AUTHORS:
Jessica Harris[1]
Sebnem Er[1]

AFFILIATION:
[1]Statistical Sciences Department, University of Cape Town, Cape Town, South Africa

CORRESPONDENCE TO:
Sebnem Er

EMAIL:
sebnem.er@uct.ac.za

DATES:
Received: 26 June 2024
Revised: 05 Feb. 2025
Accepted: 22 May 2025
Published: 29 Sep. 2025

HOW TO CITE:
Harris J, Er S. Pineapple fruit detection and size determination in a juicing factory in the Eastern Cape, South Africa. S Afr J Sci. 2025;121(9/10), Art. #18277. https://doi.org/10.17159/sajs.2025/18277

ARTICLE INCLUDES:
☒ Peer review
☐ Supplementary material

DATA AVAILABILITY:
☒ Open data set
☐ All data included
☐ On request from author(s)
☐ Not available
☐ Not applicable

EDITORS:
Michael Inggs[1]
Thywill Dzogbewu

KEYWORDS:
image classification, mask R-CNN, pineapple fruit, size determination

FUNDING:
None

Pineapple fruit detection and size determination in a juicing factory in the Eastern Cape, South Africa

This research presents a deep learning approach to determine pineapple size from images, to identify the instances of pineapples, and subsequently to extract fruit dimensions. This was achieved by first detecting pineapples in each image using a Mask region-based convolutional neural network (Mask R-CNN), and then extracting the pixel diameter and length measurements and the projected areas from the detected mask outputs. Various Mask R-CNNs were considered for the task of pineapple detection. The best-performing detector (Model 4: COCO Fliplr Res50) made use of MS COCO starting weights, a ResNet50 CNN backbone, and horizontal flipping data augmentation during the training process. This model achieved a validation AP@[0.5:0.05:0.95] of 0.914 and a test AP@[0.5:0.05:0.95] of 0.901, and was used to predict masks for an unseen data set containing images of pre-measured pineapples. The distributions of measurements extracted from the detected masks were compared to those of the manual measurements using two-sample Z-tests and Kolmogorov–Smirnov tests. There was sufficient similarity between the distributions, and it was therefore established that the reported method is appropriate for pineapple size determination in this context.

Significance:

- The methods we applied are traditional CNN models. Our main contribution is testing the application of different starting weights with multiple augmentation techniques used under different CNN backbones, ultimately comparing seven different model specifications of Mask R-CNNs.
- We have validated the results against manually measured fruits using statistical techniques and show that fruit size can be confidently determined from images instead of manual measurement, which is very labour and time intensive.
- The method developed is currently being used within a factory.

Introduction

In recent years, there has been an increased interest in the use of machine vision approaches in fruit industries.[1,2] Object detection has played a crucial role in precision agriculture, enabling automated fruit counting, disease detection and yield estimation. Recent advancements in deep-learning-based object detection methods have significantly improved accuracy and efficiency in agricultural settings. Our study focused on the use of machine vision systems for fruit detection and measurement, particularly those making use of convolutional neural networks (CNNs).[1]

The development of a non-destructive pineapple fruit size determination approach based on images would be valuable in allowing the factory to obtain representative size data.[1] This, in turn, would allow for a better understanding of the relationship between fruit size and juicing efficiency, which would better position factories to incentivise farmers to deliver optimally sized fruit by including a size-related factor into the pricing scheme.[1]

Fruit size is described by weight, or by some dimensional parameter such as length, diameter, volume, circumference, projected area, or some combination of these.[1,3,4] In-field, size determination allows for yield prediction[5,6] and can be used as a parameter in predicting optimum harvest time[1,4]. However, there is also a need for size determination post-harvest.[1] Post-harvest size determination is important in several contexts, such as sorting fresh market fruits into size groups for packaging purposes, and for assigning prices.[1,4] In the context of this work, the data were collected from a pineapple juicing factory in the Eastern Cape of South Africa, where size determination is not linked to any sorting activity, as all fruits are homogenised in the juicing process.[1] However, the first processing step involves peeling of the pineapples, and it should be noted that the peelers remove a set width of peel from each pineapple, regardless of fruit size.[1] Theoretically, this means that there is greater wastage with small fruits, as a larger percentage of the fruit is removed by the peeler.[1] Hence, larger pineapple fruits with a lower ratio of peel to flesh are expected to have a higher juice yield per kilogram of fruit. In the context of this work, size determination is a necessary step towards being able to quantify the relationship between fruit size and juice yield.[1]

Considering the main aims of this study, there were two tasks that needed to be accomplished.[1] The first task involved the training and evaluation of a pineapple detector, while the second task involved determining the fruit size based on the object mask output of the detector.[1] Given these two tasks, two separate image data sets were used in this work.[1]

Methodology

The first task involved pineapple detection from images – an object detection problem. The goal of object detection is to locate all object instances in an image from a list of predefined classes.[1,7] Hence, the problem involved both predicting the category and drawing a bounding box around each object in an image.[1] A bounding box is the minimum-sized rectangular border that fully encloses an object and indicates its location within an image.[1] Figure 1 illustrates the bounding box coordinates (x,y,w,h) of the object of interest in an image, including the (x, y)

coordinates of the centre of the box, as well as the height (h) and width (w) of the box.[1] By convention, the top left corner of the image has (*x, y*) coordinates of (0, 0).[1]

While classification and localisation problems tend to have only one object per image, object detection problems may have multiple objects and multiple classes represented in a single image.[1,8] If one of those objects appears in a given input image, an object detector should draw a bounding box around the object and predict the category of the object.[1,7] In this study, we had multiple objects from a single class.

The methods of object detection were based on CNNs that work particularly well for processing data with a grid-like topology such as images[1,9], where the same feature needs to be detected in multiple places within the grid[1,10]. Just like a fully connected feed-forward neural network, CNNs consist of an input layer, several hidden layers and an output layer, which are shown in stages in Figure 2. Colour images of dimensions ($n_H^{[0]} \times n_W^{[0]} \times n_C^{[0]}$), where *H* and *W* denote height and width, and *C* represents the red, green and blue (RGB) channels, are input into the network ($l = 0, .., L$ the layer is denoted with a superscript [0]). Fully connected feed-forward neural networks are not well suited to image classification, because an image of size 970 x 605 RGB has around 1.76 million pixel values as inputs to the network, which results in a large number of weights to be optimised, especially the weights from the input to the next layer. This process requires a very large data set with known classes, which is very impractical for an image classification problem. Moreover, such fully connected neural networks are prone to overfitting, which needs to be avoided by strict regularisation. CNNs, on the other hand, take advantage of image characteristics, such as spatial connectivity, to improve the overfitting and regularisation issues of fully connected feed-forward neural networks. The images pass through a series of sequential processing steps in the hidden layers[1,11], which are also called convolutional and pooling layers and perform feature extraction, while the fully connected layer is responsible for classification[1,12].

