Report of Machine Vision

Tengfeng Zhang - 21070337 Wentao Gao - 21070393 Jingya Zhang - 21070391

Shiyao Huang - 21070388 Boxuan Xia - 21070365 Tianjian Zhong - 21070353

No.	Name of group members	Contribution to project	Contribution to report	Signature
1	Tengfeng Zhang	Data processing, ML model construction, part of model training, design of experiment of comparison between two approaches.	Section 3, Section 4.2, the ML part of Section 5 and Section 6.	Tengfeng Zhang
2	Wentao Gao	Achievement of approach A, comparison of two approaches (20%).	Section 1,2,4 (conventional), 5 (conventional), 6.	Wentao Gao
3	Jingya Zhang	Implementation of approach B	40% of Section 1, 50% of Section 2, 20% of Section 6, Section 7	Jingya Zhang
4	Shiyao Huang	Achievement of approach A, comparison of two approaches (20%).	Section 1,2,4 (conventional), 5 (conventional), 6.	Shiyao Huang
5	Boxuan Xia			Boxuan Xia
6	Tianjian Zhong	Achievement of approach A, comparison of two approaches (20%).	Section 1,2,4 (conventional), 5 (conventional), 6.	Tianjian Zhong

1 INTRODUCTION

1.1 Overview

For growers, the efficiency of labour resources in orchards plays an important role in yield, and the method of measuring yield determines the efficiency of labour resource use. Manual metering is the dominant method in most areas of the orchard, it is labour intensive, expensive, and disruptive. As technology has developed, machine vision technology has opened up more possibilities for agricultural metering. With machine vision technology, it is possible to achieve spatial and efficient data on a large scale, including the inspection of fruit, vegetables, and grading (Häni et al. 2018).

1.2 Aims and Objectives

The objective of the experiment was to achieve the task of accurately locating, detecting and counting apples in an orchard. In general, there are two types of machine vision-based inspection: one is based on hand-engineered features such as conventional image processing, and the other one is based on deep learning. In this paper, the number of apples in a photograph of an apple orchard is calculated and the results are compared and analyzed using both approaches.

The objectives of the study includes:

- To review relevant literature and code about the conventional image processing based approach and machine learning approach which can be used in apple counting.
- To review the relevant literature about the MinneApple dataset to understand the composition of this dataset.
- Based on the dataset chose the method to preprocess the dataset.
- Based on the previous review design the algorithms using both conventional image processing based approach and machine learning approach.

- Test the performance of the code and models.
- Analyze the results and compare the two approaches.

1.3 Challenges

The main challenges in achieving the orchard counting task are as follows:

1) Individual differences of apples, including colour, size, and shape.
2) The shading of apples by leaves or other obstacles, or the shading and overlap of multiple apples.
3) The effect of ambient light on the apples, which varies between apples at different light angles.
4) The effect of variable camera views.

2 RELATED WORKS

2.1 Conventional Image Processing

This paper summarises some papers with main conventional image processing methods currently used as follows:

- Mage Filtering: linear filtering (mean filtering, Gaussian filtering, etc.), non-linear filtering (median filtering, bilateral filtering, etc.), morphological filtering (erosion expansion, opening and closing operations, etc.);
- Thresholding (Otsu algorithm, Watershed algorithm, etc.);
- Image pyramid;
- Edge detection (Sobel operator, Canny operator, etc.);
- Hough transform (straight line detection and circle detection);

In addition, an overview of the work that has been carried out in the area of traditional image processing and related papers are as follows:

Nandyal & Jagadeesha (2013) on the use of circle fitting, edge detection and morphological manipulation to predict crop growth.

Patel et al. (2012) used colour and shape features to identify fruit regions, followed by image segmentation and noise reduction to identify the edges of the regions and finally circle fitting.

Zhou et al. (2012) for implementing the required algorithm for segmenting apples in an orchard by different colour models and further comparing fruits by manual techniques.

Chaudhuri et al. (1995) developed a technique for detecting occluded circular objects by morphological manipulation. They designed the algorithm to execute it on different machines in a parallel manner because morphological operations are slower than other basic image processing operations.

Lee et al. (2012) proposed a fast target detection algorithm based on colour histograms and local binary patterns (LBP) with two steps: coarse target object detection using integral colour histogram matching and precise target object detection using LBP feature distribution.

Guo et al. (2013) proposed a novel and robust method to significantly enhance collaborative detection by extracting a shared low-rank representation of object instances in multiple feature spaces.

2.2 Machine Learning

VGG16 is a Convolutional Neural Network proposed by the Visual Geometry Group at the University of Oxford (Simonyan & Zisserman 2014). It has deep layers and uses consecutive 3x3 convolutional kernels instead of larger ones. For a given receptive field, using stacked small convolutional kernels is preferable to using larger ones, as multiple non-linear layers increase the network depth to ensure more complex patterns are learned at a lower cost (fewer parameters). The VGG convolutional layers are followed by three fully connected layers. The final Top-5 accuracy on ImageNet was 92.3%. Although VGG can perform

well on ImageNet, deploying it on a modestly sized GPU is difficult because of the high computational requirements of VGG in terms of memory and time. Due to a large number of channels in the convolutional layers, VGG is not efficient.

