Implementation and Comparison of Machine Vision Algorithms for Detecting Apples at an Orchard

Frank S, Heath F, Nair RRP, Rolland E, Rowland T, Sharma M
(Dated: January 10, 2023)

ABSTRACT

Calculating and predetermining the yield of crops is vital for estimating the seasonal turnover in the agricultural industry. Machine vision algorithms are used to make this process more efficient and accurate. This report is a comprehensive analytical investigation for implementing and comparing machine vision algorithms in the detection of apples at an orchard. Four different computer vision approaches were used for the detection: two conventional methods, which entailed a direct manipulation of the images input through the algorithms, and two machine learning methods, whereby a convolutional neural network is trained to recognise patterns in the input data. The dataset applied to both methods consisted solely of red apples on trees. The results of our investigation illustrated that there is a significant difference in accuracy between conventional and machine machine learning based methods. Utilising the YOLO object detection model proved to be the most accurate having only overestimated the apples in the dataset by 2.6%. For comparison the best conventional approach counted 65.1% of the number of apples.

CONTENTS

I. INTRODUCTION

The efficiency and accuracy of the statistics within the agricultural industry are deemed volatile as they are directly affected by the population of the data, as well as the control variables in place of the environments [9]. To establish a solid set of results, computer vision has been incorporated increasingly within this industry to improve yield estimation, monitoring, disease prevention and detection [10]. The purpose of this report is to implement and compare machine vision algorithms, to find the most effective approach for the detection of apples at an orchard.

To ensure that the models tested produce reliable statistics, a large dataset was acquired and sorted into types of images with their corresponding labels. For this investigation, only red apples were examined and accounted for in the detection algorithms. The initial approach was a conventional method, whereby the images from the dataset were manipulated by their colour properties before the application of morphological operations. The second approach is applying machine learning algorithms within computer vision. This process consists of a convolutional neural network trained to recognise and detect set patterns provided by the input dataset. Two separate approaches are used, YOLO and Mask-RCNN.

The assumptions within this report are focused around the environment involving the dataset. Specifically features such as the lighting where the image has been captured and the localised background. It becomes increasingly difficult to train conventional algorithms as certain patterns are not perceivable due to changes between the dataset images. In order to successfully determine the best approach for detecting the apples at an orchard, specific aims and objects were set as checkpoints. The aim was to find the method which has the highest level of accuracy at detecting and counting red apples on tree, where accuracy is measured against the corresponding image labels. The objectives are listed as the following:

Objectives:

- Sort and select the images and labels from the dataset.
- Complete data acquisition to know what features must be taught/manipulated for the approaches.
- Investigate elements in images and methods to manipulate to apply to the conventional algorithm.
- Tune conventional methods for optimal performance across the dataset.
- Create code for machine learning algorithms and train to detect patterns in images.
- Test and tune machine learning algorithms for optimal performance across the dataset.
- Compare conventional and machine learning algorithms with the same dataset to discern the best approach.

Successfully completing these objectives required overcoming challenges such as establishing a suitable green filter threshold and deciding the best order of morphological operations as well as ensuring the dataset was fit for each machine learning model.

II. RELATED WORKS

Fruit counting by detection has been extensively researched in the following section which provides a comprehensive discussion of classification schemes used to sort fruits. The limitations of present methods used are discussed.

Segmentation and localisation of fruit detection for robotic harvesting has gained significant attention from the research community. Older approaches used colour thresholds that required manual setup. This required manual extraction of colour, shape and texture features, followed by classification and clustering techniques, until deep learning techniques became more prevalent [10]. The various approaches used differ in terms of the sensing techniques used - Bargoti et al.(2017), Chen et al.(2017) used RGB [1],[5], Stein et al. (2016) used LIDAR [22], Wu, D et al.(2020) used YOLO [26] and Si et al.(2015) used stereoscopic vision to detect fruits [21].

To classify the pixels of the apple images from the background, back propagation was used by Shahin et al.(2016) in [20]. After employing segmentation, edge detection was used to sort out individual apple samples, after which they were summed to yield total count result. Over 89.5 % of the fruits were successfully identified from 160 testing photos by Si et al.(2018) in [21], a system that employs stereoscopic vision to find apples in trees. Stein et al.(2015) employs a cutting-edge multi-sensor architecture that effectively maps and identifies individual fruits in a mango orchard [22].With only a 1.36 % error rate for individual trees, this method requires little calibration.

Faster R-CNN [6] which combines both region proposal and classification phase into a single network has been adapted by Chen et al.(2018) for detecting fruits and vegetables. This method was employed by [22] to count apples, mangoes and almonds, which resulted in F1 scores greater than 90 %. Hence, this was the preferred method in this experiment.

The majority of articles reviewed thus far have aimed towards precise fruit counts. After post-processing the output from the neural networks, they acquire the counting results. While the detection portion of the processing pipeline has experienced fast advancements, culminating in the deployment of Faster R-CNN in orchards, the counting portion has been mostly overlooked.

Given the above research, this study investigates both conventional and machine learning approaches. To make it easier for the algorithm to distinguish between the surrounding elements and the apples, the dataset applied to both approaches was limited to red apples on trees.

III. DATA ACQUISITION

Computer vision applications within agriculture include disease prevention, yield estimation, ripeness estimation, weed detection and livestock monitoring which has lead to the requirement for many datasets of images for computer vision within agriculture. To make a system that can count and detect apples we require a dataset containing many images of apples. We envision however that our system for counting apples may be used to estimate the yield within an orchard. We therefore require a dataset of apples with context of their surrounding, that is, still attached to the branches of a tree.

