop on robotic vision and action in agriculture, Brisbane, Australia
- Walsh, K., & Wang, Z. (2018). Monitoring fruit quality and quantity in mangoes. In V. Galán Saúco & P. Lu (Eds.), *Achieving sustainable cultivation of mangoes* (pp. 313–338). Cambridge, UK: Burleigh Dodds Science Publishing.
- Wang, Z., Underwood, J., & Walsh, K. B. (2018). Machine vision assessment of mango orchard flowering. *Comput Electron Agric*, 151, 501–511. <https://doi.org/10.1016/j.compag.2018.06.040>.
- Wang, Z., Walsh, K. B., & Verma, B. (2017). On-tree mango fruit size estimation using RGB-D images. *Sensors*. <https://doi.org/10.3390/s17122738>.
- Zeiler, M. D. (2014). Visualizing and understanding convolutional networks. *LNCS*. [https://doi.org/10.1007/978-3-319-10590-1\\_53](https://doi.org/10.1007/978-3-319-10590-1_53).

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

## Affiliations

A. Koirala<sup>1</sup>  · K. B. Walsh<sup>1</sup> · Z. Wang<sup>1</sup> · C. McCarthy<sup>2</sup>

K. B. Walsh  
k.walsh@cqu.edu.au

Z. Wang  
z.wang@cqu.edu.au

C. McCarthy  
cheryl.mccarthy@usq.edu.au

<sup>1</sup> Institute for Future Farming Systems, Central Queensland University, Building 361, Bruce Highway, Rockhampton, QLD 4701, Australia

<sup>2</sup> Centre for Agricultural Engineering (Operations), University of Southern Queensland, Building P9-132, West Street, Toowoomba, QLD 4350, Australia