Xception is an improvement on Inception-v3 (Szegedy et al. 2016) by Google. It uses Separable Convolution (Extreme Inception Module) to replace the convolution operation in the original Inception-v3, separating the correlation between channels from spatial correlation (Chollet 2017). Xception is slightly more accurate than Inception-v3 on ImageNet, while the number of parameters has decreased. The addition of a ResNet-like residual connecting mechanism to Xception also significantly speeds up the convergence process and achieves significantly higher accuracy. However, due to the fragmented nature of the computation process, it is not efficient enough in any of the existing convolutional neural network implementations.

ResNet is the residual network developed by He et al. (2016) and winner of the 2015 ILSVRC competition. The ResNet team found that as the depth of the network increased, the accuracy did not always increase, but rather decreased when the network is too deep, a phenomenon they called "Degradation". So the ResNet team added a Shortcut Connection branch to the ResNet module: the signal input to a layer is also added to the output of the layer above the heap branch (Aurélien 2017). Thanks to the shortcut connection, signals can easily span this network. Using the residual module, a 152-layer residual network can be trained and the Top-5 error rate reduced to 3.6%. It is more accurate and computationally more efficient than VGG and GoogLeNet.

3 Data Acquisition and Datasets

Photos in the dataset for this experiment were taken with a mobile phone. It is more suitable for machine learning than using a camera to obtain the dataset, because of the high resolution of a camera, the dataset may be too large, resulting in a slow training speed.

The dataset was obtained from a video (Häni et al. 2020). Within the data collection, the photographer moved at a speed of one metre per second to minimise camera shake and ensure a clear picture. The camera was kept in a plane parallel to one side of the apple tree row. A photo is then acquired every 5 frames as part of the dataset to ensure a large scale.

At the same time, the diversity of the dataset needs to be guaranteed. For instance, apples can be collected in a variety of colours, such as red, cyan, yellow, etc. In addition, apples may be obscured by obstacles such as leaves, another apple, etc. These images can be collected to enhance the variety of the data.

The dataset, which contains two sub-dataset, detection and counting. MinneApple is used in this project, which is an open-source dataset collected by researchers from the Robotic Sensor Networks Lab at the University of Minnesota. The detection dataset has a training set of 670 1280×720 pixel images and includes photos of apple trees in sunlight and shade, with a maximum number of 120 fruit. Over 60,000 images from 6 different scenes were collected into the counting dataset, with 0-6 apples in a single image. The photos vary in size, which leads to the requirement to resize the input images when applying machine learning methods, but which may affect the final result. The two datasets are also consistent with the large amount and high diversity that is expected.

Although the dataset only has a small number of object categories, it contains a very large number of apple instances. For the conventional method, the detection dataset can be used, as it contains plenty of apples that can be easily labelled and counted. The machine learning method, on the other hand, can use the counting dataset because it has labels that can be easily applied to training. For the later comparison between the two methods, we will use the detection dataset with images containing tens to over a hundred apples, as this is more relevant to realistic applications.

4 METHODOLOGY

4.1 Approach A: Conventional Image Processing

In the machine vision part, an apple recognition and counting algorithm is designed. The counting algorithm is shown in figure 1 and consists of the following steps: Gaussian blurring the image, converting the RGB image to HSV, creating binary graphs based on thresholds, Noise reduction through opening, and counting.

4.1.1 Gaussian Blurring the Image

First, use a Gaussian Filter to smooth the image, which means convolved the source image with the Gaussian kernel of 11×11. When the Gaussian filter is used to smooth the pixels in the image neighbourhood, the pixels at different positions in the neighbourhood are given different weights. This means that the image is smoothed. By applying gaussian filter to image,

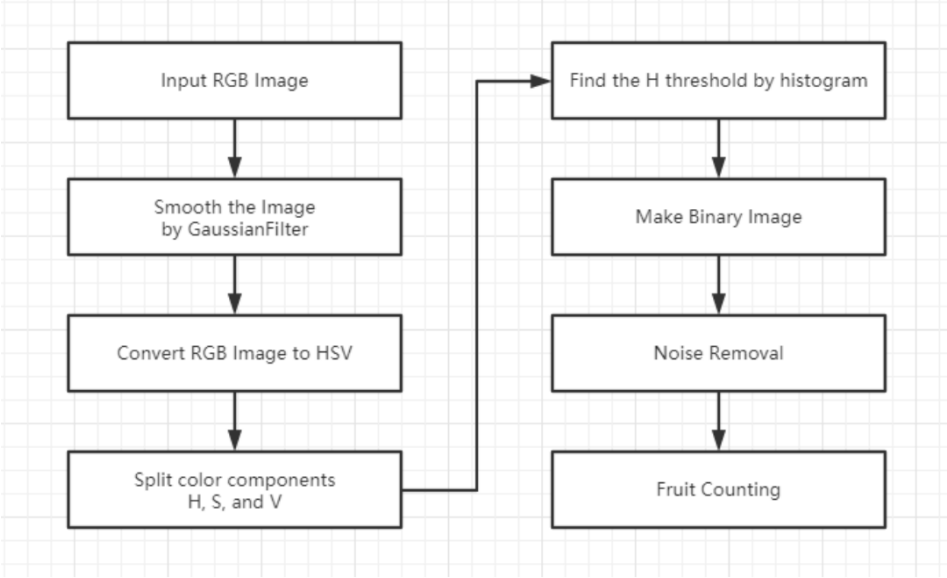

Figure 1: Flowchart of conventional image processing approach

objects with a smaller size and brightness can be filtered, while larger objects can be easily detected. In this way, it is easier to extract objects from image processing.