Convolutional layer

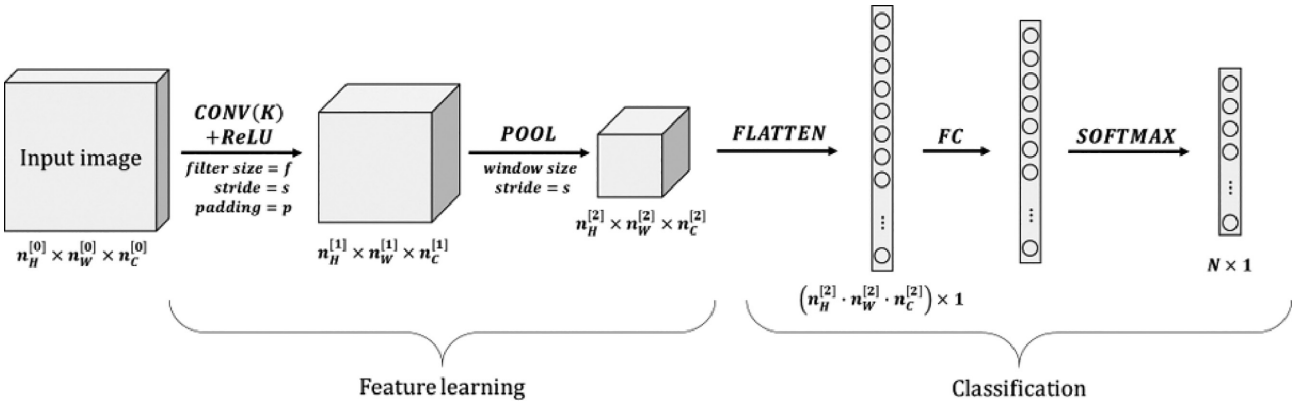
The convolution of the input image is implemented by using *K* filters of size $f \times f \times n_C^{[l]}$, where *f* is the height and width of the filters, which are smaller than the input image. While filter weights can be predefined by the researcher, they are often the hyperparameters which are learned using back propagation. After the filtering, padding of zeros is applied to the input image by appending *p* rows and columns to retain the original image size and to prevent the width and the height of the output from shrinking at each layer. Finally, stride (*s*), the number of horizontal or vertical pixel steps a filter makes over the input matrix, is used to reduce the number of parameters learned. After this process of the convolution of an input image, the linear activations in the feature map are passed through a non-linear activation. Typical activation functions are sigmoid, hyperbolic tangent (tanh), rectified linear activation (ReLU) and Leaky ReLU functions used to solve the problem of vanishing gradients.[9] The training time for the ReLU activation function has been found to be much shorter than those for the sigmoid and tanh functions.[13]

Pooling layer

The height and the width of the output of the convolutional layer are reduced by the pooling layer in all the channels independently, by aggregating the low-level features over a small neighbourhood defined by the window size and the stride, which are the predefined parameters of the pooling process. The main purpose of the pooling layer is to speed up computation as well as to prevent overfitting. The most frequently used pooling functions are maximum and average pooling functions.[1] The representation is then flattened into a vector before the fully connected layer.[1]

Source: ©2021 Harris[1] (reproduced with permission)

Figure 1: An image containing an object of interest (left), and the same image with a bounding box drawn around the object (right).



FC, fully connected

Figure 2: Structure of a convolutional neural network, showing the different types of layers and the dimensions of the data at each stage. Inspired by Gu et al.[14]

Fully connected layer

The elements of the flattened vector are the nodes in one layer which are fully connected to the next layer. As in the fully connected neural networks, matrix multiplication is applied to the flattened vector.

Softmax layer

Finally, in the case of a classification task, a softmax layer with N nodes is used to classify images into one of N categories.[1,13]

The weights in the trainable filters are optimised using a backpropagation method which includes forward and backward propagation and weight update stages. The weights are initialised, generating a predicted class which is then compared to the actual classes using a loss function. The derivative of the loss function for each weight is computed via the chain rule to guide the weight update using gradient descent or other variations of gradient descent such as stochastic gradient descent, root mean square propagation (RMSProp), and adaptive moment estimation (Adam optimiser). Irfan et al.[15] showed that Adam optimiser is superior in a classification problem, followed by RMSProp and stochastic gradient descent. The ultimate goal is to find the optimal weights that minimise the loss function. The process is repeated until a predefined number of iterations is reached or the change in loss function is negligible.

In this study, the number of objects per image was not known beforehand; therefore, a standard CNN could not be used for object detection, as it would have had a fixed-length output.[1] However, the CNNs can be adapted, and therefore used for tasks other than classification. One such task is object detection.[1] While classification identifies a single main object in an image, object detection and instance segmentation both involve identifying all individual objects of interest in an image containing multiple objects.[1] In object detection, objects are classified and localised using bounding boxes, whereas in instance segmentation, all pixels belonging to an object are identified.[1]

Object detection frameworks

There are two major types of detection frameworks: region-based (two-stage) and unified (one-stage) frameworks.[1,7] Region-based frameworks (R-CNN, Fast R-CNN (RFCNN), Faster R-CNN, Mask R-CNN, Feature Pyramid Network (FPN), Relation Networks, Faster R-CNN+++, Deformable RFCNN, Cascade R-CNN, DCNv2+Faster R-CNN, PANet[15]) are referred to as two-stage frameworks as they involve first generating category-independent region proposals before applying a classifier to determine the category labels of the proposed regions.[1] In contrast, one-stage detectors – You Only Look Once (YOLO)[14,16-18] and its versions (YOLOv2-v8)[19], single shot detection (SSD)[20,21], Deconvolutional Single Shot Detector (DSSD), RetinaNet, M2Det, DCN, DCNv2, NAS-FPN, CornerNet512, EfficientDet and Fast-D – feed forward in a single pass[1]. As such, one-stage frameworks are often faster and more suited to online processing. Several review papers have been published that compare object detection frameworks in terms of advantages and disadvantages.[20,22] However, an earlier study showed that one-stage frameworks had lower accuracy than two-stage frameworks.[23]

In our study, fruit detection and size determination were required for fruit quality evaluation. However, as no online sorting was required, a higher accuracy was desirable. As such, we used a two-stage detector. One of the earlier two-stage detectors is region-based CNN (R-CNN) that combines region proposals with CNNs.[8,24] R-CNNs use a region proposal network to extract around 2000 region proposals from an input image. The rectangular window around each region is warped to 227 x 227 pixels before a large CNN is used to compute features for each region.[1] Finally, each region is classified using class-specific linear support vector machines, and bounding box regression is used to update the bounding box coordinates.[1]

Another commonly used two-stage detector – Fast R-CNN – arose from extending R-CNN. With this detector, instead of processing each region of interest separately, the whole image is run through the CNN, resulting in a convolutional feature map that corresponds to the whole image.[1,25] Fast R-CNN processes the entire input image with a CNN to produce a feature map. Region proposals are projected onto the feature map and resized using a region of interest pooling layer. Each region of interest feature vector is passed through fully connected layers before branching into two output layers: one a softmax classifier and another that provides class-specific updated bounding box coordinates.[1]

While all the object detection methods discussed up to this point identify located objects using a bounding box, Mask R-CNN takes it a step further by predicting a pixel-level object mask indicating the location of each object.[1,26] This is done by including a branch, in parallel to the one that performs bounding box regression and classification, which outputs a binary mask for each region of interest. As our main aim was to detect fruit size, we used Mask R-CNN, which predicts a pixel-level object mask indicating the location of each object.[1,26]

Fruit detection and classification

Several studies have applied CNN variants to detect occluded fruits, fruits on trees and fruits under low-light conditions, with the majority of studies focusing on apples, citrus and tomatoes. Fu et al.[27] employed two Faster R-CNN based architectures (ZFNet and VGG16) for apple detection on trees with and without background trees. An average precision (AP) of 0.893 was achieved with VGG16 on the original images, and removing background trees improved the AP values. Fuentes et al.[28] used SSD MobileNet to detect seedlings using data from six crops and four types of trays. The accuracy results for different types of crops and trays ranged from 57% to 96%. A study by Yang et al.[29] proposed the BCo-YOLOv5 network model for fruit detection in orchards, with extensive experiments on citrus, apple and grape detection. Using YOLOv5s as the base model, the bidirectional cross-attention mechanism is integrated between the backbone and neck networks to strengthen feature relationships and to improve the low accuracy of fruit detection when the fruit is covered by leaves or the fruit target is too small. Yang et al.[29] showed that the adapted method of the BCo-YOLOv5 network can effectively detect citrus, apple and grape targets in fruit images, with a 97.70% mean AP after training and testing. A survey study by Espinoza et al.[30] outlined the different aspects of deep learning in the analysis of fruit images.

