MDPI

Article

Estimating Apple Yield Using Video Images And Deep Learning

Niroj Shrestha [1,‡], Daniel L. Silver [1,†,‡*] and Brett Connell[2]

[1] Acadia University; nirojshrestha019@gmail.com, danny.silver@acadiau.ca
[2] Pomona Farms; brettconnell4@gmail.com
* Correspondence: danny.silver@acadiau.ca; Tel.:+1-902-585-1413
† Current address: Jodrey School of Computer Science, Acadia University, Wolfvile, NS Canada B4P2R2.
‡ These authors contributed equally to this work.

Abstract: We use low-cost video capture, computational vision techniques, and deep learning methodologies to estimate apple yield while the apples are still on the tree. Video of a variety bi-coloured apples is captured from both sides of an orchard row using a smartphone. A convolutional neural network (CNN) based on the YOLOv3 architecture is trained to detect apples in the sequence of video images. The tracking of each detected apple is done using a Kalman filter. The Hungarian algorithm, inclusive of motion and appearance information, is used to solve the association problem between the previously tracked and the newly detected apples. The count is derived using this tracking information. The system generates a diagnostic video that shows the distinction between the apples that are being tracked and those that are counted by a change in the color of the bounding box surrounding the apple. The experiments show that the proposed framework performs best with transfer learning and data augmentation techniques. The final predictive model for apple detection has an AP (average precision) score of 90.72, the tracking model has an MT (mostly tracked) score of 36.09, and the system counts apples with an F1-score of 94.97%.

Keywords: estimating apple yield; convolutional neural networks; YOLOv3; transfer learning; data augmentation; object detection; object tracking.

1. Introduction

As a high-value crop, apples are intensively managed with much of this management being associated with yield estimates. Accurate prediction of apple yield is relevant for agricultural and market planning by growers, wholesalers, supermarkets, and exporters in terms of labor, packing, storage, and transportation. To date, yield estimation is mainly based on the historic performance of an orchard in the previous years combined with periodic estimates from sample trees in the current year. This results in estimates of varied quality from farm to farm and between apple varieties.

Advancements in technology have made high-resolution monocular cameras readily available in most of today's smartphones. So, it would be more practical and cost-effective to build a computer vision system capable of estimating yield in the field using commodity hardware such as that found in smartphones. These factors motivated the investigation of a high-performance, cost-effective and practical apple yield estimation system that uses a monocular camera on a smartphone. The research was commissioned and funded by Scotian Gold Cooperative Limited, of Kentville, Nova Scotia, as well as by the National Research Council of Canada Industrial Research Assistance Program (NRC IRAP) and the Mitacs Accelerate program.

The objective of this research is to develop a low-cost but effective computer vision method to predict the apple yield of an orchard up to 4 weeks prior to harvest. To achieve this objective our approach will be as follows: (1) Capture video of apple trees in the orchard environment using a smartphone up to one month in advance of harvest; (2) Use computational vision and deep learning techniques to detect apples in the video; (3) Track

Citation: Lastname, F.; Lastname, F.; Lastname, F. Title. *Journal Not Specified* **2022**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

detected apples across a sequence of images (video frames) avoiding the double counting of apples; (4) Estimate the overall apple yield based on the count of apples.

The detection of apples from a sequence of images is a challenging task for several reasons, including appearance variability due to illumination and occlusion due to foliage and fruits. So, the success criterion for the detection problem is set to a minimum average precision (AP) of 90 based on prior work [1]. Further, tracking each detected individual apple in a continuous sequence of images to prevent double counting is another challenge. The success criterion for the counting problem is set at a modest F1-score of at least 90%.

2. Prior Methods of Fruit Yield Estimation

Older hand-crafted computer vision methods usually have difficulty generalizing to variations in illumination and object occlusion levels. Consequently, data-driven methods have become the state of the art, primarily as a result of advances in deep learning methods. Chen, Steven W., et al. used a blob detector based on a Fully Convolutional Network (FCN) to extract candidate fruit regions from stand-alone high-quality images [1]. A counting algorithm based on a second convolutional network estimates the number of fruit in each region. Finally, a linear regression model maps the fruit count estimate to a final fruit count. The mean absolute percentage error (MAPE) was 13.8% on an orange orchard dataset and 10.5% on an apple orchard dataset.

Various data collection methods have been used for capturing video images of fruit on the trees in an orchard. Liu, Xu, et al. used a monocular camera system and a sensor-suite-based system for data collection in a mango orchard [2]. These systems were mounted on an unmanned ground vehicle (UGV) which consisted of a 3D LIDAR, GPS (global positioning system) inertial navigation system capable of real-time-kinematic correction, and a Prosilica GT3300C camera with a Kowa LM8CX lens that captured RGB images of size 3296 x 2672 pixels (8.14 megapixels) at 5 Hz.

Liu, Xu, et al. also used two other approaches to collect data in an orchard environment [3]. Images from an orange orchard, each of size 1280 x 960, were captured using a Bluefox USB 2 camera at 10 Hz mounted on SteadiCam gimbal carried by a human operator travelling at walking speed during daylight hours. Similarly, images of apples on the tree, each of size 1920 x 1200, were captured using a PointGrey USB 3 camera at 6 Hz mounted on a utility vehicle with an external flash at night driving down the row at around 1 m/s.