4.1.2 Converting the RGB Image to HSV

Converting RGB images to HSV images can extract target features more effectively (Dorj et al. 2017). This is because RGB images acquired in a natural environment are easily affected by natural lighting, occlusion, and shadows, which change the brightness of colours. The red, green, and blue components of the RGB colour space are all closely related to brightness. If the brightness changes, the three components will change accordingly. Therefore, in image processing, it is difficult to find a colour through accurate RGB values. On the other hand, as shown in figure 2, the HSV colour space remaps the RGB model, which can intuitively express the hue (H), saturation (S) and brightness value (V) (PM & Chezian 2013) of the colour. Hue (H) represents different colours by the hue circle of 0-360°, and red is usually represented by 0-15° and 165-180°. Saturation (S) describes the vividness of the colour by 256 components. The more vivid the colour, the higher the saturation. Brightness Value (V) also describes the brightness of the colour by 256 components, the lower the value, the darker the colour.

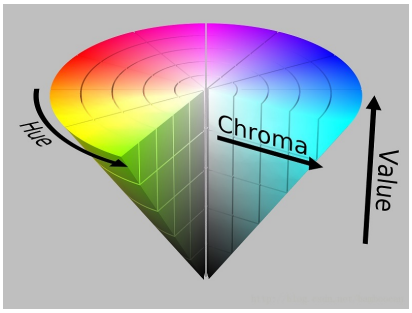

Figure 2: Illustration of HSV

4.1.3 Create Binary Graphs Based on Thresholds

The HSV histogram shows the overall distribution of H, S, V pixel values in the image. As shown in figure 3, the horizontal axis is the pixel value, and the vertical axis is the frequency of occurrence of the pixel value. According to the H, S, V curves of the histogram, the hue range of the apple colour is between 0°-10° and 170°-180°, the saturation range is between 30°-255°, and the brightness value range is between 30°-255°. Based on the HSV range obtained by observing the histogram, manually adjust the threshold to find the red area in the picture and make the extracted area into a binary image. The effect is shown in figure 4 and 5.

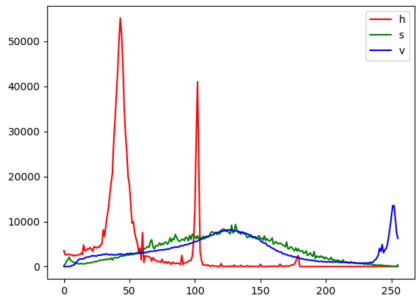

Figure 3: Distribution of H, S, V pixels respectively

Figure 4: Red area of the image

Figure 5: Binarisation of the image

4.1.4 Noise Reduction Through Opening, and Counting

As shown in figure 6, there are some small noise points in the extracted binary image. Through comparison with the original image, it is found that these noise points do not represent an apple very well, so these noise points will affect the count of apples. It is necessary to reduce the noise of the binary image, and the smaller noise can be eliminated through the opening operation. The opening is a filter based on geometric operations, which can filter some objects that are too small, while the overall position and shape are unchanged. The principle of opening to eliminate noise is to first eroded the image and then dilated. Eroded is the process of fitting each pixel in a binary image in turn. Fit is to determine whether all the pixels in the structuring element overlap with the image area of the pixel. If they all overlap, the pixel is 1, otherwise it is 0. Dilated is the process of hitting each pixel in the image in turn. Hit is to determine whether there is a pixel in the structuring element that overlaps with the image area of the pixel. If there is overlap, the pixel is 1, otherwise it is 0. In order to get a better image, eroded and dilated may need to be performed multiple times. In image processing, judging whether it is necessary to perform eroded and dilated multiple times usually depends on experience and intuitive observation of the image. In this experiment, in order to eliminate noise, the binary image was eroded and dilated twice with a 5×5 structuring element. The effect is shown in figure 6. After that, use the opencv function to find the centroid in the processed binary image, the position of the apple in the image (figure 8), and get the total number of apples that could be found.