Research on the pineapple fruit industry has included crown detection, detection of ripe pineapples, fruit classification and pineapple quality detection. Cuong et al.[31] used an improved version of the YOLO-v4 algorithm to determine when the pineapples were ripe. They used 5 000 000 pineapples harvested from a pineapple farm in Vietnam's central region and achieved a 98.26% accuracy. Syazwani et al.[32] used artificial neural networks and various machine-learning classifiers to identify and recognise the pineapple's crown images and to count the fruits obtained from aerial images. They showed that an effective image analysis of the pineapples could be achieved with high accuracy (94.4%). Kanjanawattana et al.[33] demonstrated the use of AlexNet to classify pineapples into four different sweetness levels from photos and achieved an accuracy of 91.78% and an F1 score of 92.31%. Liu et al.[34] proposed an improved YOLO-v3 model by adding modifications on the backbone network and compared the detection result of this method against YOLOv3, Faster-RCNN and Mobilenet-SSD, showing that the improved YOLOv3 model performed the best. Chang et al.[35] proposed the use of YOLO-v2 for detecting harvestable (ripe and ripening) and unharvestable (unripe) pineapples from single pineapple image data collected in the field for offline training. The proposed method achieved an accuracy of 95%. Most of these studies achieved high accuracy rates with different detectors and backbones used. These studies show that YOLO and its versions are the most commonly tested algorithms in fruit detection problems with varying environments and lighting conditions. Despite advancements, object detection in agriculture still faces the challenges of high computational cost and limited generalisation to new environments due to data set related differences. Apart from these, the lack of explainability and the high cost of implementation of these models limit the use of these deep learning methods by farmers. In recent years, we have seen many advancements in improving pineapple detection and classification methods, such as the integration of RGB images with thermal imagery with the use of unmanned aerial vehicles that can collect RGB, thermal and hyperspectral data.[36] The high computational cost is dealt with through the use of lightweight models such as YOLO-v8 for mobile deployment. Furthermore, there have been attempts to remove the domain dependency by knowledge transfer from different fruit data

sets. In light of these advances, there are still very limited use cases in South Africa in the pineapple industry. The aim of this research was to raise awareness of the accurate use of these automated systems.

Data

The image data used in this study came from a pineapple juicing factory in the Eastern Cape of South Africa, where two conveyor belts carry the fresh pineapples into the factory for processing.[1] Video footage was collected during March–June 2020 using two progressive scan CMOS cameras (Hikvision DS-2CD2145FWD-I(S)) located above the two conveyor belts (Camera A and Camera B) delivering pineapples to the factory and positioned directly overhead and parallel to the belts to prevent perspective distortion.[1] VLC Media Player[37] was used to extract images from the video footage and the OpenCV library[38] was used to crop the images to a size of 970 x 605 pixels[1]. In total, 160 images of size 970 x 605 pixels were extracted from the video footage.

These 160 images were randomly split into training, validation and test sets, ensuring that each set contained an equal number of images from each camera.[1] A 70/20/10 per cent split was chosen for the training, validation and test sets.[1] Due to resource limitations, only one validation set was used, rather than cross-validation. All images in the training set contained at least one fruit, and a total of 2138 pineapples were then manually labelled using the open-source VGG Image Annotator[39] – a tool used to draw polygon shapes around objects (Figure 3)[1].

In all cases, the polygon tool was used at 2.5x level zoom. VGG Image Annotator[39] is a convenient tool as it runs in a web browser and does not require any installation or setup. The final output is a json file with the coordinates of each polygon shape. These polygon shapes are used to define the pixel-level binary masks required for training a Mask R-CNN: points located within the polygon are set to a value of one, indicating that an object is present at that location in the image, while pixels outside the polygon are set to zero.[1]

The number of pineapples per image varied between 2 and 30, with the number of pineapples per image being fairly similar throughout the three subsets.[1] As the main aim of this research was to determine pineapple fruit size from images, it was necessary to assess whether the fruit size obtained from images was sufficiently similar to the fruit size obtained by manual measurement of each pineapple.[1] Hence, for evaluation of the size determination approach, we obtained images of 120 pineapples that had previously been manually measured.[1] The 120 pineapples had been pre-measured manually using a calliper system. The length and width of each fruit were recorded, as well as the weight, which was measured using a balance (CAS PR PLUS, accurate to 0.002 kg) for further evaluation.[1]

Implementation

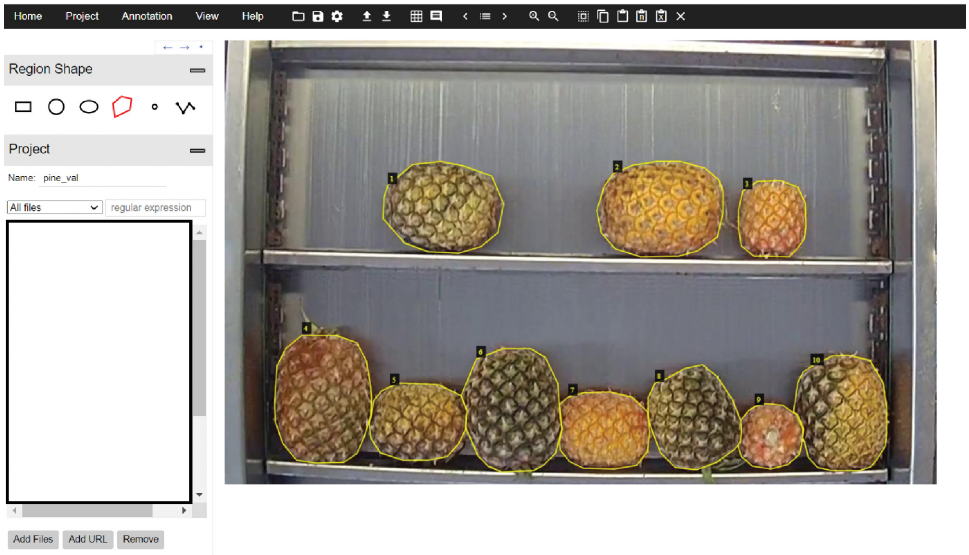
Several instance segmentation models were considered in this work. The pineapple detection performance was assessed with respect to transfer learning using pretrained COCO and ImageNet weights, the choice of CNN backbone and data augmentation techniques.[1]

Transfer learning using pretrained COCO and ImageNet weights

In computer vision tasks, it has become common practice to pretrain a CNN on a large data set and then transfer the learned features to a new task that has a smaller number of labelled examples[40] and fine tune it for the new task.[1] In many cases, starting weights are readily available for different network architectures, particularly those pretrained on the ImageNet[41] and MS COCO[42] databases, which are commonly used in image recognition tasks[1]. ImageNet, for example, contains 1000 object classes and more than 1.2 million images[43], while MS COCO contains 1.5 million object instances from more than 330k images[1,44]. Neural network architectures that perform well on a given computer vision task often perform well on other computer vision tasks, as many of the features learned during training, particularly lower-level features, are applicable across multiple applications.[1] Tuning an existing architecture using its pretrained weights instead of random weight initialisations can reduce the time and computation requirements.[1] In this study, these pretrained weights were used to initialise the parameters of the network before training on a new task. While low-level features are likely to be similar across different data sets, higher-level features learned by deeper layers in the CNN network are more specific to the task at hand.[1,40,45] Therefore, the performance of an object detector is affected by the similarity of the new target classes to the base classes used for pretraining.[1,45] In this study, we initially investigated the performance of the models with COCO and ImageNet weights and thereafter decided which weights to use with different suitable networks (backbones).[1]