Liu, Xu, et al. presented a fruit counting pipeline approach that combined deep segmentation, frame to frame tracking, and 3D localization to accurately count visible fruits across a sequence of images [3]. First, a Fully Convolutional Network was trained to segment video frame images into fruit and non-fruit pixels. Fruits across frames were tracked using the Hungarian algorithm where the objective cost was determined by a Kalman filter. In order to correct the estimated count from the tracking process, the Structure from Motion (SfM) algorithm was used to calculate relative 3D locations for rejecting the outliers and double-counted fruit tracks. Here, a mean error of 0.2% on the fruit count was calculated for the orange dataset and likewise, a mean error of 3.3% for the apple dataset.

Automatic robotic fruit counting systems using LidAR have been developed [4] [5]. These systems have demonstrated success in counting a variety of fruits including apples, mangoes, and oranges; however they are currently quite costly. For example, a sensor suite equipped with cameras, LiDAR, and a computer can cost more than $10,000. The infrastructure, technical knowledge, and high cost constraints make it impractical to use such a complex sensor suite in many agricultural environments.

3. Theory and Approach

This section provides the theory and approach of our research. Specifically, it describes the data collection process, image pre-processing techniques, the network architectures for apple detection, tracking and counting, and the evaluation methods used.

3.1. Data Collection

A Samsung Galaxy S3 smartphone was used as the primary camera for data collection. Video was captured from the first week of August until the second week of September 2020. Data was collected from a row of bi-coloured (red and yellow) apples at Pomona Farms, in Canard, Nova Scotia. The row consisted of 179 trees in total. Several image capturing approaches were considered: (1) Camera with tripod, (2) Camera mounted on vehicle, (3) Camera with stabilizer, (4) Person holding the camera on the back of ATV, (5) Camera with artificial light, and (6) Intel RealSense depth camera D435. The method that worked best for this research was to have a person hold the camera on the back of an ATV as shown in Figure 1. The video frames were clear and sharp despite having some minor stability problems. Videos were taken of all 179 trees of a specific row from both sides. Data was collected once per week from the first week of August until harvest day (10 September 2020) using this method.

Two sources of ground truth were used for the apple count: (1) *Video ground truth*: the count of apples as seen by the research team on each tree in the videos. (2) *Orchard ground truth*: the actual apple count on each tree as seen by human eyes. Three teams consisting of two people each manually counted the fruit from 179 trees in a row of the apple variety videoed. The average count over the three teams was taken as ground truth coincidental with the video captured the same day.

Figure 1. Primary data collection method - person holding camera on back of ATV.

3.2. Data Preparation for Apple Detection

This section describes the preprocessing of the smartphone video data into a sequence of images for training and testing the apple detection model.

Conversion of video into image frames - The captured videos of the orchard row are about 6 minutes and 30 seconds long and shot at 30 frames per second (fps). Since 30 frames per second contain images with redundant information, a Python script was written to select 2 frames per second from the video. For our experiment, we selected a video that was captured on the 9th of September, just prior to harvest the next day which consist of mostly red apples. Figure 2 shows examples of image frames that were extracted from the video. These converted images were then annotated and used for training and testing the initial apple detection model.

Image annotation - Our research uses a supervised machine-learning technique for apple detection. This requires that the train and test data be labeled with the correct locations of apples. So, an open-source Python-based image annotation tool called LabelImg [6] is used to annotate the apples in the sequence of images. The annotated files are saved in Pascal Visual Object Classes (VOC) format. About 250 images were annotated. Verification

Figure 2. Sample image frames from smartphone video.

of the annotations is done to make sure that the apples are correctly labeled. Figure 3 shows the screenshot of the LabelImg tool that is used to annotate the apples in the images.

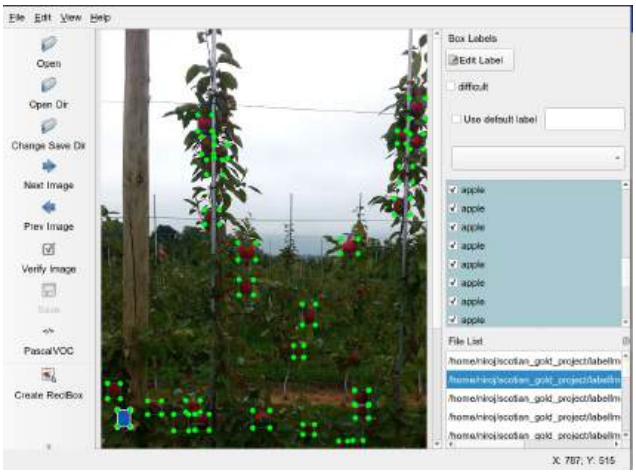


Figure 3. LabelImg tool used to annotate the image.

Image normalization and scaling - Preprocessing of each image is done to decrease the model training time and increase the model accuracy. Image normalization is done to ensure a uniform homogeneous distribution of the data. Each pixel value is subtracted by the mean pixel value of the image and then is divided by the standard deviation. As most of the images produced by modern cameras are high resolution, image scaling is used for reducing the input size to save computational time and space. We tried various image scales (320 x 320, 416 x 416, 736 x 736) and found that 416 x 416 pixels provide good accuracy with reasonable neural network training times.

3.3. Data Preparation for Appearance Feature Extractor

Images were prepared for training the appearance feature extractor, which was critical for tracking each apple from frame to frame. The source of data were the images prepared and annotated for object detection. A Python script was written to crop each bounding box of apples into a separate image file. The training set for the appearance feature extractor consisted of 2,972 images that are resized to 128 $\times$ 256. Figure 4 shows several cropped individual apples from the annotated sequence of images. An image scaling technique, similar to that described in Section 3.2, is applied to these images. The only difference is that the images are resized to 128 x 256 pixels before feeding them into the neural network. Since the cropped images tend to be of different sizes, we do not preserve the aspect ratio while resizing.