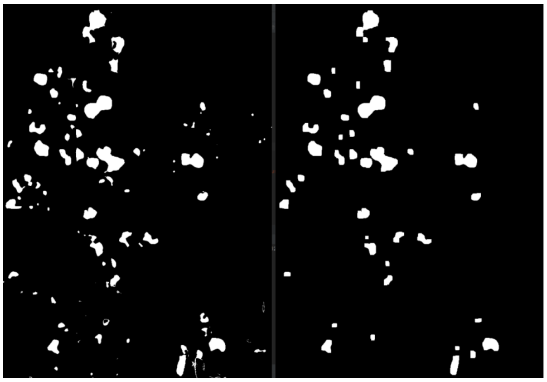

Figure 6: The effect of opening operation on noise

Figure 7: Original RGB photo

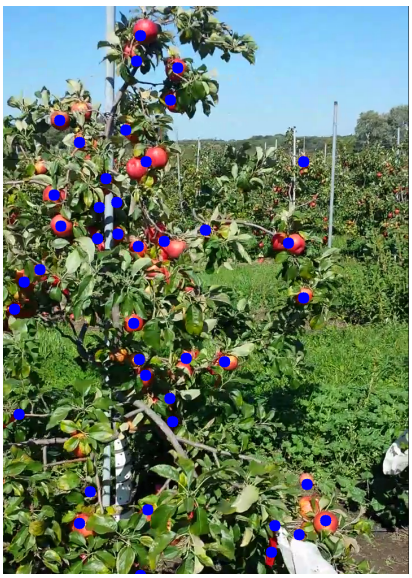

Figure 8: Labelling the positions of apples

4.2 Approach B: Machine Learning

The experiments used existing popular backbones rather than building new networks, such as resNet50, VGG16, Xception, EfficientNetB0 and InceptionV3. Because using these network structures can accomplish tasks with higher quality and more efficiently. More importantly, no sufficient cost helps learn to build more efficient network structures.

More experiments have been done on the model types and the training of single models. At first, resNet50 was used as the baseline (Häni et al. 2018). The first reusable model was successfully trained with no changes to any of the layers except for the output layer as it needed to match the specific classification problem. Subsequently, the regularisation layer and the dynamic learning rate were applied to make the model more generalised.

In addition, when processing the dataset, the method of reading images is modified because all the images could not be stored in memory at once, which means that only the required images are read when the current batch is being trained, which releases the memory pressure.

Overall, the implementation of counting apples using machine learning methods is roughly divided into the following stages.

1) Data acquisition. The required data is downloaded from the website.

2) Indices shuffling. As the original dataset was labelled with images in the order of 0 6. After reading the images and labels sequentially, the data would be unbalanced, so the indices are shuffled for an even distribution of the categories.

3) Splitting dataset. The dataset has a training set, a validation set, and a test set, where the training and validation sets have labels and the test set does not (needed to be submitted to the competition platform published by the dataset creator to obtain the results). For local testing, 10% of the training set images are separated as the test set.

4) Resize images. As the photos are available in a large number of sizes, it was necessary to resize the image to a fixed input size to facilitate later training.

5) Build the model. Use the Keras.applications library to download the model backbones, modify the output to the number applicable to this project, adjust the network structure appropriately, and set the appropriate optimiser and learning rate.

6) Train the model. Set epoch to 10 and train each model in each structure.

7) Evaluate the model. Firstly observe the accuracy of the training and validation sets. If the models perform similarly and at a high level on these two values, the models are considered to have good performance. Otherwise, it is overfitting or underfitting and needs to be adjusted. Finally, the model is evaluated using the test set.

Training robust models with deep learning method to solve the counting apples problem is ideal because there is a large amount of data for this experiment.

5 EXPERIMENT AND IMPLEMENTATION

5.1 Conventional Image Processing

In the machine vision method, the code is based on Python 3.9, the Python IDE used for programming is PyCharm, the operating platform is Windows platform, the CPU of the operating environment is Intel Core i7-9750H 2.60GHz, and the memory is 16GB. The pseudocode for the program implementation is indicated below, the main Python libraries used are numpy, matplotlib.pyplot, from scipy import ndimage, opencv2.Processing images with traditional computer vision methods requires manual adjustment of some parameters. This includes the threshold of the hue in the HSV image when forming the binarized image, and the number of opening iterations of the opening operation in the noise reduction process. In this process, it needs to observe the HSV histogram of the picture, combine the RGB colour distribution of the picture, and try an appropriate threshold according to experience to set the threshold of hue. For example, in the HSV histogram (graph 3), the hue is mainly concentrated between 0-10, 30-70, 90-110 and 170-180. In the corresponding RGB image (figure 7), the picture is mainly composed of green, part of blue and a small amount of red. Therefore, it can be inferred that in the HSV histogram, the most pixels are between hue 30°-70°, representing the green part, between 90°-110° is the blue part, and between 0-10° and 170-180° are red part. The threshold of hue is set as in the pseudocode to find all the red areas in the image. After that, in the noise reduction process, the number of opening iterations is set, and the size of the extraction target in the original image needs to be determined. If the size of the extraction target in the original image is small, the number of iterations is usually set between 1 and 3, because more iterations will cause some targets to be lost. On the other hand, if the size of the extraction target is larger in the original image, a higher number of iterations will help to obtain more accurate results.

```
# imput image
img_bgr = image read

# Gaussian Blurring by 11x11 convolution kernel
blurred = cv2.GaussianBlur(img_bgr, (11, 11), 0)

#BGR convert to HSV
img_hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)

# use h s v threshold to make a mask
RedRange1 = (0,50,20) to (10,50,20)
RedRange2 = (170,50,20) to (180,50,20)
mask1 = cv2.inRange(img_hsv, lower_RedRange1, upper_RedRange1)
mask2 = cv2.inRange(img_hsv, lower_RedRange2, upper_RedRange2)
mask_raw = mask1 + mask2

# noise removal for mask by 5x5 kernel
kernel = np.ones((5,5), np.uint8)
mask = cv2.morphologyEx(mask_raw, cv2.MORPH_OPEN, kernel, iterations=2)

# find contours
labels, nlabels = ndimage.label(mask)
centroid = ndimage.center_of_mass(mask, labels, np.arange(nlabels) + 1)
```

5.2 Machine Learning

When applying the machine learning approach, the version of Python used is 3.7 and the IDE is jupyter notebook, running on Windows 10 OS, the CPU is Intel core i7-11800H, the GPU is GeForce RTX 3060, and the memory is 16GB.