Choice of CNN backbone

A CNN backbone is the feature extraction part of the network which takes an input image and extracts features before passing them to later layers. There are various types of backbones specific to object detection and instance segmentation, such as AlexNet, VGGs, ResNets, Inceptions, Xception, DenseNets, Inception-ResNet, ResNeXt, SqueezeNet, MobileNet, EfficientNet, RegNet, and many others.[46,47] Differences, and advantages and disadvantages of each network are detailed in Elharrouss et al.[46] The choice of CNN backbone is mainly decided based on the Matterport Mask



Source: ©2021 Harris[1] (reproduced with permission)

Figure 3: Screenshot showing annotation using the open source VGG Image Annotator tool.[39] Polygons were drawn around each pineapple visible in the image.[1] Two sections of the conveyor belt are visible in each frame.[1]

R-CNN[48] implementation[1]. In this work, ResNet-50 and ResNet-101 CNN backbones were chosen.[1] A ResNet is a residual network, characterised by residual blocks containing shortcut connections that perform identity mapping.[1,49] Prior to the introduction of ResNet, very deep networks suffered from a problem of vanishing gradients, which resulted in deeper networks having higher training errors than their shallower counterparts.[1] However, the introduction of residual blocks ensures that information from earlier layers is retained.[1,49] ResNet-50 and ResNet-101 have similar structures, with both employing the concept of residual blocks; however, ResNet-50 has 50 layers, while ResNet-101 has 101 layers.[1] The models initialised with COCO weights were implemented with ResNet-50 and ResNet-101 backbones. Thereafter, data augmentation techniques were applied to the chosen weights and CNN backbone.[1]

Data augmentation techniques

Data augmentation techniques are applied in cases in which not much labelled training data is available.[1] This is a common method to avoid overfitting on the available data to artificially enlarge the data set, and the variability thereof, using label preserving transformations[13,40] and can be performed using a Python library such as *imgaug* (image augmentation for machine learning experiments).[1] Frequently used augmentation techniques include horizontal mirroring, random cropping, rescaling and colour shifting.[1,40] Distortions to the colour channel help the model become more resistant to changes in illumination.[1] While it is possible to use data augmentation to expand the data set with copies of the augmented versions, it is preferable to randomly augment the data with each training epoch.[1,40] We investigated the effect of Gaussian blurring and noise, horizontal flipping and colour shifting augmentation techniques[1] and discuss the performance of the different models considered using validation and test sets in the next section.

Results and discussion

The aim of the first task in this study was to train an instance segmentation model to effectively localise pineapples within images.[1] To this end, several Mask R-CNN models, as described in Table 1, were considered and were assessed using several evaluation metrics such as multitask loss, intersection over union (IoU) and AP.

During the training of Mask R-CNN, a multitask loss is defined for each region of interest. The multitask loss (Equation 1) is the combination of the losses associated with the tasks of classification (L_{cls}), localisation (L_{bbox}) and instance segmentation (L_{mask})[1,25]:

$$L = L_{cls} + L_{bbox} + L_{mask} \quad \text{Equation 1}$$

IoU is another metric that can be used to describe how similar the predicted bounding box is to the groundtruth bounding box. IoU is defined as the ratio between the intersection ($\cap$) and union ($\cup$) of the predicted bounding box (B_P) and the groundtruth bounding box (B_{GT}), as shown in Equation 2[1,50,51]:

$$IOU = \frac{Area\{B_P \cap B_{GT}\}}{Area\{B_P \cup B_{GT}\}} \quad \text{Equation 2}$$

where $0 \leq \text{IoU} \leq 1$. The IoU is essentially a measure of the overlap of the predicted bounding box and the groundtruth bounding box.[1] A perfect detection would have an IoU of one, while a predicted bounding box that does not overlap at all with the groundtruth bounding box will have an IoU of zero.[1] The IoU is used to determine whether a detection is a true positive or false positive.[51] A predicted bounding box is considered to be a true positive if the IoU is greater than some user-defined threshold, usually at least 0.5.[1,50] For a given IoU threshold, the true positive (TP) and false positive (FP) predictions can be used to determine the recall (Equation 3) and precision (Equation 4)[1]:

$$Recall = \frac{TP}{TP + FN} \quad \text{Equation 3}$$

$$Precision = \frac{TP}{TP + FP} \quad \text{Equation 4}$$

Recall is the true positive rate divided by the ratio of true positive predictions to the number of actual (groundtruth) positives, while precision is a measure of how many of the positive predictions are true positives.[1] Each predicted bounding box has an associated confidence level, which can be used to rank the output.[1,50] A precision/recall curve can then be computed from the ranked output.[1,50] The AP summarises the shape of the precision/recall curve.[1,50] The MS COCO Benchmark challenge averages the AP over a range of IoU values, from 0.50 to 0.95 at intervals of 0.05, denoted as AP@[0.50:0.05:0.95]. This rewards detectors with better localisation. In this work, the MS COCO metric was used to evaluate the performance of object detectors.[1]

To investigate the effect of employing different CNN backbones, a combination of the previous implementation structures was investigated.[1] The setup of different models is provided in Table 1.

The first three models in Table 1 have no augmentation applied to the data sets. The main decision is based on starting weights and CNN backbone. The training and validation losses associated with Model 2, initialised with COCO pretrained weights, are considerably lower than those of Model 1, which was initialised with ImageNet weights.[1] This may be due, in part, to the fact that the ImageNet data set has an average of only 1.5 objects per image, while the COCO data set has an average of 7.3 objects per image, which is more comparable to the average number of objects per image in the pineapple data set (an average of 13.4 pineapples per image).[1]

The AP performance of the two models on the validation set is summarised in Table 2.[1] Both models have a ResNet101 CNN backbone without data augmentation and performed well in terms of AP@0.5[1]. However, the COCO-initialised model, COCO NoAug Res101,

Table 1: Different model specifications for the investigation of different convolutional neural network (CNN) backbones[1]

Model #	Model name	Starting weights	CNN backbone	Data augmentation
1	ImageNet_NoAug_Res101	ImageNet	Res101	None
2	COCO_NoAug_Res101	COCO	Res101	None
3	COCO_NoAug_Res50	COCO	Res50	None
4	COCO_Fliplr_Res50	COCO	Res50	Horizontal flip
5	COCO_GaussNB_Res50	COCO	Res50	Gaussian noise and blur
6	COCO_Colour_Res50	COCO	Res50	Lightening and darkening
7	COCO_All_Res50	COCO	Res50	All of the above

Table 2: Validation average precision (AP) summary of two Mask region-based convolutional neural networks (R-CNNs) for pineapple detection trained using different transfer learning procedures.[1] Model 1 was initialised with ImageNet starting weights, while Model 2 was initialised with MS COCO starting weights.[1]

Model #	Model name	Starting weights	# Epochs	Validation AP	
				IoU 0.5	IoU 0.50:0.05:0.95
1	ImageNet_NoAug_Res101	ImageNet	30	1.000	0.860
2	COCO_NoAug_Res101	COCO	30	0.998	0.892

IoU, intersection over union

Table 3: Validation average precision (AP) summary of two Mask region-based convolutional neural networks (R-CNNs) for pineapple detection with different CNN backbones.[1] Model 2 has a ResNet101 backbone, while Model 3 makes use of ResNet50 architecture.[1] Both models were initialised with MS COCO starting weights and did not employ data augmentation.[1]

Model #	Model name	CNN backbone	# Epochs	Validation AP	
				IoU 0.5	IoU 0.50:0.05:0.95
2	COCO_NoAug_Res101	ResNet101	30	0.998	0.892
3	COCO_NoAug_Res50	ResNet50		1.000	0.884