3.4. Object Detection Model:

The single-stage detector YOLOv3 [7] architecture is adopted for training the apple detection model. The whole architecture can be divided into two major components as shown in Figure 5: a *feature extractor* and a *detector*. A new image goes through the feature extractor first so that feature embeddings are obtained at three different scales. Then, these

Figure 4. Cropped individual apple images.

features are fed into three (or more) branches of the detector to get bounding boxes and class information.

Figure 5. Components breakdown of YOLOv3 model.

Feature Extractor: The feature extractor that YOLOv3 uses is called Darknet-53. The Darknet version from YOLOv1 [8] had only 19 layers. ResNet [9] provided the idea of skip connections to help neuron activations propagate through deeper layers without the gradient diminishing. Darknet-53 borrows this idea and successfully extends the network from 19 to 53 layers, as shown in Figure 6.

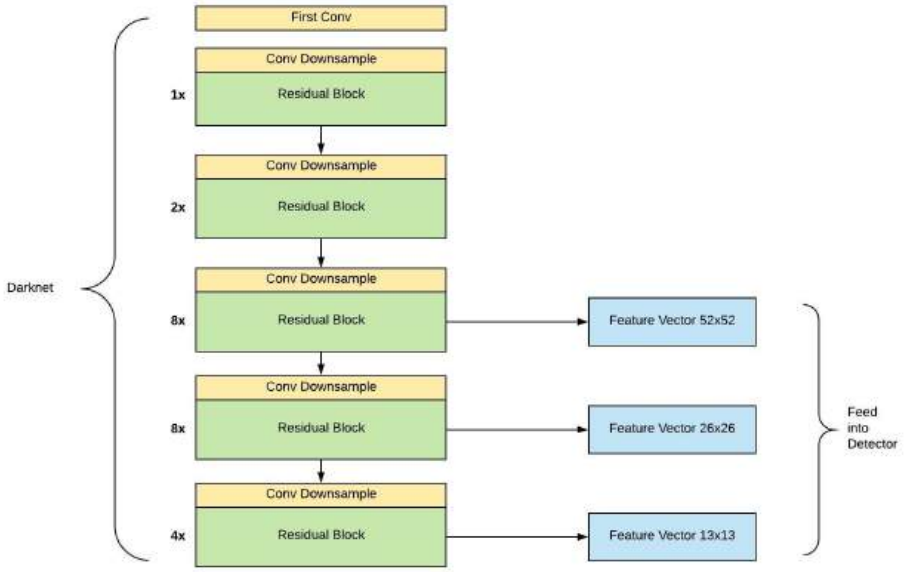


Figure 6. YOLOv3 multi-scale feature extractor.

Multi-scale Detector: Once we have three scaled feature vectors, we feed them into the detector as shown in Figure 7. Multiple 1 x 1 and 3 x 3 convolutional layers are used before

a final 1 x 1 convolutional layer to form the final output. The 1 x 1 filters [10] downsample the depth or number of feature maps. For medium and small-scale features, the filters also concatenate the features from the previous scale. By doing so, small-scale detection can also benefit from the result of large-scale detection.

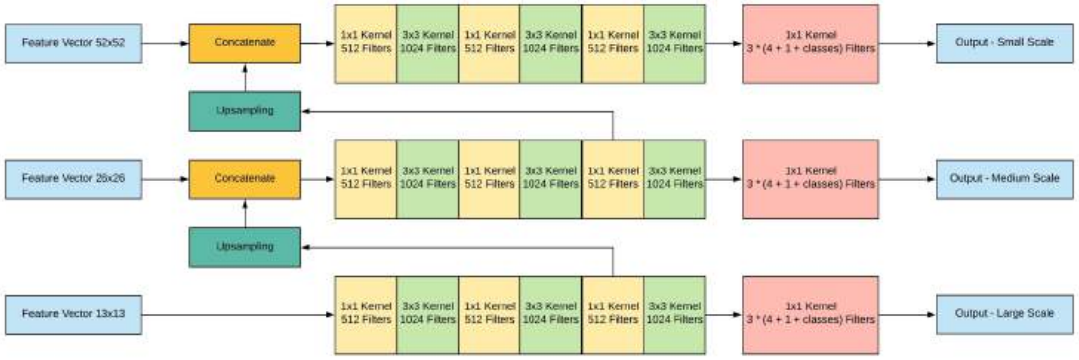


Figure 7. YOLOv3 multi-scale detector.

Anchor Box: The anchor box can have different pre-defined aspect ratios. These aspect ratios are determined before training by running k-means on the entire dataset. In this research we use the anchor boxes that were determined by k-means clustering on the COCO dataset as mentioned in the original YOLOv3 paper [7] and they are (10 x 13), (16 x 30), (33 x 23), (30 x 61), (62 x 45), (59 x 119), (116 x 90), (156 x 198), and (373 x 326). The convolution outputs a square matrix of feature values (like 13 x 13, 26 x 26, and 52 x 52 in YOLO). We define this matrix as a grid and assign anchor boxes to each cell of the grid. In other words, anchor boxes anchor to the grid cells, and they share the same centroid. Once we defined those anchors, we can determine how much the ground truth box overlaps with the anchor box and pick the one with the best IoU (intersection over union), and couple them. Figure 8 shows the three different scale anchor boxes that we apply to each grid of the whole image grid cells. These multi-sized anchor boxes make it possible to detect varying size apples.