The main Python libraries used in this experiment include Keras in TensorFlow, matplotlib.pyplot, and OpenCV. The GPU is used to run the program in order to accelerate the training process.

1) Data processing

The labels of the training and validation set are first read into memory and the filenames of all the images in the 'images' folder are obtained. Then the filenames are sorted so that they could correspond to the labels.

10% of the training package is selected for testing, and the validation package remains unchanged. In addition, the training assembly images are distributed sequentially, meaning that each category is grouped together. Gradient descent works best when the instances in the training set are independent of each other and evenly distributed, so the shuffle command is used to change the order of indices.

The images need to be resized before being trained, as the fully connected layer of the network structure has a fixed number of neurons and requires the same image size to be the correct input. The images are eventually scaled down to a size of 200*200 pixels and interpolation hyperparameter uses cv2.INTER_AREA, which is generally the preferred method as it resamples the images according to the pixel-region relationships to make the results more accurate.

The way to read images is that they are only loaded at the current batch that is being trained based on the filenames, so a generator function is implemented and personalised for training using fit_generator. This generator function is shown in the following pseudocode.

```
class My_Custom_Generator(keras.utils.Sequence) :
def __init__(self, image_filenames, labels, batch_size, path) :
self.image_filenames = image_filenames
self.labels = labels
self.batch_size = batch_size
self.path = path

def __len__(self) :
return total_batch_count

def __getitem__(self, idx) :
batch_x = filenames_for_current_batch
batch_y = labels_for_corresponding_images
```

```
return read_and_resize_images, batch_y
```

2) Model training

The next step is to build the models. The experiments are carried out using 5 popular models for training: ResNet50, VGG16, Xception, EfficientNetB0, and InceptionV3. The characteristics of each model are listed in table 1 (Keras 2022), including the size of the memory, the number of parameters, depth, and the respective CPU and GPU training time (but the running times on different hardware devices may be various).

Table 1: Details of each backbones

Model	Size (MB)	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
Xception	88	22,910,480	126	109.42	8.06
VGG19	549	143,667,240	26	84.75	4.38
ResNet50	98	25,636,712	-	58.20	4.55
InceptionV3	92	23,851,784	159	42.25	6.86
EfficientNetB0	29	5,330,571	-	46.00	4.91

In addition, the Adam optimiser is adopted with an initial learning rate of 0.00001, while the initial weights are set as the one that is pre-trained on the image net dataset. In addition, a reminder function for an exponential decay scheduler is also used to dynamically adjust the learning rate. The formulation is presented in equation 1.

$$lr^{(i+1)} = lr_{initial} \times 0.1^{\frac{epoch_i}{epoch_all}} \tag{1}$$

Apart from that, the final layer of the network is modified by adding a global average pooling layer after removing the output layer of the original backbone. Three regularisation structures are also tested, as illustrated in figure 9.

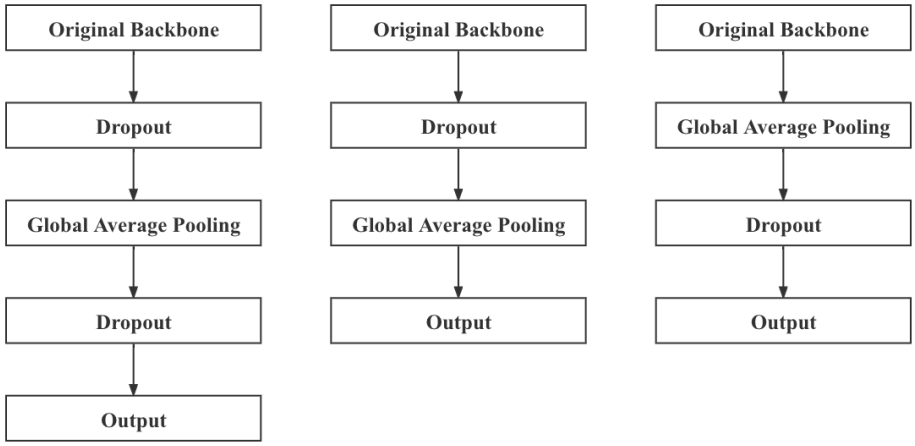

Figure 9: Demonstration of three structures

As a result, the output layer is defined by 7 neurons and the activation function uses softmax.

When compiling the model, sparse_categorical_crossentropy is set as the loss function, as the labels of the dataset are sparse and the classes are mutually exclusive. The metrics used to measure the performance of the models are accuracy and the epoch is 10.

At the end of the training, the metrics of the models on the training and validation sets can be observed to determine whether the model is generalised enough, overfitting or underfitting.

At last, the model is evaluated on the test set to obtain the latest results of the loss function and accuracy, which quantifies the degree of generalisation of the model.