IoU, intersection over union

achieved an AP@[0.50:0.05:0.95] value of 0.892, compared to the ImageNet-initialised model, ImageNet NoAug Res101, which achieved an AP@[0.50:0.05:0.95] value of 0.860.[1] As the MS COCO features were better suited to the task of identifying pineapples, these were chosen as starting weights for all subsequent networks.[1]

A comparison of the AP performance of Model 2 (COCO NoAug Res101) and Model 3 (COCO NoAug Res50) (Table 3) shows that both models performed well in terms of AP@0.5.[1] Interestingly, Model 3 (COCO NoAug Res50) performed almost as well as its larger counterpart in terms of AP@[0.50:0.05:0.95] (Table 3), achieving a value of 0.884, compared to Model 2 (COCO NoAug Res101), which achieved an AP@[0.50:0.05:0.95] value of 0.892.[1] As Model 3, with its smaller CNN backbone, had comparable performance to that of the larger network, it was chosen for subsequent steps in this work.[1] The remainder of the models considered were initialised using MS COCO starting weights and a Res50 CNN backbone with different augmentation techniques, such as horizontal flip, Gaussian noise and blur, lightening and darkening, and all the augmentation methods applied together. The use of data augmentation resulted in a decrease in both training and validation set losses, although all augmentation strategies appear to have a similar performance (Figure 4).[1]

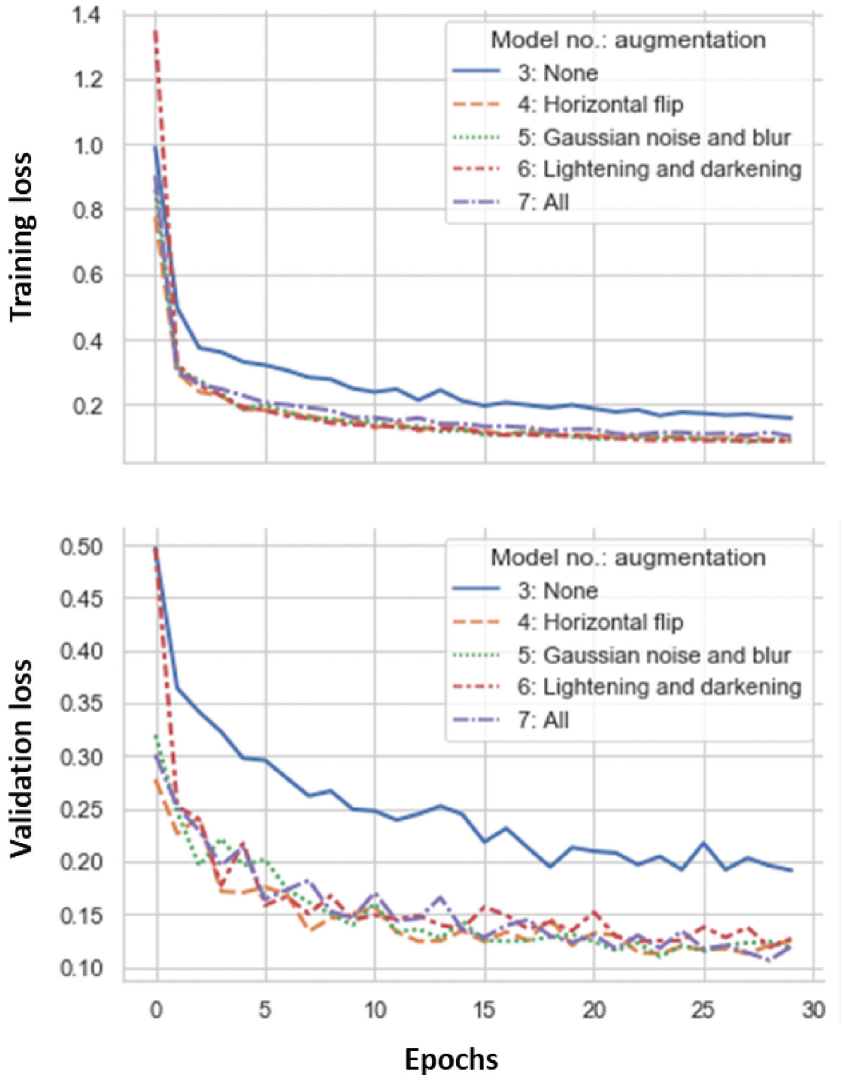
A summary of the average training and validation set losses in Table 4 shows that Model 4 (COCO Fliplr Res50) had the lowest training loss and validation loss, on average.[1] All models had a ResNet50 CNN backbone and were initialised with MS COCO starting weights.[1] The average training and validation losses were calculated across all 30 epochs.[1] For each model, the minimum validation loss is reported, together with the epoch in which the minimum value was achieved.[1]

Table 5 confirms that Model 4 had the best performance, with an AP@[0.50:0.05:0.95] of 0.914.[1] Model 3 (COCO NoAug Res50) was considered the best model that did not make use of data augmentation, while Model 4 (COCO Fliplr Res50) was identified as the overall best pineapple detector considered in this work, based on validation AP values.[1] As such, the performance of these two models on the withheld test set was evaluated.[1] The pineapple detector Model 3 (COCO NoAug Res50) achieved a test AP@0.50 of 0.997 and a test AP@[0.50:0.05:0.95] of 0.874.[1] However, as expected, Model 4 (COCO Fliplr Res50) improved upon this, achieving a test AP@[0.50:0.05:0.95] of 0.901.[1]

As Model 4 (COCO Fliplr Res50) was determined to be the best pineapple detector considered in this work, it was the model chosen for determining the size of pineapples from images.[1] As such, Model 4 (COCO Fliplr Res50) was used to predict masks for pineapples in the previously unseen data set of pineapple images to post-validate the size determination approach.[1] Two approaches to size determination were employed.[1] The first approach involved extracting the diameter and length of pineapples from the predicted masks. In this approach, the Mask R-CNN trained to identify pineapples was used to predict masks for pineapples in the previously unseen data set of pineapple images.[1] The model outputs a binary mask for each detected object and OpenCV library's[38] findContours() function was first used to extract the coordinates of the points describing the polygon outline of the binary mask.[1] After the coordinate extraction, the minAreaRect() function was used to find the diameter and length measurements of each fruit by fitting a minimum area rotated rectangle to the mask.[1] The dimensions obtained by manual measurement using callipers were compared to the dimensions extracted from the predicted object masks by visual inspection and by performing a Z-test and a two-sample Kolmogorov–Smirnov test.[1] Beyond visual assessments which revealed similarity, for both the fruit diameter and fruit length, two-sample Z-tests were used to indicate whether the mean of the dimensions obtained by manual measurement was equal to the mean of the dimensions extracted from the predicted object masks.[1] The two-sample Z-tests for both the mean diameter and mean length had very high p-values ($p >> 0.10$), indicating that there is not enough evidence to suggest that the two population means are different.[1]

To further investigate whether the distribution of predicted fruit dimensions was the same as the distribution of actual, hand-measured dimensions, we considered the empirical cumulative distribution functions shown in Figure 5[1]. Visually, the empirical cumulative distribution functions for the predicted measurements seem similar to those of the actual measurements.[1]

A two-sample Kolmogorov–Smirnov test was performed to compare the measured and the predicted distributions of both the pineapple diameter and the length measurements. The two-sample Kolmogorov–Smirnov test showed that the distributions of the measured and predicted **diameters** were found to have a Kolmogorov–Smirnov value of 0.083, and a p-value of 0.801.[1] As the Kolmogorov–Smirnov test had a high p-value ($p >> 0.10$), there is not enough evidence to reject the null hypothesis that the two distributions are equal.[1] According to this test, the difference between the two distributions is not significant enough to say that they are explicitly different.[1] The two-sample Kolmogorov–Smirnov test showed that the distributions of the measured and predicted **lengths** were found to have a Kolmogorov–Smirnov value of 0.058, and a p-value of 0.987.[1] The high p-value indicates that there is not enough