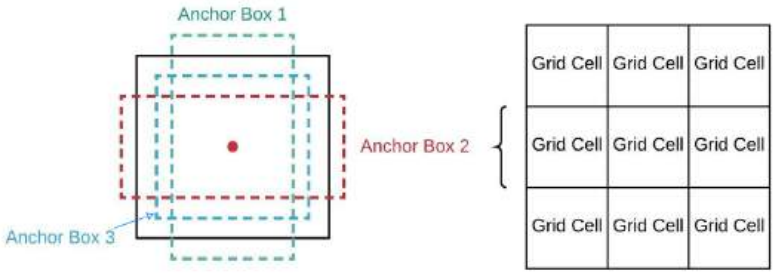


Figure 8. YOLOv3 anchor box and grid cells.

In YOLOv3, we have three anchor boxes per grid cell, as mentioned before, as well as three scales of grids. Therefore, we will have 52x52x3, 26x26x3, and 13x13x3 anchor boxes for each scale. Each anchor box makes three predictions:

- The location offset against the anchor box: t_x, t_y, t_w, t_h. This has 4 values.
- The objectness score to indicate if this box contains an object. This has 1 value.
- The class probabilities to tell us which class this box belongs to. This has num classes values.

Assuming the input image is (416, 416, 3), the final output of the detectors will be in shape of [(52, 52, 3, (4 + 1 + *num* classes)), (26, 26, 3, (4 + 1 + *num* classes)), (13, 13, 3, (4 + 1 + *num* classes))]. The three items in the list represent detections for three scales and (4 + 1 + *num* classes) values are the predictions for one anchor box. The real coordinates of the bounding box are not t_x, t_y, t_w, t_h. These are just the relative offsets compared with a particular anchor box. The following formula describes how the network output (i.e., the relative offsets for anchor box) is transformed to obtain bounding box predictions:

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_w &= p_w e^{t_w} \\ b_h &= p_h e^{t_h} \end{aligned} \tag{1}$$

where (t_x, t_y, t_w, t_h) are outputs of the network, (b_x, b_y, b_w, b_h) are the predicted bounding box, (c_x, c_y) are the top-left coordinates of the anchor, and (p_w, p_h) are width and height of the anchor.

3.5. Object Tracking Model

The multiple objects tracking DeepSORT [11] architecture was adopted for tracking the detected apples. DeepSORT extends SORT [12] by incorporating appearance information for each tracked object. The position information of all detected apples, stored in an eight-dimensional matrix, is extracted and used to represent the current state of the target. The matrix is given by *(u, v, γ, h, x′, y′, γ′, h′)* that contains the bounding box center position *(u, v)*, aspect ratio γ, height h, and their respective velocities in image coordinates which is (0,0,0,0) initially. A standard Kalman filter [13] with constant velocity motion and linear observation model is used where it takes the bounding coordinates *(u, v, γ, h)* as direct observations of the object state. The state of the current target is fed into the Kalman filter, and the target is predicted and updated.

The association problem between the predicted Kalman states and newly arrived measurements is solved with the Hungarian algorithm [14], which is a combinatorial optimization algorithm that solves the assignment linear-programming problem (matching between previously seen apple detection and the current detection) in sequential time (i.e., consecutive frames of video images). The Mahalanobis distance is used to incorporate motion information by calculating the distance or discrepancy between the predicted Kalman state and the newly arrived measurements of the detection [15].

Mahalanobis distance only considers the distance relationship between the detection target and the prediction target, which is suitable for the association when motion uncertainty is low. The unaccounted camera motion can introduce rapid displacements in the image plane, making the Mahalanobis distance an uninformed metric for tracking through any occlusions of the object in an image sequence. Another metric, known as the appearance metric, is incorporated into the assignment problem. It measures the dissimilarity in appearance feature metric into the cosine distance between the j^{th} detection and the history of the i^{th} track. For the previously calculated Mahalanobis and the appearance cosine distances, a fusion is required and given by $c_{i,j} = \lambda d^{(1)}(i,j) + (1-\lambda)(d^{(2)}(i,j)$ where λ is a hyperparameter used to adjust the weights of two items. We set λ to 0.1 in this research work.

3.6. Evaluation

Different evaluation metrics are used to determine the performance of the various models for object detection, object tracking, and object counting.

Object Detection: Average Precision (AP) is used as an evaluation metric for object detection models because it takes into consideration both the classification loss and local-

ization loss. It is calculated from the following statistics: *precision, recall*, and *intersection over union or IoU*.

Precision measures relative predictions that are correct, and is given by $P = \frac{TP}{TP+FP}$, where TP is the total number of true positives, and FP is the total number of false positives [16]. *Recall* measures the proportion of the total correctly classified and is given by $R = \frac{TP}{TP+FN}$ where FN is the number of false negatives [16].

IoU is given by the ratio of the intersection area of the ground truth box and the predicted box over the union area of both boxes.

$$IoU = \frac{\text{Area of intersection}}{\text{Area of union}} \tag{2}$$

Figure 9 shows an example of intersection over union. We consider the yellow highlighted box as predicted and the blue highlighted box as a ground truth. The intersection between them is denoted by the green highlighted box. The first example has no intersection between the predicted and ground truth bounding box, so the IoU is 0. Likewise, the third example has IoU=1 as both the predicted and ground truth bounding boxes completely overlap.