6 RESULTS AND EVALUATION

6.1 Conventional Image Processing

The key point of the traditional image processing approach designed in this paper is the extraction of the red colour in the original image using the hsv colour space, while taking into account the possible yellow colour of the apples and resetting the relevant threshold range. In this paper, 30 different orchard apple maps were tested using this approach, and the recognition accuracy was around 65%-80%. The range fluctuations are mainly influenced by the colour of the apples, for example, for cyan apples the approach is ineffective; it is included the visual impact on the surface colour of the apples due to lighting and shadows, which can cause the apples to look white or dark, and also reduce the accuracy of the conventional image processing approach. However, for apples with red colour features, the method is able to detect and identify them efficiently. At the same time, the red areas extracted using morphological filtering processing then further segment the apples to improve recognition accuracy.

Compared to the minimum Euclidean distance-based segmentation technique proposed by Syal et al. (2014) to segment fruit regions from the input image, which can perform edge detection of apples, the image processing approach proposed in this paper is limited by colour. However, colour-based image segmentation is more efficient for images in a multi-apple complex environment.

6.2 Machine Learning

In terms of the machine learning method, all the training results are shown in Table 2.

Table 2: Original result of ResNet50

Backbone	acc_train	acc_validation	acc_test
ResNet50	>90%	>60%	67.74%

The training is first performed using an unaltered ResNet50. It is found that the model does process gradient descent, but the result is upwards of 90% on the training set and not as well on the validation and test sets, at around 60%, so the model is regarded as overfitting.

A dropout regularisation method is then added to the network to reduce the probability of overfitting by setting the hyperparameter of the dropout rate to 0.5. Three different ways of appending dropout layers are then tested. '2 dropouts' represents a dropout layer before and after the global average pooling layer, while 'dropout_1' means a dropout before the pooling layer, and 'dropout_2', a dropout layer after the pooling layer is added. Five backbones are trained using 10 epochs and 90% of the training set (58,135 photos), and their performance is shown in Tables 3 to 7.

Table 3: Result of ResNet50 of three structures

	acc_train (%)	acc_validation (%)	acc_test (%)	avg_time (s)	acc_bias_on_train_and_validation (%)
2 dropouts	91.79	88.47	88.18	373.4	3.32
dropout_1	98.91	91.04	91.25	373	7.87
dropout_2	95.34	90.42	89.85	369.6	4.92

Table 4: Result of VGG16 of three structures

	acc_train (%)	acc_validation (%)	acc_test (%)	avg_time (s)	acc_bias_on_train_and_validation (%)
2 dropouts	82.69	83.99	84.59	389.6	-1.3
dropout_1	93.79	91.98	92.42	420.3	1.81
dropout_2	88.39	88.62	89.17	422.2	-0.23

Table 5: Result of EfficientNetB0 of three structures

	acc_train (%)	acc_validation (%)	acc_test (%)	avg_time (s)	acc_bias_on_train_and_validation (%)
2 dropouts	60.05	65.8	67.04	390.5	-5.75
dropout_1	68.36	72.05	70.68	387	-3.69
dropout_2	63.11	69.58	68.44	395.9	-6.47

Table 6: Result of Xception of three structures

	acc_train (%)	acc_validation (%)	acc_test (%)	avg_time (s)	acc_bias_on_train_and_validation (%)
2 dropouts	89.93	86.53	86.17	640	3.4
dropout_1	95.47	88.83	89.52	637.6	6.64
dropout_2	91.24	86.26	86.86	637.4	4.98

Table 7: Result of InceptionV3 of three structures

	acc_train (%)	acc_validation (%)	acc_test (%)	avg_time (s)	acc_bias_on_train_and_validation (%)
2 dropouts	89.65	80.37	80.41	247.5	9.28
dropout_1	98.32	82.99	83.27	244.4	15.33
dropout_2	92.9	81.99	81.78	244.5	10.91

where 'acc_train', 'acc_validation', and 'acc_test' represent the accuracy on the training set, validation set, and test set respectively. 'avg_time' demonstrates the average time taken to train the model for one epoch. 'acc_bias_on_train_and_validation' explains the difference between the accuracy on the training and validation sets, which is used to analyse the goodness of fit.