Source: ©2021 Harris[1] (reproduced with permission)

Figure 4: Training and validation set losses for Mask region-based convolutional neural networks (R-CNNs) employing different data augmentation strategies.[1] Training losses are shown in the top panel, while validation losses are shown in the bottom panel. Models 4–7 were trained using data augmentation and had lower training and validation losses than Model 3, which did not make use of data augmentation.[1]

Table 4: Summary of training and validation set losses for Mask region-based convolutional neural networks (R-CNNs) using different data augmentation strategies[1]

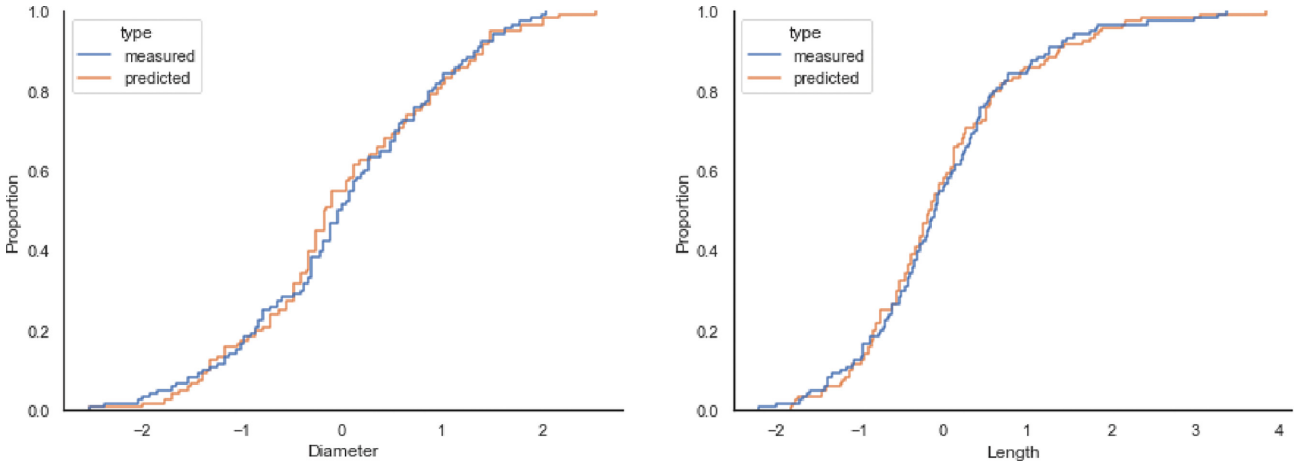
Model #	Model name	Data augmentation	Average training loss	Average validation loss
3	COCO_NoAug_Res50	None	0.261	0.25
4	COCO_Fliplr_Res50	Horizontal flip	0.158	0.146
5	COCO_GaussNB_Res50	Gaussian noise and blur	0.161	0.151
6	COCO_Colour_Res50	Lightening and darkening	0.174	0.164
7	COCO_All_Res50	All the above augmentations	0.179	0.155

evidence to reject the null hypothesis that there is no difference between the two distributions.[1] Therefore, from both tests we conclude that the measured and predicted diameters and lengths are similar.

The second approach to size determination involved finding the projected area of each fruit.[1] As the mask output of the Mask R-CNN model contains a binary mask for each object instance, this can be achieved by summing pixels over each mask layer.[1] To evaluate this method, the pixel area of the detected mask was compared to the pixel area of the hand-labelled groundtruth mask.[1] As the groundtruth mask is labelled by a human using VGG Image Annotator[39], this method gives an indication of the best human performance in terms of determining fruit size from images.[1] Moreover, as the predicted masks are being compared to the groundtruth masks rather than the actual fruit measurements, differences in metrics should not arise due to the orientation problem when some pineapples stand upright.[1] In these cases, the longitudinal

Table 5: Validation average precision (AP) summary of Mask region-based convolutional neural network (R-CNN) pineapple detectors employing different data augmentation strategies during training.[1] All models had a ResNet50 CNN backbone and were initialised with MS COCO starting weights. For each model, the weights associated with the epoch with lowest validation loss were used to determine the AP.[1]

Model #	Model name	Data augmentation	Minimum validation loss	Validation AP	
				IoU 0.5	IoU 0.50:0.05:0.95
3	COCO_NoAug_Res50	None	0.192	1.000	0.884
4	COCO_Fliplr_Res50	Horizontal flip	0.113	1.000	0.914
5	COCO_GaussNB_Res50	Gaussian noise and blur	0.110	1.000	0.905
6	COCO_Colour_Res50	Lightening & darkening	0.119	1.000	0.898
7	COCO_All_Res50	All of the above	0.107	1.000	0.898



Source: ©2021 Harris[1] (reproduced with permission)

Figure 5: Empirical cumulative distribution functions for the scaled diameter (left) and scaled length (right).[1] The dimensions obtained by manual measurement using callipers are shown by the blue lines, while the dimensions obtained from the predicted object masks are shown in orange.[1]

section of the pineapple is not visible, resulting in underestimation of length measurements.[1] Hence, in these kinds of images, comparison of the projected areas is useful, as the groundtruth area gives an indication of what the human performance would be in this task of determining fruit size from images.[1]

Similar tests have been applied to projected area comparisons and the two-sample Z-test for projected area had a fairly high p-value ($p > 0.10$), indicating that there is not enough evidence to reject the null hypothesis that the two population mean areas are equal.[1] A two-sample Kolmogorov–Smirnov test showed that the distributions of the groundtruth and predicted projected areas were found to have a very small Kolmogorov–Smirnov value (0.117) with a high p-value (0.389)[1]. As the p-value is large, there is no strong evidence to indicate that the two distributions are not the same. The results achieved in this study were satisfactory; however, there were a few limitations, as discussed below.[1]

Summary and conclusion

We have presented an approach[1] to determine pineapple size from images, using Mask R-CNN to identify the instances of pineapples and subsequently extract fruit dimensions using the OpenCV library[38]. Several Mask R-CNNs were trained on the pineapple data set. Analysis of transfer learning showed that Model 2 (COCO NoAug Res101), initialised with MS COCO features, performed better than Model 1 (Imagenet NoAug Res101), which was initialised with ImageNet weights.[1] Both ResNet101 and ResNet50 CNN backbones were considered for the Mask R-CNN pineapple detector.[1] It was found that Model 3, with the smaller ResNet50 backbone, performed almost as well as its larger counterpart.[1] The pineapple detector Model 3 (COCO NoAug Res50) achieved a satisfactory performance, with a validation AP@[0.50:0.05:0.95] of 0.884 and a test AP@[0.50:0.05:0.95] of 0.874.[1] This performance was, however, improved upon by including data augmentation.[1] Model 4, which made use of horizontal flipping during the training process (COCO Fliplr Res50), achieved a validation AP@[0.50:0.05:0.95] of 0.914 and a test AP@[0.50:0.05:0.95] of 0.901.[1] Model 4 (COCO Fliplr Res50), having been identified as the best performing detector, was then used to predict masks for pineapples in the previously unseen data set of pineapple images, to post-validate the size determination approach.[1] The distributions of the predicted fruit dimensions were found to be equal to the manually measured fruits using two-sample Kolmogorov–Smirnov tests.[1] It was, therefore, established that this method is appropriate for pineapple size determination, in this context.[1]