The most suitable dataset for the task was deemed to the be MinneApple dataset [13]. The MinneApple dataset was created to provide a large dataset of high-resolution images of apples in an orchard environment. Images were taken using a standard mobile phone camera and have been annotated by a human. The objects are annotated as individual polygon masks for each apple instance. In total the dataset contains 40,000 apple instances from 1000 images. Our needs dictated that we would not use the full dataset, a large subsection within the dataset contains small bounding-box style images that may be useful for classification problems over detection and counting.

FIG. 1. An example image from the MinneApple dataset [13]

We approach the challenge of precisely counting apples by noting two observations: The apples in the dataset are (1) sporadically scattered throughout the entire image and (2) frequently grouped together in bunches.

The system analysed red apple identification and classification from a dataset of real images [13] containing complicated disturbance information; the background was similar to the surface of the apples and trees. The image contains apple clusters that are identified by segmentation method, and a counting method computes per-image counts. Clusters of apples will be apparent in multiple images. The contour images were created using pre-treatments such as noise removal and area filling. A model that identifies a set of general but distinguishing image characteristics of the apple was used to extract the features. The location of each cluster must be monitored, and counts must be avoided to prevent double counting. The feature extractor's output was mapped to labels using training data for the model. This method is prone to false positives and therefore requires the counting algorithm to reject any false positives.

In order to compare conventional methods with machine learning, the same MinneApple subset must be used against both for validation. The chosen subset is 32 images of full apple trees in the orchard, which have the number of apples labelled in a separate .txt file corresponding to each image. Results of this comparison are detailed in section VI. It should be noted that labels for these images count only those still on the tree as apples that are desirable to harvest. Hence the algorithms aim to avoid counting apples which have fallen to the floor.

The conventional algorithm requires no training data, as it manipulates each image directly to separate and count instances of apples. Hence, the only input is the subset of 32 images and their labels. The images are then run through the algorithm, the number of detected apples is counted and compared to the labelled number as a percentage of labelled apples detected.

In contrast, the machine learning approach requires a collection of images for which the neural network weights can be optimised for. These images must have associated annotations whereby the images object information can be used by the model.

The dataset had been originally used with a Mask R-CNN and a Faster R-CNN by the original curator and the annotations where incorrect for use with the YOLO model as they were in the COCO format. For the subset of the dataset that we intended to use the annotations must be converted. A python script using the Pylabel package [12] had to be employed to read the annotations from a .JSON file, create a text file for each image and write the instance dimensions in the correct format.

IV. METHODOLOGY

A. Approach 1: Conventional

Conventional approaches to machine vision entail direct manipulation of images input to the algorithm. Our method has two distinct processing stages that are applied after the image set is recorded in an array along with a corresponding labels array.

Initially, we used the thresholding methods from the tutorial and the lecture [4][3] which are the adaptive, manual and Otsu. We chose the parameters for these to have the highest possible accuracy. The best method among those was Otsu when applied after a HSV filter on the image. This was highly effective at identifying single apples, but when looking at entire trees, the apples are no longer distinguished from the background, illustrated in FIG. 2. We therefore decided to employ a tailor made thresholding method.

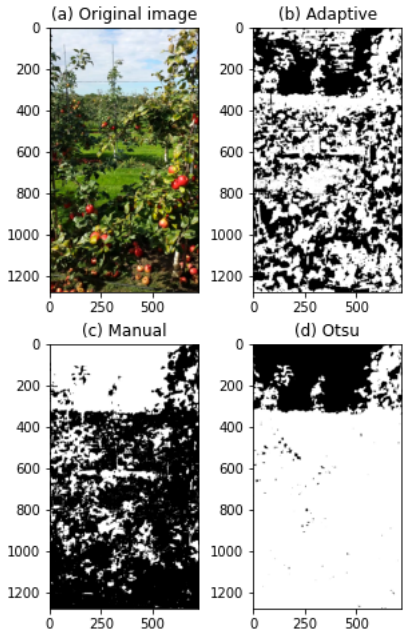


FIG. 2. Comparison between the three existing thresholding methods. Though initially promising, performance on entire trees is too poor to be used.

We made the decision to limit our scope to red apples that are on the tree only. They are more easily distinguished against the green of the tree, and the labelled dataset used for the machine learning approach considered only those, so we can compare the approaches. The first stage of our method is therefore the application of a custom colour filter

to remove “non-red” pixels. This choice and the process will be detailed in section V A. Two algorithms have been developed which differ only slightly. While both filter out ”non-red” pixels, one splits the image into ten segments before setting the threshold independently for each segment. The intention was to increase accuracy, section VI, details why the non-split algorithm was kept in use.

The colour filter converts red regions in the original image, to white on a black background. At this point, we are already able to spot in one look the apples and have a approximate idea of the amount, as shown in FIG. 6 (original) images after the threshold is applied. Binary images are much more easily processed by machines than the original complexly coloured image. Morphological operations can then be applied to great effect, adding smoothing and separating each of the detection regions.

Finally, the white detection regions remaining are labelled with circles which are counted, layered over the original image and compared to the labelled number of apples. The combined combination of visual and numerical results allows the user to discern if false-positives are causing significant inaccuracy.

B. Approach 2: Machine Learning

Machine learning within computer vision is where a convolutional neural network “learns” to recognise patterns within the input data [18]. A lot of data is required to achieve this successfully. During the training process the network gets exposed to many training examples within the dataset which contain the ground truth bounding boxes and class labels. The model makes a prediction for that input image which is compared to the ground truth with a loss function. The loss is then back-propagated through each layer in the network from the output layer through each hidden layer to the input layer [18]. All layers with the exception of the input layer (which has no weights) will have their weights updated as to minimise the loss. This process is then repeated through many iterations to converge the network to a point with minimal loss. It is important to have a large set of data to prevent overfitting - where the model gets too optimised for the limited training data, this then impedes its ability to generalise to other object examples that it has not previously seen.