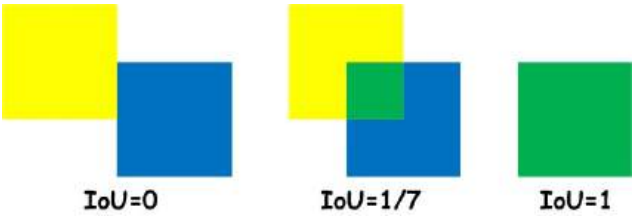


Figure 9. Intersection over union example [17].

An IoU threshold value is used to determine if a predicted bounding box is TP, FP, or FN. Bounding boxes are considered as FP when their IoU is under a certain threshold or if there are duplicated bounding boxes. They are considered FN when ground truth is present in the image, and the model failed to detect the object. Only the predicted bounding boxes with IoU above the threshold value are considered TP.

The precision P is plotted against the recall R value for each of the detected objects to get the average precision (AP) [18]. AP is the area under the precision-recall curve (AUC-PR) for a class. It is defined as the weighted sum of precisions at each threshold, where the weight is the increase in recall. Mathematically, AP is given as

$$AUC\text{-}PR = AP = \sum_{k=0}^{n-1} [Recalls(k) - Recalls(k+1)] * Precisions(k) \tag{3}$$

where n is the number of thresholds.

Object Tracking: The evaluation metric used for object (apple) tracking in this research is Mostly Tracked (MT). A target is mostly tracked if it is tracked for at least 80% of its life span [19]. This means if a target object can be seen in 10 video frames then that object will only be considered as mostly tracked when it is tracked for at least 8 frames. The higher the MT value the better.

Object Counting: The evaluation metric for apple counting is the *F1-score*, which is defined as the harmonic mean of Precision and Recall, $F1\text{-}Score = \frac{2*Precision*Recall}{Precision+Recall}$.

4. Model Development Using 5-Fold Cross-validation with Transfer Learning and Data Augmentation

Objective: Prior research has shown that transfer learning and data augmentation techniques can improve deep learning model performance significantly. The objective of this experiment is to train our object detection model with both transfer learning and data

augmentation techniques and to evaluate its performance using a k-fold cross-validation experimental design.

Data and Methods: For this experiment, 40 images are randomly chosen from a sequence of images taken during the daytime. The image processing technique described in the prior section is used to prepare the data for this experiment.

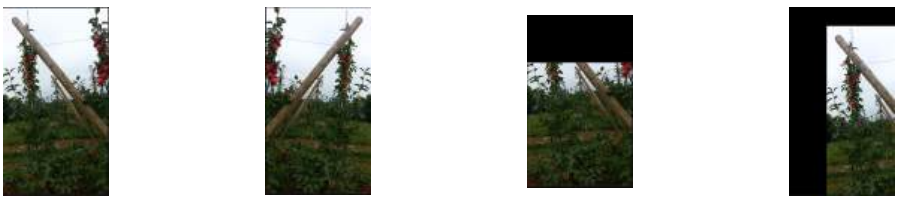
The YOLO architecture is adopted for training our object detection model using the Tensorflow and Keras frameworks as described in Section 3.4 and shown in Figure 6. Features from the last three residual blocks are extracted for multi-scale detection. Both the F1 score and IoU threshold are set at 0.7 by the iterative method. The output consists of bounding box coordinates, objectness score (0 or 1 to indicate if the box contains an object), and confidence of the detected apples. The cost function consists of four parts: centroid loss, width and height loss, confidence loss, and classification loss. The total loss is the sum of all the losses where the mean square error is used to calculate the first two losses, and cross-entropy loss is used for the last two losses.

We conducted early experiments to determine the best input image size so that we can decrease model training time without compromising on performance. We chose three different image sizes: 320 x 320, 416 x 416, and 736 x 736 for training respective models. We found that images that are 416 x 416 would be the best choice because their models have minimum computational time with a negligible decrease in AP score.

Transfer learning is a technique where a neural network model is trained on one task and then used as the starting representation (weights value) for learning a different but related task. The pre-trained model used to transfer knowledge has been trained on the Microsoft Common Objects in Context (MS COCO) [20] dataset in which the apple is one of 80 categories for object detection. Weights from a model trained on the MS COCO dataset are used to initialize our apple detection model which only outputs one class (apple) before training begins and then all the weights in the network are fine-tuned using the training dataset.

The idea of data augmentation is to increase the amount of available training data by creating variants of the original images through linear transformations of the pixels. Three data augmentation techniques are applied to the images used to train the apple detection model: (1) *Horizontal flip*: the columns of pixels for each image sample are reversed. (2) *Random crop*: a random portion of the original image is selected, and the cropped image is padded to match the input size 416 x 416 before feeding it into the neural network. (3) *Translation*: the object's location is shifted along the x and y-axis randomly with respect to the width and height of the original image [21]. Figure 10 shows examples of the different data augmentation techniques used. The annotations of the augmented images are also translated respectively following the change in the original image.

The data augmentation and transfer learning techniques described above are applied during training and the model evaluation is conducted using a 5-fold cross-validation approach. The model is trained for 500 epochs using the Adam optimizer with an initial learning rate of 0.0001 and with early stopping to prevent overfitting.



(a) original image. **(b)** horizontal flip. **(c)** random crop. **(d)** translation.

Figure 10. Different data augmentation techniques.