While the results achieved in this work were satisfactory, there were a few limitations.[1] Firstly, cameras were not installed at the same height as each other from the conveyor.[1] If these heights were adjusted, the size determination accuracy could be increased.[1] Secondly, a relatively small data set was used to train the Mask R-CNNs to detect pineapples.[1] Additionally, these data were acquired using convenience sampling during a short period of time, due to delays associated with the start of the COVID-19 pandemic in 2020.[1] While convenience sampling was used in this work, future work could employ an experimental design approach, incorporating data from different seasons, as well as night shifts.[1] Utilising training data obtained throughout the year might provide a detector that is more robust to variation in seasonal colour changes.[1] Finally, the training data could be extended to include different lighting conditions and examples of foreign objects that are sometimes deposited onto the conveyor belts along with the fruit.[1] Foreign objects may damage the peeler, resulting in downtime

that negatively affects operating efficiency.[1] If this could be implemented on a real-time basis, an auto-stop function could be incorporated to avoid damage to equipment.[1] In future work, the size data obtained from images could be used in conjunction with information about the growing conditions of the pineapple plants to better understand the factors that affect fruit size, and to allow for more accurate yield predictions.[1]

Acknowledgements

We thank the factory Summerpride Foods (Pty) Ltd. in East London, South Africa, for installing the cameras and for their assistance throughout the data collection process. We also thank the reviewers for their invaluable insights, corrections and contributions that hugely improved the paper.

Data availability

The codes and the data sets generated and/or analysed during the current study are available in the 'measure-pineapple' Github repository: https://github.com/Jess-cah/measure-pineapple.

Declarations

We have no competing interests to declare. We have no AI or LLM use to declare. This paper is based on a published master's thesis[1].

Authors' contributions

J.H.: Conceptualisation, methodology, investigation, sample analysis, formal analysis, validation, data curation, project leadership, project administration, writing – original draft, writing – review and editing. S.E.: Supervision, project leadership, project administration, writing – original draft, writing – review and editing. Both authors read and approved the final manuscript.

References

1. Harris J. Object detection and size determination of pineapple fruit at a juicing factory [master's thesis]. Cape Town: University of Cape Town; 2021. https://open.uct.ac.za/handle/11427/35596
2. Naranjo-Torres J, Mora M, Hernandez-Garcia R, Barrientos RJ, Fredes C, Valenzuela A. A review of convolutional neural network applied to fruit image processing. Appl Sci. 2020;10(10), Art. #3443. https://doi.org/10.3390/app10103443
3. Blasco J, Aleixos N, Molto E. Machine vision system for automatic quality grading of fruit. Biosyst Eng. 2003;85(4):415–423. https://doi.org/10.1016/S1537-5110(03)00088-6
4. Moreda G, Ortiz-Canavate J, Garcia Ramos FJ, Ruiz-Altisent M. Non-destructive technologies for fruit and vegetable size determination – A review. J Food Eng. 2009;92(2):119–136. https://doi.org/10.1016/j.jfoodeng.2008.11.004
5. Moonrinta J, Chaivivatrakul S, Dailey MN, Ekpanyapong M. Fruit detection, tracking, and 3D reconstruction for crop mapping and yield estimation. In: Proceedings of the 11th International Conference on Control Automation Robotics & Vision; 2010 December 7–10; Singapore. Singapore: IEEE; 2010. p. 1181–1186. https://doi.org/10.1109/ICARCV.2010.5707436
6. Koirala A, Walsh KB, Wang Z, McCarthy C. Deep learning for real-time fruit detection and orchard fruit load estimation: Benchmarking of 'MangoYOLO'. Precis Agric. 2019;20(6):1107–1135. https://doi.org/10.1007/s11119-019-09642-0
7. Liu L, Ouyang W, Wang X, Fieguth P, Chen J, Liu X, et al. Deep learning for generic object detection: A survey. Int J Comput Vision. 2020;128(2):261–318. https://doi.org/10.1007/s11263-019-01247-4
8. Girshick R, Donahue J, Darrell T, Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: O'Connor L, editor. Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition; 2014 June 23–28; Columbus, OH, USA. Los Alamitos, CA: IEEE; 2014. p. 580–587. https://doi.org/10.1109/CVPR.2014.81
9. Goodfellow I, Bengio Y, Courville A. Deep learning. Cambridge, MA: MIT Press; 2016.
10. Trask AW. Grokking deep learning. Shelter Island, NY: Manning; 2019.
11. Wu J. Introduction to convolutional neural networks [document on the Internet]. c2017 [cited 2024 Jun 26]. Available from: https://project.inria.fr/quidiasante/files/2021/06/CNN.pdf
12. Zhao X, Wang L, Zhang Y, Han X, Deveci M, Parmar M. A review of convolutional neural networks in computer vision. Artif Intell Rev. 2024;57(4), Art. #99. https://doi.org/10.1007/s10462-024-10721-6
13. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Commun ACM. 2017;60(6):84–90. https://doi.org/10.1145/306538
14. Gu J, Wang Z, Kuen J, Ma L, Shahroudy A, Shuai B, et al. Recent advances in convolutional neural networks. Pattern Recognit. 2018;77:354–377. https://doi.org/10.1016/j.patcog.2017.10.013
15. Irfan D, Gunawan TS. Comparison of SGD, RMSprop, and Adam optimization in animal classification using CNNs. In: Proceedings of the 2nd International Conference on Information Science and Technology Innovation; 2023 February 24–25; Yogyakarta: International Conference on Information Science and Technology Innovation (ICoSTEC); 2023. p. 45–51. Available from: https://icostec.respati.ac.id
16. Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: Unified, real-time object detection. In: O'Conner L, editor. Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition; 2016 June 26 – July 01;Las Vegas, NV, USA. Los Alamitos, CA: IEEE; 2016. p. 779–788. https://doi.org/10.1109/CVPR.2016.91
17. Redmon J, Farhadi A. YOLO9000: Better, faster, stronger. In: O'Conner L, editor. Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition; 2017 July 21–26;. Honolulu, HI, USA. Los Alamitos, CA: IEEE; 2017. p. 7263–7271. https://doi.org/10.1109/CVPR.2017.690
18. Redmon J, Farhadi A. YOLOv3: An incremental improvement [preprint]. arXiv1804.02767; 2018. https://doi.org/10.48550/arXiv.1804.02767
19. Terven J, Córdova-Esparza DM, Romero-González JA. A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS. Mach Learn Knowl Extr. 2023;5(4):1680–1716. https://doi.org/10.3390/make5040083
20. Lohia A, Kadam KD, Joshi RR, Bongale AM. Bibliometric analysis of one-stage and two-stage object detection. Libr Philos Pract. 2021; Art. #4910. Available from: https://digitalcommons.unl.edu/cgi/viewcontent.cgi?article=9123&context=libphilprac
21. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, et al. SSD: Single shot multibox detector. In: Leibe B, Matas J, Sebe N, Weiling M, editors. Proceedings of the 14th European Conference on Computer Vision; 2016 October 11–14;Amsterdam, the Netherlands. Cham: Springer; 2016. p. 21–37. https://doi.org/10.1007/978-3-319-46448-0_2
22. Kaur R, Singh S. A comprehensive review of object detection with deep learning. Digit Signal Process. 2023;132, Art. #103812. https://doi.org/10.1016/j.dsp.2022.103812
23. Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A. Speed/accuracy trade-offs for modern convolutional object detectors. In: O'Conner L, editor. Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition; 2017 July 21–26; Honolulu, HI, USA. Los Alamitos, CA: IEEE; 2017. p. 7310–7311. https://doi.org/10.1109/CVPR.2017.351
24. Girshick R, Donahue J, Darrell T, Malik J. Region-based convolutional networks for accurate object detection and segmentation. IEEE Trans Pattern Anal Mach Intell. 2015;38(1):142–158. https://doi.org/10.1109/TPAMI.2015.2437384
25. Girshick R. Fast R-CNN. In: O'Conner L, editor. Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV); 2015 December 11–18; Santiago, Chile. Los Alamitos, CA: IEEE; 2015. p. 1440–1448. https://doi.org/10.1109/ICCV.2015.169
26. He K, Gkioxari G, Dollar P, Girshick R. Mask R-CNN. In: O'Conner L, editor. Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV); 2017 October 22–29; Venice, Italy. Los Alamitos, CA IEEE; 2017. p. 2961–2969. https://doi.org/10.1109/ICCV.2017.322
27. Fu L, Majeed Y, Zhang X, Karkee M, Zhang Q. Faster R-CNN-based apple detection in dense-foliage fruiting-wall trees using RGB and depth features for robotic harvesting. Biosyst Eng. 2020;197:245–256. https://doi.org/10.1016/j.biosystemseng.2020.07.007