Some important hyperparameters that are involved include learning rate, weight decay and batch size. The learning rate is used by the optimisation algorithm, which in the case of both models is Static Gradient Descent (SGD) and it determines the amount at which each weight can be updated in each correction. Having a faster learning rate will mean that the loss can converge quicker, but it could also lead to oscillations or divergence. Weight decay can be altered to try mitigate any overfitting, it is implemented with the loss function and penalises the CNN for having weights that are too large. Instead by having smaller weights the network can learn more generalisable patterns [2]. Batch size is commonly altered and is often dictated by the computing resources when training. It is the number of images that the network makes predictions on before there is an update to the weights. It is generally set to 32, 64 and 128 but could go anywhere from 16 - 512.

There are many different convolutional neural network architectures and adaptations. We have decided to explore the use of 2 popular architectures, YOLO and the Mask R-CNN.

1. YOLO

The YOLO architecture has become one of the most widely used object detection algorithms in recent years. The initial release with v1 was in 2015 with a paper titled “You Only Look Once: Unified, Real-Time, Object Detection” [18]. It has since been re-released with many improvements. For our application YOLOv5 was chosen, the large amount of accompanying documentation meant that it could be more readily implemented than other versions. YOLOv5 unlike previous versions was not released with an accompanying paper, instead it undergoes ongoing development through the Ultralytics GitHub page [14].

The YOLO object detection algorithm uses bounding box style annotations and produces a bounding box with a class label as an output as shown in FIG. 9. The bounding box is drawn over the desired object and the whole area within is attributed to that class label. This has some advantages over semantic segmentation annotation such as it is quicker to annotate, it will however be subject to noise from the areas within the bounding box that are not actually part of the object in question. Training could therefore require more examples as the neural network weights get optimised.

YOLOv5 has several different model sizes that can be implemented. YOLOv5s was chosen to be the most appropriate for our requirement as it trades off the model size and computing resources with accuracy. The model starts with a total of 214 layers. After training model layers get fused to 157 layers and over 7 million weights [14] to reduce the computational burden of inference [16].

2. Masked region based convolution neural networks (Mask-RCNN)

This architecture of neural networks called masked region-based convolution neural networks (Mask-RCCN) allows instantaneous identification and segmentation of objects within an image. This method makes it possible to carry out semantic segmentation where each pixel of an image is associated with a class label [11].

FIG. 3. Example of segmentation in the context of apple detection prior to training using weights trained on MS-COCO

This machine learning method is based on the R-CNN architecture. This method first extracts interesting regions from the image and then uses these regions as input data for a convolutional neural network. This separation into regions makes it possible to detect several objects of several different classes in the same image.

This architecture is then improved to become the Faster R-CNN architecture. The addition of convolution layers allows the generation of regions of interest and facilitates their classification. The region proposal network then uses sliding windows of different sizes and ratios to analyse the feature map in depth. These changes improve the accuracy and speed of the architecture compared to R-CNN.

The Mask R-CNN is the next improvement after the Faster R-CNN. This method changes the output of the final model. Indeed, the Faster R-CNN architecture allows location of distinct objects with a bounding box. The Mask R-CNN architecture allows to segment each instance of an object with a semantic mask. The Mask R-CNN architecture is similar to Faster R-CNN but adds layers in parallel to the classification. [11] [25]

Initially the Mask R-CNN has pre-trained weights from the MS-COCO dataset [7][15], we then trained the network further with the use of a tutorial [17] on the dataset of apples to optimise it for our application.

The code provided is adapted to meet the needs of our study: the dataset used presents detection masks and associated images.

V. IMPLEMENTATION & EXPERIMENT

A. Conventional

We are using Google Colab for coding, as it enables us to work simultaneously on the same file, perform cloud computing executions, using a GPU. All scripts, datasets and documents related to this project were stored in a shared Google Drive for the same reasons.

We are using matplotlib for image reading and plotting. It reads images as 3D-arrays of size $width \times height \times 3$. For each pixel, it has the RGB value as 3 floats numbers. Thus, we handle their manipulation with numpy.

The bottom 17% of each image is cropped as, across the MinneApple dataset, this is the average portion of the image taken up by the floor. With the dataset uploaded, Roboflow [19] allows the visualisation of the annotation locations . FIG. 4 provides the heat map of label placement used to discern this floor percentage value. We can see that no high density label areas are excluded, while we avoid potential detection of many apples on the floor.

The thresholding operation consists in calculating a weight value, w, that is actually our threshold. For the chosen dataset, we postulate that the green tree will always take the biggest part of the image. From this, we can design our w as in Algorithm 1. The objective was to separate the apples, that are considered as the only red parts, from the rest (see FIG. 5). Thus apples' pixels would have a low green component, which is the reason everything below the threshold is turned to white. The risk is low for the sky to be turned white, since it is generally light blue with a large enough green component, as it tends from blue to white. Any parts of the sky turned to white are successfully eliminated during the morphological operations. The exact formula for w was found empirically to give the best results with the whole detection process. In this way, all detected apples are rendered as white silhouettes on a black background.