Results and Discussion: The results of 5-fold cross-validation evaluation show that the training of the model with both transfer learning and data augmentation has a significant improvement to previous experiments with an average AP score of 95.64. There are 141

true positives and 7 false-positives out of 147 total ground truth apples. This is a TP rate of 0.95 and a FP rate of 0.05.

5. Object Detection Using a Sequence of Video Images

Objective: In the previous experiment, the system is tested on a single image at a time. The objective of this experiment is to create a system that can use this approach for a sequence of images from a video.

Data and Methods: A larger dataset is used for this experiment. A sequence of 232 daylight images are selected and split into 174 train, 32 validate, and 26 test images. Both transfer learning and data augmentation techniques are used for this experiment. The same image processing techniques from Section 3.2, the YOLO neural network architecture, and hyperparameters described in Section 3.4 are used to develop the model for this experiment. The training is done for 500 epochs.

The top portion of Figure 11 shows the sequence of steps used to go from the input video to the detection of apples in the form of bounding boxes. We added functionality to accept any video as input for testing. The core idea is to convert the input video into a sequence of images, apply image pre-processing techniques to each frame and then feed it to our trained model for prediction. Furthermore, the predicted sequence of images is stacked together to form a video. The intention behind this experiment is to detect the apples in each frame of the video and pass this information to the next phase for tracking each of them as they appear. The time required to process the test set is carefully recorded so as to determine the throughput rate in frames per second.

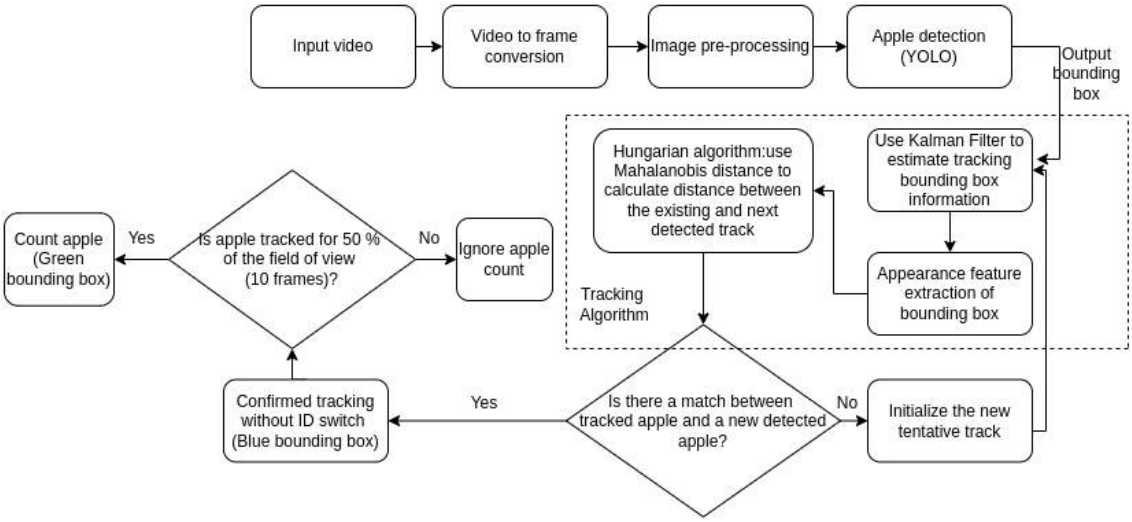


Figure 11. Apple detection and tracking workflow.

Results and Discussion: The system, with an IoU threshold of 0.7, achieved an AP score of 90.72 where 351 out of 380 video ground truth apples are detected in the test set of 26 video image frames. This is a TP rate of 0.92 and a FP rate of only 0.09. This experiment also revealed that the system can process a video file at 9.4 frames per second (fps). In the future, if the system could be optimized to 30 fps then it could be used to detect apples in real time, providing feedback to the operator or control software.

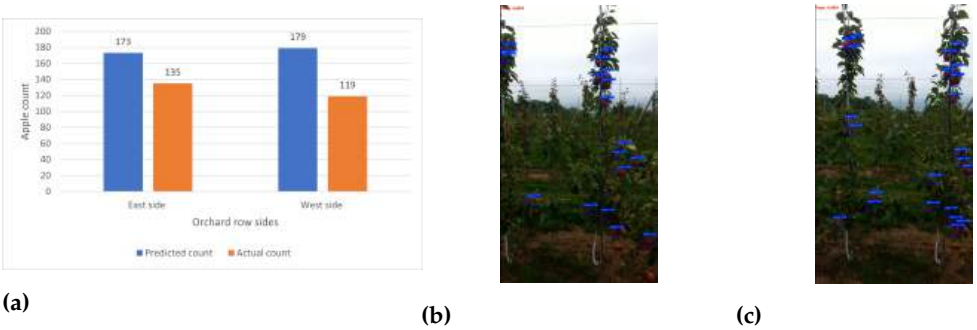
6. Object Tracking with Appearance Feature Descriptor

Objective: In the previous section the apple detection model was trained to detect apples with an AP score greater than 90. The apples are detected accurately but the same apples appear in multiple frames of a video, therefore, it is necessary to avoid double counting each apple. The objective of this experiment is to track each detected apple from frame to frame so that it is counted only once.