28. Fuentes-Peñailillo F, Carrasco Silva G, Pérez Guzmán R, Burgos I, Ewertz F. Automating seedling counts in horticulture using computer vision and AI. Horticulturae. 2023;9(10), Art. #1134. https://doi.org/10.3390/horticulturae9101134

29. Yang R, Hu Y, Yao Y, Gao M, Liu R. Fruit target detection based on BCo-YOLOv5 model. Mobile Inf Syst. 2022;2022, Art. #8457173. https://doi.org/10.1155/2022/8457173

30. Espinoza S, Aguilera C, Rojas L, Campos PG. Analysis of fruit images with deep learning: A systematic literature review and future directions. IEEE Access. 2023;12:3837–3859. https://doi.org/10.1109/ACCESS.2023.3345789

31. Cuong NH, Trinh TH, Meesad P, Nguyen TT. Improved YOLO object detection algorithm to detect ripe pineapple phase. J Intell Fuzzy Syst. 2022;43(1):1365–1381. https://doi.org/10.3233/JIFS-213251

32. Syazwani RW, Asraf HM, Amin MM, Dalila KN. Automated image identification, detection and fruit counting of top-view pineapple crown using machine learning. Alexandria Eng J. 2022;61(2):1265–1276. https://doi.org/10.1016/j.aej.2021.06.053

33. Kanjanawattana S, Teerawatthanaprapha W, Praneetpholkrang P, Bhakdisongkhram G, Weeragulpiriya S. Pineapple sweetness classification using deep learning based on pineapple images. J Image Graphics. 2023;11(1):47–52. https://doi.org/10.18178/joig.11.1.47-52

34. Liu TH, Nie XN, Wu JM, Zhang D, Liu W, Cheng YF, et al. Pineapple (*Ananas comosus*) fruit detection and localization in natural environment based on binocular stereo vision and improved YOLOv3 model. Precis Agric. 2023;24(1):139–160. https://doi.org/10.1007/s11119-022-09935-x

35. Chang CY, Kuan CS, Tseng HY, Lee PH, Tsai SH, Chen SJ. Using deep learning to identify maturity and 3D distance in pineapple fields. Sci Rep. 2022;12(1), Art. #8749. https://doi.org/10.1038/s41598-022-12096-6

36. Rodriguez-Vazquez J, Fernandez-Cortizas M, Perez-Saura D, Molina M, Campoy P. Overcoming domain shift in neural networks for accurate plant counting in aerial images. Remote Sens. 2023;15(6), Art. #1700. https://doi.org/10.3390/rs15061700

37. VideoLAN. VLC media player [software]. Version 3.0.9.2. Paris: VideoLAN; 2020. Available from: https://www.videolan.org/

38. Alekhin A, Aleksei T, Alexander N, Tulegenov A, Golubev A, Khakimova A, et al. OpenCV version 3.4.10 [software]. Available from: https://github.com/opencv/opencv

39. Dutta A, Zisserman A. The VIA annotation software for images, audio and video. In: Amsaleng L, Huet B, Larson M, editors. Proceedings of MM'19: The 27th ACM International Conference on Multimedia; 2019 October 21–25; Nice, France. New York: Association for Computing Machinery; 2019. p. 2276–2279. https://doi.org/10.1145/3343031.3350535

40. Bargoti S, Underwood J. Deep fruit detection in orchards. In: Okamura A, editor. 2017 IEEE International Conference on Robotics and Automation (ICRA); 2017 May 29 – June 03. Singapore: IEEE; 2017. p. 3626–3633. https://doi.org/10.1109/ICRA.2017.7989417

41. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L. ImageNet: A large-scale hierarchical image database. In: Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition; 2009 June 20–25; Miami, FL, USA. Los Alamitos, CA: IEEE; 2009. p. 248–255. https://doi.org/10.1109/CVPR.2009.5206848

42. Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, et al. Microsoft coco: Common objects in context. In: Proceedings of the 13th European Conference on Computer Vision; 2014 September 6–12; Zurich, Switzerland. Cham: Springer; 2014. p. 740–755. https://doi.org/10.1007/978-3-319-10602-1_48

43. Fei-Fei L, Deng J, Russakovsky O, Berg A, Li K. ImageNet [homepage on the Internet]. c2020 [cited year 2020 Apr 01]. Available from: https://www.image-net.org/

44. Lin TY, Patterson G, Ronchi MR, Cui Y, Maire M, Belongie S, et al. COCO: Common objects in context [homepage on the Internet]. c2015 [cited 2020 Apr 01]. Available from: https://cocodataset.org/#home

45. Yosinski J, Clune J, Bengio Y, Lipson H. How transferable are features in deep neural networks? In: Ghahramani Z, Welling M, Cortes C, Lawrence N, Weinberger KQ, editors. Proceedings of Advances in Neural Information Processing Systems 27 (NIPS 2014); 2014 December 08–13. Montréal, Canada. Red Hook, NY: Curran Associates, Inc; 2014. p. 1–9. Available from: https://proceedings.neurips.cc/paper_files/paper/2014/file/532a2f85b6977104bc93f8580abbb330-Paper.pdf

46. Elharrouss O, Akbari Y, Almadeed N, Al-Maadeed S. Backbones-review: Feature extractor networks for deep learning and deep reinforcement learning approaches in computer vision. Comput Sci Rev. 2024;53, Art. #100645. https://doi.org/10.1016/j.cosrev.2024.100645

47. Haruna Y, Qin S, Chukkol AH, Yusuf AA, Bello I, Lawan A. Exploring the synergies of hybrid convolutional neural network and Vision Transformer architectures for computer vision: A survey. Eng Appl Artif Intell. 2025;144, Art. #110057. https://doi.org/10.1016/j.engappai.2025.110057

48. Abdulla W. Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow [software]. Version 2.1; 2017. Available from: https://github.com/matterport/Mask_RCNN

49. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: O'Conner L, editor. Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition; 2016 June 26 – July 01; Las Vegas, NV, USA. Los Alamitos, CA: IEEE; 2016. p. 770–778. https://doi.org/10.1109/CVPR.2016.90

50. Everingham M, Van Gool L, Williams CK, Winn J, Zisserman A. The Pascal visual object classes (VOC) challenge. Int J Comput Vision. 2010;88(2):303–338. https://doi.org/10.1007/s11263-009-0275-4

51. Rezatofighi H, Tsoi N, Gwak J, Sadeghian A, Reid I, Savarese S. Generalized intersection over union: A metric and a loss for bounding box regression. In: O'Conner L, editor. Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2019 June 16–20; Long Beach, CA, USA. Los Alamitos, CA: IEEE; 2019. p. 658–666. https://doi.org/10.1109/CVPR.2019.00075