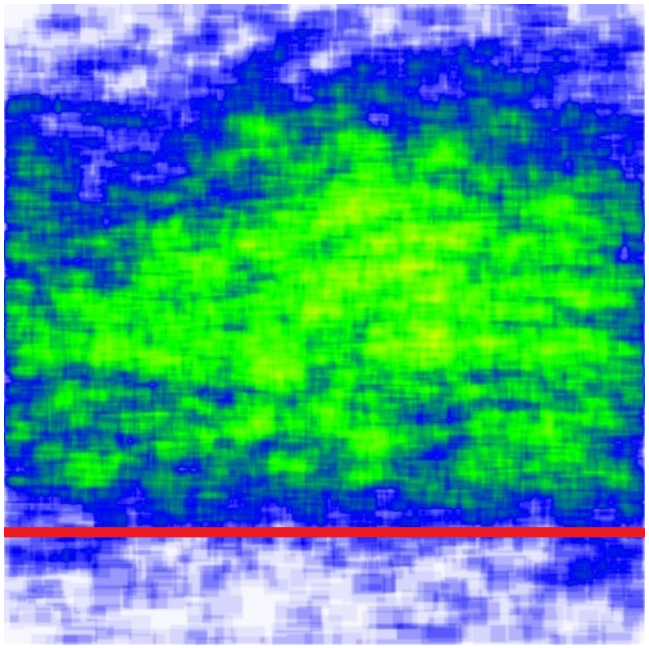


FIG. 4. A heat map of where apples are labelled on images in the MinneApple dataset. Green indicates high density, blue low density, white shows no labels. The red line marks the point at which each image is cropped (the lower 17%).

Algorithm 1 Thresholding method to separate "red" and "non-red" pixels [Pseudo-Code]

Input: *im*: image to filter and crop out the floor
Output: filtered and cropped image

*im*2 ←black image which is cropped to remove bottom 17%
weight ←average[green/(red +2*green+blue)]
#green refers to the green component of a pixel. The average is for all the pixels of *im*.

for each pixel **in** *im*
  if proportion of the green component $< w$ **then**
    the pixel in *im*2 at the same position becomes white
  end if
end for

For the split-image algorithm, the image is cropped, then split into ten segments before a threshold is calculated

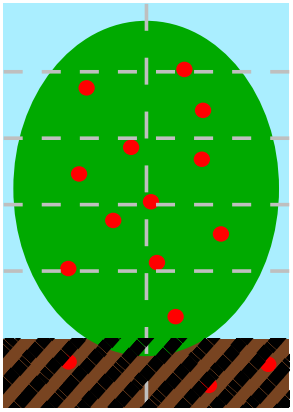


FIG. 5. Schematic of a typical image from the dataset we use. The green tree takes up most of the image, we also have the sky, the floor and the red apples. Only apples on the tree are desirable, the dashed lines represent the segmentation used in the split algorithm.

and applied to each of them. The segments are then merged back together to give a similar output as the non-split method. (Algorithm 2)

Algorithm 2 Splitting-thresholding-merging method [Pseudo-Code]

Input: *img*: image to filter
Output: filtered and cropped image

Split *img* into 10 parts

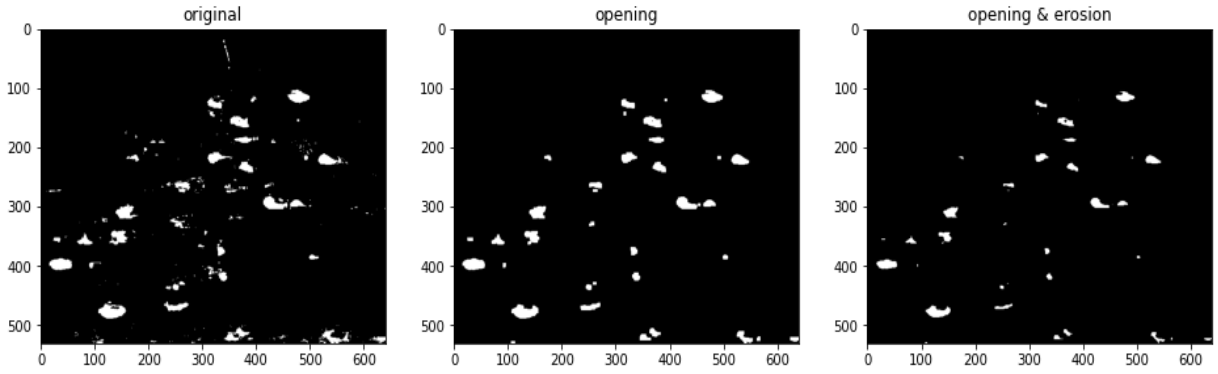
for each part **do**
  Perform Algorithm 1 excepting cropping of the *img*
end for
Merge the 10 parts #to get the full filtered and cropped version of *img*
img ← *img* without the bottom 17% #cropping the floor

Once the above has been applied, both conventional algorithms perform one round of opening, then erosion on each image. Opening allows better separation of white points of interest on a black background compared to closing [8]. Applying erosion after this better clarifies each point to minimise the number of closely bunched apples detected as a single apple. These morphological operations are displayed in FIG. 6. They have been achieved using the python cv2 package and google colab patches to adapt it.