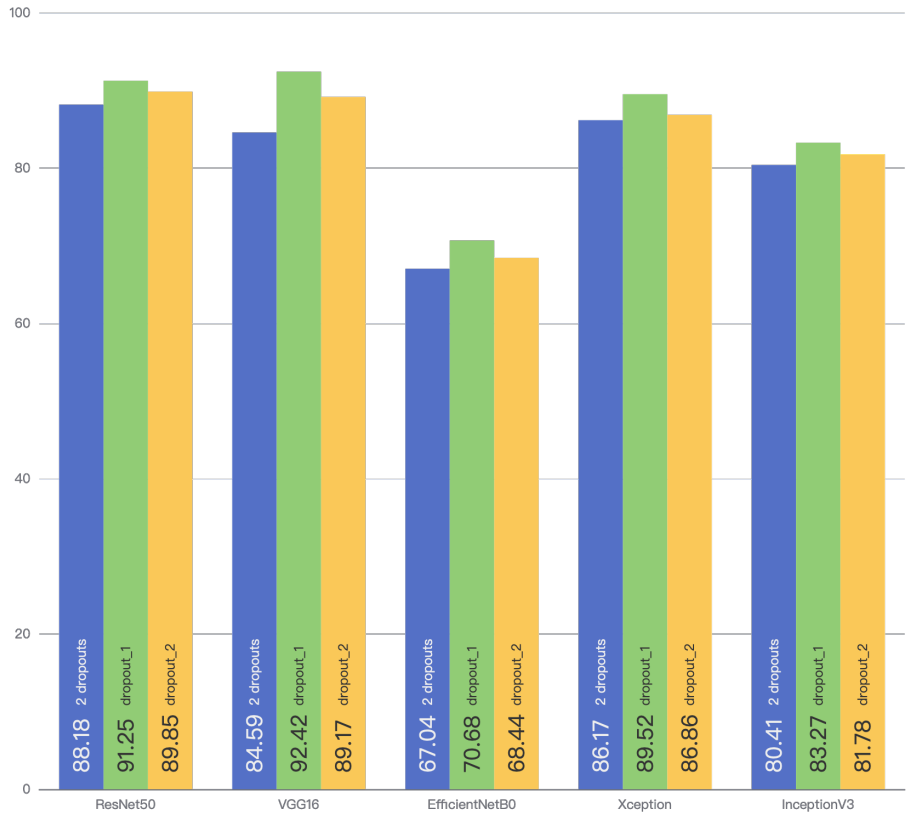

Figure 10: Accuracy of each model with various structures on test set

It can be found that the baseline model ResNet50 performs very well (accuracy of 91.25% on the test set), whilst the training results of VGG16 are better, with a maximum accuracy of 92.42% on the test set. Regarding Xception and InceptionV3, both models also reach an accuracy above 80% on the test set, but the InceptionV3 model is overfitting. The worst results are produced on EfficientNetB0, with a maximum accuracy of only around 70%. All the results are illustrated in figure 10.

To be more specific, while the dropout_1 structure does not fit best for most of the models, the performance of dropout_1 is the best on the test sets among other structures. VGG16, which performs best, the acc_bias_on_train_and_validation of only 1.81% (figure 11) could not prove that it is overfitting, so the VGG16 of dropout_1 structure is the most ideal model in terms

of accuracy. But the training time of it is not the shortest, taking roughly 420s, compared to 373s of ResNet50 with the same structure (figure 12), about one minute less, but the accuracy of ResNet50 is reduced by 1.81%.

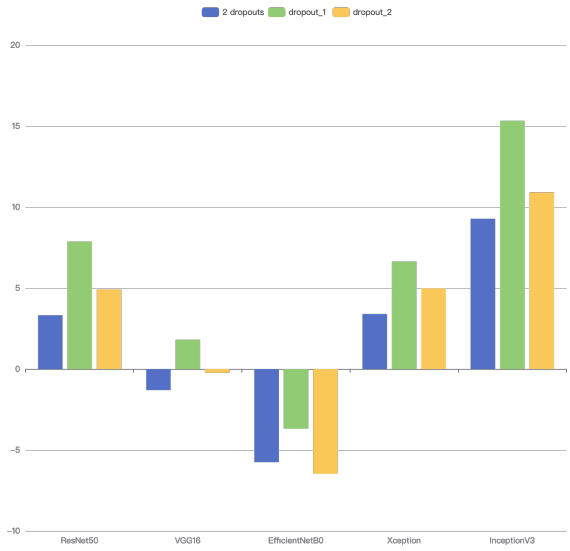

Figure 11: Difference of accuracy between training and validation sets of each model with three structures

Figure 12: Running time of all the models

Finally, the best model VGG16 is compared with the conventional approach. The method of comparison is to select 10 images from the detection dataset in MinneApple, and the number of apples in each photo is counted by team members, then both approaches are applied to the 10 photos for counting. The results are displayed in table 8.

Table 8: Comparison of two approaches

	Ground truth	Conventional approach	ML approach	Difference between GT and conventional method	Difference between GT and machine learning method
figure 1	91	0	99	91	-8
figure 2	65	38	92	27	-27
figure 3	60	20	31	40	29
figure 4	58	76	65	-18	-7
figure 5	72	73	85	-1	-13
figure 6	46	33	32	13	14
figure 7	39	38	38	1	1
figure 8	42	44	24	-2	18
figure 9	62	55	57	7	5
figure 10	50	49	52	1	-2
variance	/	/	/	29.38	15.47

Both methods can count the apples in photos. While the machine learning approach has a smaller variance, which means this method is more accurate.

The reason for the possible error is that the conventional approach requires specific tuning for each image and missing count may happen after binarisation. What's more, because a classifier is trained by the machine learning method, for images containing more than 6 apples, it needs to crop the images which result in one apple being present in two images, resulting in duplicate counts or even not being detected.

'figure 1' is a photo full of green apples, however, conventional approach could not process such kind of images, so it performs badly. Once this picture is removed, both variances are close to one another, which means these two methods have similar performance.

6.3 Comparison and Evaluation

Based on the above results, the advantages and disadvantages of conventional image processing and machine learning are summarized.