Data and Methods: The object detection model from the previous experiment is used to detect apples in each frame. The DeepSORT [11] algorithm, as described in Section 3.5, is implemented to solve the tracking problem. It takes in the output from the previously trained object detection model (i.e., bounding box coordinates of detected apples). A Kalman filter is used for tracking the detected apples from frame to frame. The Hungarian algorithm, which is inclusive of the appearance features information, is used for solving the association problem between the tracked and the detected apples. The appearance feature extractor is trained to extract the appearance information from each detected apple to enhance tracking in scenarios where there is occlusion of an apple in a frame. Figure 11 shows the workflow of the tracking method surrounded by a dashed square.

A total of 2,972 images of apples were extracted from the annotated object detection training set, as described in Section 3.3, for training the feature extractor. The Adam optimizer is used to train the appearance feature extractor model with an initial learning rate of 0.001 and cosine distance as the cost function. The model is trained for 500 epochs with a batch size of 32. Videos taken from both sides of the first 10 trees in the row are used as a test set.

Results and Discussion: The evaluation of the DeepSORT tracking algorithm determined that 36.09% of the tracked apples have the same track ID for at least 80% of their life span (those Mostly Tracked). Figure 12a shows the comparison of predicted results from this experiment versus the video ground truth count of the apples on the first 10 trees. The total count of the predicted apples is more than the actual count due to the switching of the IDs assigned to the tracked apples. One of the reasons for ID switching is the occlusion of apples by leaves and branches. Another reason is the appearance feature similarity between different detected apples that are close to each other. Figure 12b and 12c shows two consecutive output image frames from the tracking algorithm. The blue bounding box shows the detected apples and has the same tracking ID in the consecutive image. This tracking information is important to determine the accurate count of the apples in the video as described in the following experiment.



(a) **(b)** **(c)**

Figure 12. (a) Comparison of predicted vs actual apple count, (b) and (c) example output frames from tracking algorithm.

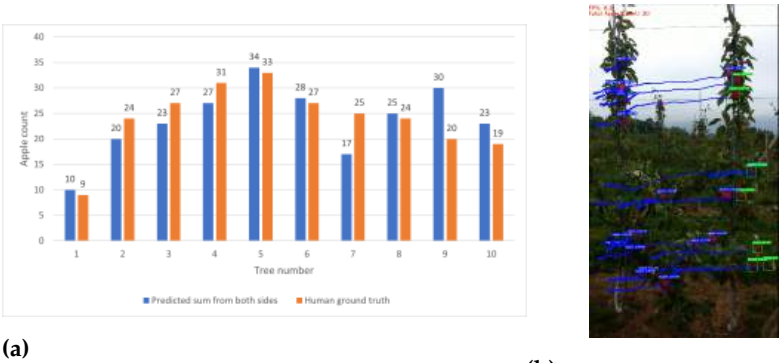
7. Counting Tracked Apples Based on Their Age in the Video

Objective: In the previous experiment, the count is only dependent on the track ID of each detected apple and this results in predicting more apples than are actually present, due to occlusions and other noise in the video frames. The objective of this experiment is to determine the accurate count of the apples by considering the *age* of the tracked apple in the video. We also develop a method to visualize the tracking of apples in the video, which serves as a diagnostic tool for refining our approach to object detection and tracking.

Data and Methods: We define the *age* of the tracked object as the number of contiguous frames of a video in which an object ID is tracked. Only an apple with an age above a predefined threshold is considered for counting. Videos taken from both sides of the first 10 trees in the row are used as the test set. It takes 20 frames on average for an apple to enter and exit from the defined region of interest. The age threshold to consider counting an

apple is chosen as 10 frames or 50% of the average number of frames. A different bounding box color is used for the tracking and counted apple. A blue bounding box is used to indicate a detected apple in the video, and the color is changed to green and counted after being tracked for more than 10 frames. A blue tracing path for each apple is also included to provide diagnostic information. This is done by labeling the central location of each tracked apple by a blue dot and these dots make the tracing path in the consecutive video frames. The left side of Figure 11 shows the workflow of the counting method used.

Results and Discussion: This experiment shows that the proposed system achieves an F1-score of 94.97% for counting apples, which falls well within our success criteria. The model's precision was 98.29 and its recall was 91.87. Figure 13a shows the comparison of the sum of the predicted count from both sides (east and west) of the first ten trees with the orchard ground truth for those trees. The total predicted count is 241 whereas the actual orchard ground truth count is 239. Although the sum of predicted apples from both sides is close to the ground truth, there are two problems that need to be addressed: some apples continue to be missed due to occlusion from leaves and branches, and some apples are double-counted because they are visible from both sides of the row. To solve the double-counting issue, we have explored a method that is able to determine on which side of a tree an apple is located. This work will be documented in a future paper where we also consider counting the apples by their size (grade).



(a) **(b)**

Figure 13. (a) Comparison of predicted sum of apples vs the actual human count in orchard and (b) Screenshot from video output of our final model.

Figure 13b shows a screenshot of the output video produced by our system for this experiment. Blue bounding boxes indicate detected apples, blue lines are the traced path of the tracked apple, green bounding boxes indicate the counted apples, and the text at the upper top left provides the apple count and the frame rate of the images processed by our model. The results of apple detection with tracking on the video files show that our model performs on average 8.6 fps which is just a little slower than apple detection. This is because of the additional apple counting steps.

8. Conclusion

The accurate prediction of apple yield is relevant for agricultural and market planning by growers, wholesalers, supermarkets, and exporters for estimating labor, packing, storage, and transportation requirements. Currently, yield estimation is based on the historic performance of an orchard combined with estimates from sample trees in the current year. This results in estimates of varied accuracy from farm to farm and between different apple varieties. The objective of this research is to develop a low-cost but effective computer vision method to predict the apple yield of an orchard up to 4 weeks prior to harvest.