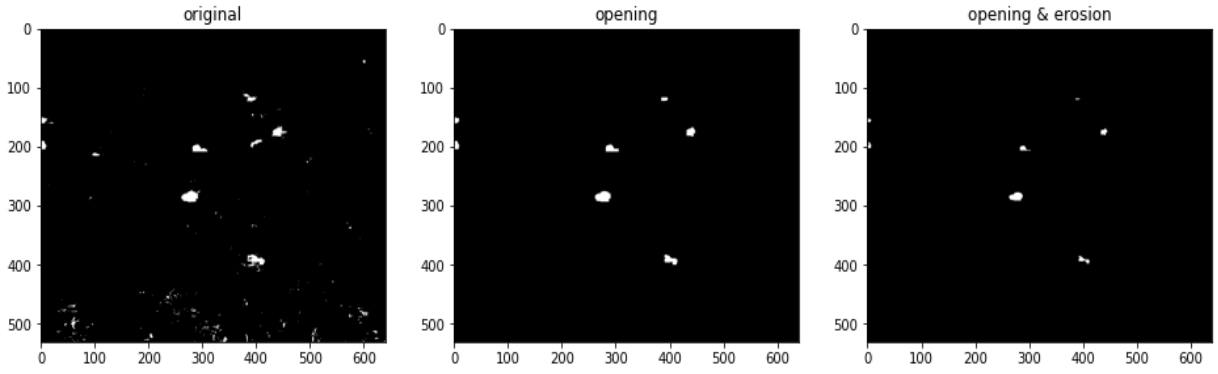
It is clear from FIG. 6 that applying a different threshold to ten segments in the image results in much harsher designation of green overall, with comparatively few apple detections remaining after morphological operations. This due to the difficulty in finding an adapted w formula here. Our previous solution worked because the tree-sky surface distribution was very similar from one image to another.

With the splitting algorithm, this distribution varies a lot from one segment to another, as shows FIG. 5. This leads the w values to be lower compared to non-segmentation, so fewer apples are detected using Algorithm 2. Due to the diversity of the segments, finding a universal w expression is not possible, yet specific thresholds for each segments - w_1 for segment 1, w_2 for segment 2,... - does not produce improved results. Even if the images appear similar, when zooming to a segment scale, they can to be very different. However, this splitting method still has advantages which are detailed in section VI.

Each of the remaining white spots are then converted to a white circle and layered over the original image to show where the algorithm has detected apples. The ndimage library from the scipy package was used here. Circles are counted and compared to the number of labels for the image to generate a detection percentage. Obviously this percentage is inaccurate if the algorithm has any false-positive detections.



(a) No Image Segmentation



(b) Image Segmentation Used

FIG. 6. Here we can see the image output from the green filter function, labelled as "original" on both (a) and (b). Opening and erosion are then applied in sequence as shown.

B. Machine Learning

1. YOLO

The GPU required for training was gained by using Google Colab. The converted dataset could be easily loaded from the Roboflow server for use within Google Colab without the need upload images to the current runtime. The dataset contained 331 images which collectively contained 12,285 apple instances for the network to train on. This was split into 75% training images, 15% validation and 10% remaining for the comparitive test. YOLO requires the input images from the dataset to be accompanied with their own annotation file, this comes in the form of a .txt file, this must be the same name as the associated image aside from the .txt suffix instead of .jpg/png. Within this text file, each row represents one object instance annotation within the image, the row starts with a class identifier and is followed with 4 numbers in the range *0 - 1*. These four numbers represent the centre x and y coordinates and the width and height of the box with the image dimensions normalised to 1. The image dataset and annotations must be accompanied with a .YAML file. This is a configuration file that contains the training setting and hyperparameter options for training the model and information about the dataset. The path to the training, validation and test datasets are contained within in this file as well as the class label map. As our requirement is to only train for and detect one class, nc = 1 and names = ["apple"]. The class identifier in the annotation file will correspond the the element number in the names array, hence every row is prefixed with a *0*.

Prior to training the images are resized to 640x640 pixels and training was carried out for 85 epochs with a batch size of 16. This was enough training iterations for the loss to decrease sufficiently without causing the model to overfit to the dataset. Learning with the default hyperparameters was sufficient with the learning rate = 0.01 and a weight decay of 0.0005.

Key metrics when evaluating the success of a machine learning system include, precision and recall - which can be used to calculate mean average precision (mAP), and box loss. Precision is a measure of the number of true positive predictions divided by the number of total number of predictions. This shows that the model is good at identifying an object but it wont take into account any false negatives where the model failed to identify an apple. Recall on the other hand would be measure of how many of the apples the model correctly managed to identify, but it doesn't take into account the false positives. mAP is a metric that combines precision and recall to judge the overall accuracy of the system. Box loss is measure of how well the model predicted the bounding boxes, this is then used to make corrections to the weights. As shown in FIG.7, after 85 epochs the model had a mAP0.5 of 0.814 which equated to an accuracy of 81.4% which is calculated from a precision 0.845, a recall 0.733. The model has a box loss which started at approximately 0.13 was reduced to 0.0392 indicating that the loss had converged. The graphs in FIG.7 show that further training may have diminishing return as all recall, precision, mAP and box loss had plateaued indicating no further improvements could be gained with the current dataset and algorithm.

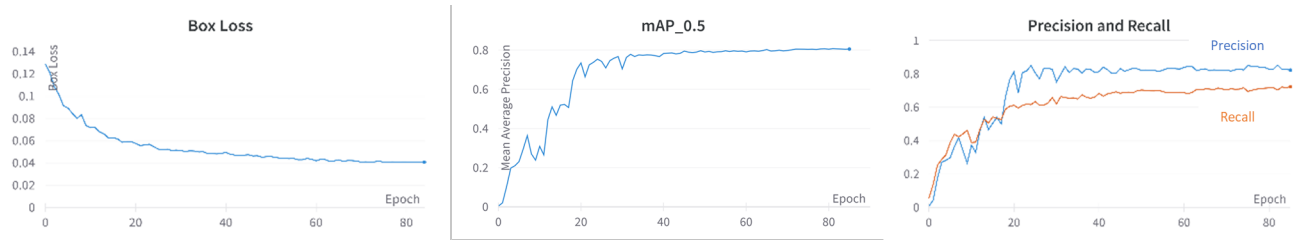


FIG. 7. Training graphs for YOLOv5 after 85 epochs. From left to right the graphs show box loss, mean average precision (mAP) and, precision and recall.