6.3.1 Evaluation of Convention Image Processing

- **Advantages**

  1) Fast recognition and small amounts of computing.
  2) Ability to solve certain scene-specific, manually definable, designable and understandable image tasks, with good scene- specific results and greater interpretability.
  3) Cost-effective in extreme low-computing scenarios and can be deployed on low-cost microprocessors.
  4) Limit the problems that deep learning techniques can solve by highlighting specific features in the data, augmenting the data or aiding in dataset annotation.

- **Disadvantages**

  1) When the number of objects contained in the image is large, detail processing is prone to problems such as unrecognisable and incorrect recognition.
  2) Parameters must be manually adjusted to recognise different images and are extremely dependent on expertise and experience.
  3) Generally weak in generalisation and unable to process large volumes of scenes.

6.3.2 Evaluation of Machine Learning

- **Advantages**

  1) It can be very generalised to recognise different images and achieve a high accuracy rate with high precision.
  2) No very targeted technical processing is required, the process of feature extraction is self-learning and the model can be made to learn the apple counting method through a large number of data sets.
  3) There are many existing more advanced network structures for image recognition in the keras library, which can be directly called and simply modified and applied to the project.

- **Disadvantages**

  1) High memory consumption Long training period.
  2) Speed and cycle of algorithm building, deep learning requires GPUs, long training cycles, slow iteration of algorithms.
  3) Visualisation of the algorithm flow, traditional image processing methods, at which step you get what result can be shown and seen.
  4) Deep learning sees a feature map, which cannot be compared with the input map to make a more intuitive feeling.
  5) Relying on a large number of high-quality data sets, for supervised learning, there is the problem of data annotation. But requires a lot of training labeled data and high hardware requirements.

7 CONCLUSIONS AND FUTURE WORKS

The task of counting apples is implemented by using conventional image processing approach and several models trained by different neural networks. Both of them can realise apple counting at a relatively high accuracy. In Particular, the accuracy of the machine learning model is up to 92.4%. However, the conventional methods cannot distinguish the overlapped apples and the colour of the detected apples is limited. Whilst the accuracy of the machine learning model decreased when testing on pictures of apple trees which simulates the real world. That's because the apples' images in the training set are more complete and clearer than the apples in the real world.

The future work will be improving the performance of the algorithms in reality . Based on the analysis of the advantages and disadvantages of the two approaches, combining these two methods might be a promising orientation to enhance the performance. Some features can be extracted manually since this method is more intuitive and memory saving. Machine learning models then implement the classification with the extracted features.

References

Aurélien, G. (2017), 'Hands-on machine learning with scikit-learn & tensorflow', *Geron Aurelien* .

Chaudhuri, A. R., Chanda, B. & Chaudhuri, B. B. (1995), 'Detection of occluded circular objects by morphological operators', *Signal processing* **46**(2), 233–242.

Chollet, F. (2017), 'Xception: Deep learning with depthwise separable convolutions'.

Dorj, U.-O., Lee, M. & Yun, S.-s. (2017), 'An yield estimation in citrus orchards via fruit detection and counting using image processing', *Computers and Electronics in Agriculture* **140**, 103–112.

Guo, X., Liu, D., Jou, B., Zhu, M., Cai, A. & Chang, S.-F. (2013), Robust object co-detection, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 3206–3213.

Häni, N., Roy, P. & Isler, V. (2020), 'Minneapple: a benchmark dataset for apple detection and segmentation', *IEEE Robotics and Automation Letters* **5**(2), 852–858.

He, K., Zhang, X., Ren, S. & Sun, J. (2016), Deep residual learning for image recognition, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 770–778.

Häni, N., Roy, P. & Isler, V. (2018), Apple counting using convolutional neural networks, *in* '2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', pp. 2559–2565.

Keras (2022), 'Keras applications'.
URL: *https://keras.io/api/applications/usage-examples-for-image-classification-models*

Lee, K., Lee, C., Kim, S.-A. & Kim, Y.-H. (2012), Fast object detection based on color histograms and local binary patterns, *in* 'TENCON 2012 IEEE Region 10 Conference', IEEE, pp. 1–4.

Nandyal, S. & Jagadeesha, M. (2013), 'Crop growth prediction based on fruit recognition using machine vision', *International Journal of Computer Trends and Technology (IJCTT)* **4**(9), 3132–3138.

Patel, H. N., R.k.jain & M.v.joshi (2012), 'Article: Automatic segmentation and yield measurement of fruit using shape analysis', *International Journal of Computer Applications* **45**(7), 19–24. Full text available.

PM, N. & Chezian, R. M. (2013), 'Various colour spaces and colour space conversion algorithms', *Journal of Global Research in Computer Science* **4**(1).

Simonyan, K. & Zisserman, A. (2014), 'Very deep convolutional networks for large-scale image recognition', *arXiv preprint arXiv:1409.1556* .

Syal, A., Garg, D. & Sharma, S. (2014), Apple fruit detection and counting using computer vision techniques, *in* '2014 IEEE International Conference on Computational Intelligence and Computing Research', pp. 1–6.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. & Wojna, Z. (2016), Rethinking the inception architecture for computer vision, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 2818–2826.

Zhou, R., Damerow, L. & Blanke, M. M. (2012), 'Recognition algorithms for detection of apple fruit in an orchard for early yield prediction', *Precision Agriculture* **13**(5), 568–580.