This research develops a data collection and deep learning approach that captures video of apples on the tree using an inexpensive smartphone and provides frames from the video to a deep learning neural network for the detection, tracking, and counting of each apple leading to a yield estimation. Empirical results have shown that our method can develop models that detect apples with an AP score of 90.72, track apples with an MT score

of 36.09, and count apples with an F1-score of 94.97%. Our apple counting pipeline paves the way for yield estimation using commodity smartphone technology. Such a system has applications in a wider variety of farm environments where cost and environment constraints prevent the usage of high-cost, larger sensors.

The most significant findings are: (1) Video captured by an inexpensive smartphone on the back of an ATV during daylight hours provides the fidelity needed in single frames to detect and track apples on the tree; (2) Combining both the transfer learning and data augmentation techniques had the overall best performance in the apple detection model; and (3) Tracking each detected apple is difficult due to the similarity of different apples. To determine a more accurate count, the *age* factor (number of contiguous frames in which an apple is tracked) was created and used successfully.

Areas of future work include: (1) The use of 3D camera technology, such as the Intel RealSense depth camera, to capture RGB and depth images that will accurately estimate the distance to each pixel and provide a method of determining the size of the apples; (2) Development of a method of mounting the video camera to an ATV for data collection in the orchard; (3) Use data augmentation by changing the color of the apples from red to green or yellow and perform more tests on different varieties of apples; (4) Explore different object detection algorithms like Single Shot Detector (SSD) or Faster R-CNN which might improve the performance of the apple detection model; and (5) Increase the speed of the system and the machine learning models to count apples in the field fast enough to provide real-time feedback to the operator.

References

1. Chen, S.W.; Shivakumar, S.S.; Dcunha, S.; Das, J.; Okon, E.; Qu, C.; Taylor, C.J.; Kumar, V. Counting apples and oranges with deep learning: A data-driven approach. *IEEE Robotics and Automation Letters* **2017**, *2*, 781–788.
2. Liu, X.; Chen, S.W.; Liu, C.; Shivakumar, S.S.; Das, J.; Taylor, C.J.; Underwood, J.; Kumar, V. Monocular camera based fruit counting and mapping with semantic data association. *IEEE Robotics and Automation Letters* **2019**, *4*, 2296–2303.
3. Liu, X.; Chen, S.W.; Aditya, S.; Sivakumar, N.; Dcunha, S.; Qu, C.; Taylor, C.J.; Das, J.; Kumar, V. Robust fruit counting: Combining deep learning, tracking, and structure from motion. In Proceedings of the 2018 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, 2018, pp. 1045–1052.
4. Stein, M.; Bargoti, S.; Underwood, J. Image based mango fruit detection, localisation and yield estimation using multiple view geometry. *Sensors* **2016**, *16*, 1915.
5. Gené-Mola, J.; Gregorio, E.; Guevara, J.; Auat, F.; Sanz-Cortiella, R.; Escolà, A.; Llorens, J.; Morros, J.R.; Ruiz-Hidalgo, J.; Vilaplana, V.; et al. Fruit detection in an apple orchard using a mobile terrestrial laser scanner. *Biosystems engineering* **2019**, *187*, 171–184.
6. Tzutalin. LabelImg. Github, 2015.
7. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv prepr. arXiv:1804.02767* **2018**.
8. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the Proc. of IEEE conf. on comp. vision and pat. recog., 2016, pp. 779–788.
9. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the Proceedings of IEEE conf. on computer vision and pattern recognition, 2016, pp. 770–778.
10. Brownlee, J. A Gentle Introduction to 1x1 Convolutions to Manage Model Complexity. Machine Learning Mastery, 2019.
11. Wojke, N.; Bewley, A.; Paulus, D. Simple online and realtime tracking with a deep association metric. In Proceedings of the 2017 IEEE international conference on image processing. (ICIP). IEEE, 2017, pp. 3645–3649.
12. Bewley, A.; Ge, Z.; Ott, L.; Ramos, F.; Upcroft, B. Simple online and realtime tracking. In Proceedings of the 2016 IEEE international conf. on image processing (ICIP). IEEE, 2016, pp. 3464–3468.
13. Kalman, R.E. A new approach to linear filtering and prediction problems **1960**.
14. Kuhn, H.W. The Hungarian method for the assignment problem. *Naval research logistics quarterly* **1955**, *2*, 83–97.
15. Pinho, R.R.; Manuel, R.T.J.; Correia, M.V. Efficient approximation of the Mahalanobis distance for tracking with the Kalman filter. In *Computational Modelling of Objects Represented in Images*; CRC Press, 2018; pp. 349–354.
16. Course, G.M.L.C. Classification: Precision and Recall. Google Machine Learning Crash Course, 2019.
17. Sheremet, O. Intersection over union (IoU) calculation for evaluating an image segmentation model. Towards Data Science, 2020.
18. Hui, J. mAP (mean Average Precision) for Object Detection. Medium, 2018.
19. Milan, A.; Leal-Taixé, L.; Reid, I.; Roth, S.; Schindler, K. MOT16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831* **2016**.
20. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In Proceedings of the European conference on computer vision. Springer, 2014, pp. 740–755.
21. OpenCV. Geometric Transformations of Images. OpenCV.