2. Mask-RCNN

The Mask R-CNN was also trained with Google Colab. To train the Mask R-CNN deep learning model, we used a different dataset than with YOLO as it requires two distinct sets of images: the apple tree images and the associated masks as presented in figure 8.

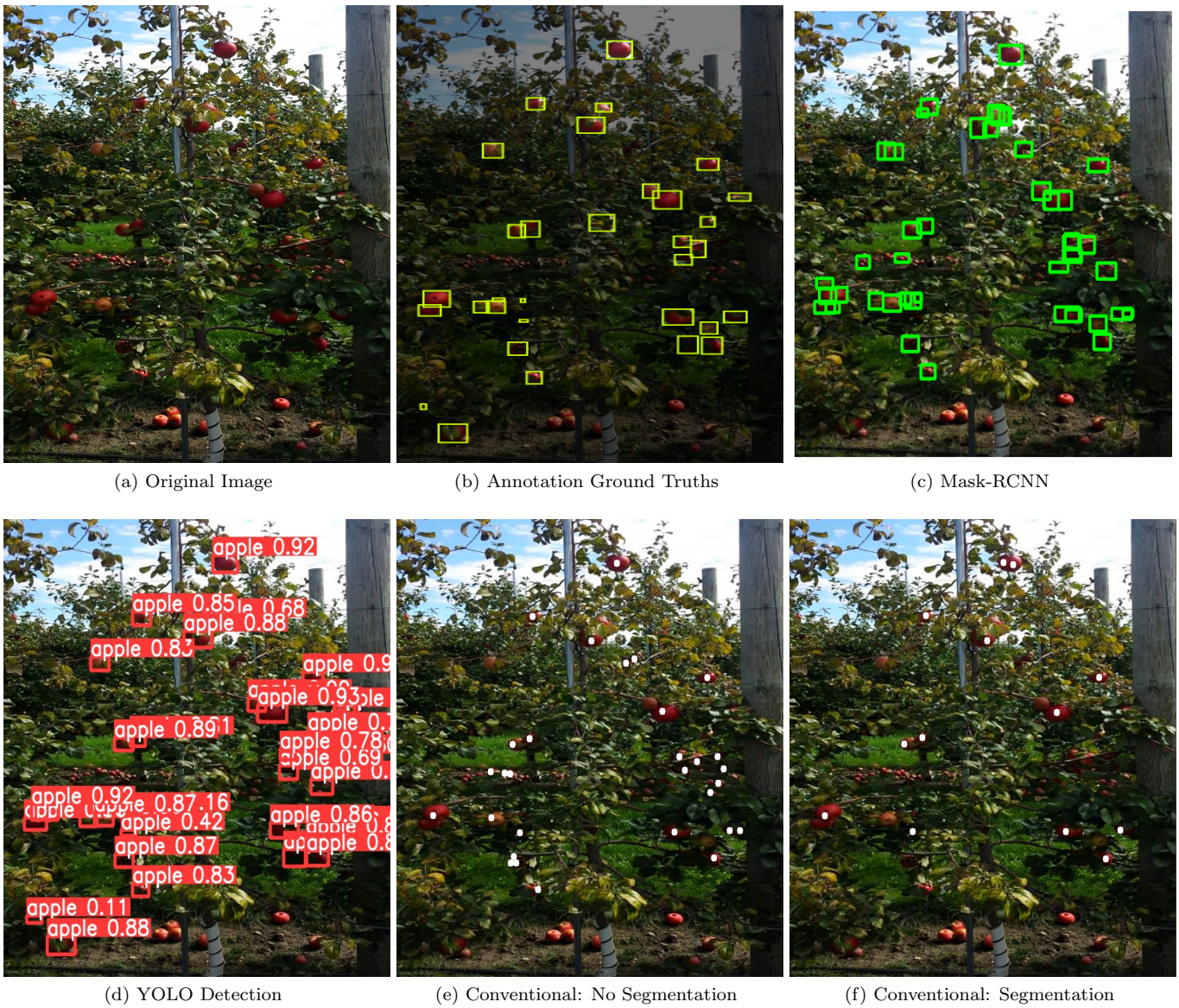
FIG. 8. Image from the dataset with the associated mask

The torchVision python package was employed to download and interact with the neural network model [24]. torchVision is a package within the Pytorch open source machine learning framework. The model, like YOLO, came pre-trained on MS-COCO [15][7] so further training aimed to optimise the weights for our purpose of apple detection and was implemented with a tutorial [17] originally written for pedestrian detection. The hyperparameters weight decay and batch size were set to 0.0005 and 16 respectively. Pytorch enables the use of some learning rate schedulers.

In this case the "stepLR" which stepped the learning rate down from a starting point of 0.005 by a factor of 0.1 every 3 epochs. This was done to prevent overfitting and improve generalisation [23].

The training carried out in this study aims at refining the provided model by taking into account the MinneApple dataset [13] which contains 670 images and its 670 associated masks. Training was carried out over 50 epochs.

VI. RESULTS & EVALUATION



(a) Original Image (b) Annotation Ground Truths (c) Mask-RCNN

(d) YOLO Detection (e) Conventional: No Segmentation (f) Conventional: Segmentation

FIG. 9. Here we see the output from each of the four algorithms with same input image. (a) shows the unprocessed image, (b) shows the 32 labels on the image. (c) and (d) display the two machine learning algorithm results, while (e) and (f) show the conventional algorithm results for non-segmentation and segmentation respectively.

Regarding the two conventional methods, one can see in FIG. 9 (e) and (f), using segmentation leads to far fewer apples detected overall; many prominent apples in the image are not detected. However the segmentation approach leads to far fewer false-positives or instances of multiple detection points on single apple. While avoiding segmentation, leads to fewer apples missed, the high number of multiple detections on single apple with this method not only results in cases where more apples are detected than are in the image, it invalidates a purely numerical output as you cannot be sure that the apples detected are separate apples before the annotated image is viewed. For example, FIG.

9 (e) contains 32 points of detection, matching the number labelled. While on the surface this looks to be a perfect success, the image shows that some apples are missed and some are detected multiple times. Distinguishing multiple bunched apples is where the machine learning methods truly outmatch our conventional algorithms.

Utilising YOLO as a CNN model to detect apples in an orchard has proven itself to be a good method. The training dataset although relatively small with the number of images contains a sizeable number of object instances which has allowed the network weights to optimise well. FIG. 9 shows that both the YOLO and Mask R-CNN have successfully identified most of the apples annotated. The bounding boxes in the image for YOLO inference (d) are accompanied by the class label and an associated confidence score that the models prediction is correct.

A comparison was made between all of the methods tested in this paper. A dataset of 32 images was created and each method used these images to make a prediction for the number of apples in the images. All of the images in this dataset were new images that both machine learning models had not been exposed to in their respective training processes.

Labelled Number of Apples	**Conventional: No Split**	**Conventional: Split**	**ML: YOLO**	**ML: Mask R-CNN**
1246	811	295	1279	1143

TABLE I. Overall comparison of the number of apples labelled across the 32 image dataset and how many were detected by each algorithm.

Table I shows that the machine learning approaches have a significantly improved accuracy when compared to conventional computer vision methods. The YOLO model performed best having only overestimated the number of apples in the dataset by 2.6%, the Mask-RCNN performed slightly worse having underestimated by 8.2%. Comparatively, both conventional methods performed poorly with the segmentation detecting only 23.7% of apples. The non-segmentation method theoretically detected 65.1% of apples but looking at the direct results per image there are multiple instances of overestimation. This implies less credence should be placed on the overall result for this method. FIG. 10 shows that the error rate per image was much higher for conventional methods, with YOLO outperforming Mask R-CNN in terms of accuracy.

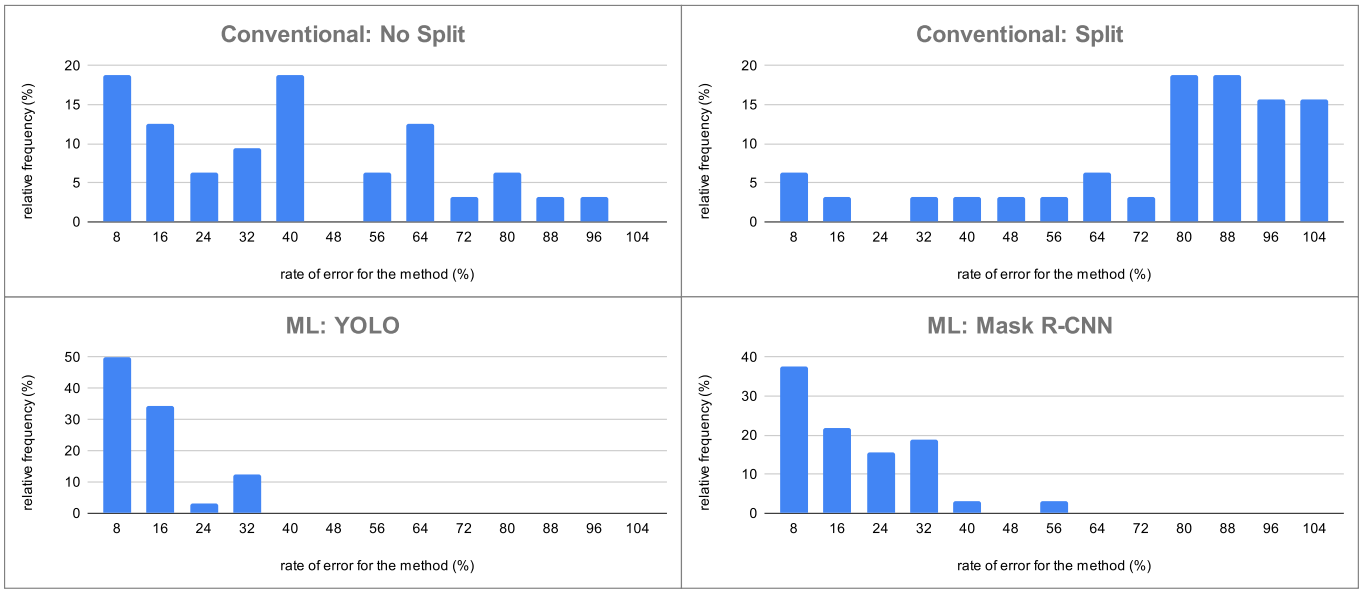


FIG. 10. Relative frequency histograms for each algorithm. These display the percentage detection error for an image on the horizontal axis, the vertical axis shows the percentage of images where that error rate occurred.

VII. CONCLUSION & FUTURE WORK

Having investigated the various machine learning algorithms for detecting apples at an orchard, our machine learning based approaches produce more accurate results compared to conventional methods when applied to a dataset solely consisting of red apples on full trees.

The conventional method was analysed with and without image segmentation. The results showed that the method with no segmentation detected more apples, but also produced more false-positive detections. Two machine learning algorithms were used, both outperforming the conventional methods. This performance difference is owed primarily to these aglorithms improved discernment of separate apples in bunches, compared to the conventional algorithms. Machine learning also produced far fewer false-positives, improving their real world utility. The YOLO algorithm produced lower levels of error compared to the MASK R-CNN, making it the most successful algorithm developed in this study.

This project is only a brief investigation into the agricultural application of machine vision. With further development, this work could be developed for real utility within the agricultural industry. Crop yield estimation to enable precision agriculture is essential in apple orchard management. There are difficulties in designing dependable, automated counting systems that are particular to the task at hand. Uneven lighting, noise, obscured objects, and clumped objects are a few challenges in counting encountered during this experiment. The most common source of error is difficulty counting fruit clusters and avoiding counting apples on the ground. Future algorithm iterations must focus on counting clustered apples precisely and prevent counting of fallen apples.

[1] Bargoti, S. and Underwood, J. [2017], Deep fruit detection in orchards, *in* '2017 IEEE international conference on robotics and automation (ICRA)', IEEE, pp. 3626–3633.

[2] Brownlee, J. [2018], 'A gentle introduction to weight constraints in deep learning'.
URL: *shorturl.at/dgT35*

[3] Chen, D. [2022*a*], 'Tutorial_four.ipynb', Blackboard.

[4] Chen, D. [2022*b*], 'Week 4: Image thresholding and binary morphology', Blackboard.

[5] Chen, S. W., Shivakumar, S. S., Dcunha, S., Das, J., Okon, E., Qu, C., Taylor, C. J. and Kumar, V. [2017], 'Counting apples and oranges with deep learning: A data-driven approach', *IEEE Robotics and Automation Letters* **2**(2), 781–788.

[6] Chen, Y., Li, W., Sakaridis, C., Dai, D. and Van Gool, L. [2018], Domain adaptive faster r-cnn for object detection in the wild, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 3339–3348.

[7] *Coco : Common objects in context* [n.d.], Accessed Jan 02, 2023 [Online].
URL: *https://cocodataset.org/*

[8] *Computer Vision: Opening and Closing* [n.d.], Accessed Jan 01, 2023 [Online].
URL: *https://cvexplained.wordpress.com/2020/05/18/opening-closing/*

[9] *Farm performance and productivity* [n.d.]. Accessed: 2022-11-20.
URL: $://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/955919/fbs - evidencepack - 28jan21.pdf$

[10] Gongal, A., Amatya, S., Karkee, M., Zhang, Q. and Lewis, K. [2015], 'Sensors and systems for fruit detection and localization: A review', *Computers and Electronics in Agriculture* **116**, 8–19.

[11] He, K., Gkioxari, G., Dollár, P. and Girshick, R. [n.d.], 'Mask r-cnn'.
URL: $https://openaccess.thecvf.com/content_ICCV_2017/papers/He_Mask_R-CNN_ICCV_2017_paper.pdf$

[12] Heat, A. [2022], 'pylabel: Transform, analyze, and visualize computer vision annotations.'.
URL: *https://pypi.org/project/pylabel/*

[13] Häni, N., Roy, P. and Isler, V. [2019], 'Minneapple: A benchmark dataset for apple detection and segmentation'.

[14] Jocher, G. [2020], 'YOLOv5 by Ultralytics'.
URL: *https://github.com/ultralytics/yolov5*

[15] Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L. and Dollár, P. [2014], 'Microsoft coco: Common objects in context'.
URL: *https://arxiv.org/abs/1405.0312*

[16] Neill, J. O., Steeg, G. V. and Galstyan, A. [2020], 'Compressing deep neural networks via layer fusion'.
URL: *https://arxiv.org/abs/2007.14917*

[17] PyTorch [n.d.], 'Torchvision object detection finetuning tutorial'. Accessed: 2022-11-20.
URL: $://pytorch.org/tutorials/intermediate/torchvision_tutorial.html$

[18] Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. [2015], 'You only look once: Unified, real-time object detection'.
URL: *https://arxiv.org/abs/1506.02640*

[19] *Roboflow* [n.d.], Accessed Jan 02, 2023 [Online].
URL: *https://roboflow.com/*

[20] Shahin, M., Tollner, E., McClendon, R. and Arabnia, H. [2002], 'Apple classification based on surface bruises using image processing and neural networks', *Transactions of the ASAE* **45**(5), 1619.

[21] Si, Y., Liu, G. and Feng, J. [2015], 'Location of apples in trees using stereoscopic vision', *Computers and Electronics in Agriculture* **112**, 68–74.

[22] Stein, M., Bargoti, S. and Underwood, J. [2016], 'Image based mango fruit detection, localisation and yield estimation using multiple view geometry', *Sensors* **16**(11), 1915.

[23] *StepLR — PyTorch 1.9.0 documentation* [n.d.].
URL: *https://pytorch.org/docs/stable/generated/torch.optim.lr$_s$cheduler.StepLR.html*

[24] *torchvision — Torchvision master documentation* [n.d.].
URL: *https://pytorch.org/vision/stable/index.html*

[25] *viso.ai Deep Learning : Mask R-CNN* [n.d.].

[26] Wu, D., Lv, S., Jiang, M. and Song, H. [2020], 'Using channel pruning-based yolo v4 deep learning algorithm for the real-time and accurate detection of apple flowers in natural environments', *Computers and Electronics in Agriculture* **178**, 105742.