Strawberry Counting and Ripeness detection
Computer Vision Final Project
Group report

Group 5
Weiwei Wan, 22301337
Prishita Singh, 22306048
Long Pan, 21332147

1. Introduction

Object detection is a technology related to computer vision and image processing which can detect instances of semantic objects of a certain class in digital images and videos. There are some traditional object detection methods based on hand-crafted features, such as Scale-Invariant Feature Transform (SIFT) or Speeded Up Robust Features (SURF). They typically involve detecting features in an image and then using those features to classify or locate objects. In recent years, with the development of deep learning, object detection has achieved very great improvements.
Deep learning-based object detection methods have become popular due to their high accuracy. These methods are based on CNNs and are typically more accurate than traditional methods. They learn to detect objects by training the network on a large dataset of images labelled with object bounding boxes. There are many popular object detection algorithms including YOLO (You Only Look Once), SSD (Single Shot MultiBox Detector), Faster R-CNN, Mask R-CNN.
Object detection could be used in many fields including self-driving cars, security and surveillance systems, and image retrieval systems. With the development of 5G and edge computing, object detection has also started to play an important role in real-time applications such as augmented reality, robotics and drones.Here our group chose 3 modern algorithms including Faster R-CNN, YOLOv7, and SSD to do the strawberry counting and ripeness classification.

2. Background

2.1 YOLO

YOLO (You Only Look Once) is a real-time object detection algorithm that was developed by Joseph Redmon and Ali Farhadi, it's first published in 2016 [1]. It is a single convolutional neural network (CNN) that is trained to predict bounding boxes and class probabilities directly from full images in one evaluation.

YOLO works by dividing the input image into a grid of cells and assigning each cell a probability of containing an object. The algorithm then uses these probabilities to predict the bounding boxes for the objects in the image. Each bounding box is associated with a class label, which allows YOLO to classify the objects in the image.

YOLO has very fast speed, as it can process images in real-time. It is also relatively easy to implement, making it a popular choice for object detection tasks. Another advantage of YOLO is that it is a one-stage object detection algorithm, meaning that it does not require any additional stages such as region proposal generation or non-maximum suppression to generate the final detections. This makes it faster and more efficient than other object detection algorithms, such as R-CNN and Fast R-CNN.
There have been several versions of YOLO released over the years, including YOLOv2, YOLOv3, YOLOv4, YOLOv5, YOLOx, YOLOv6, and YOLOv7. These versions have progressively improved the accuracy of the algorithm, while also increasing its speed and simplicity.

The latest YOLO version is YOLOv7, YOLOv7 is a very fast and accurate object detection model, it is able to achieve high accuracy while still being able to run at a high frame rate.YOLOv7 could reach 30 FPS or higher on GPU V100 with the highest accuracy 56.8% AP, outperforms other state-of-art object detection models in terms of both speed and accuracy including SWINL Cascade-Mask R-CNN, ConvNeXt-XL Cascade-Mask R-CNN, YOLOR, YOLOX, Scaled-YOLOv4, YOLOv5, DETR, Deformable DETR, DINO-5scale-R50, and ViT-Adapter-B in both speed and accuracy, as figure 1 shows [2].

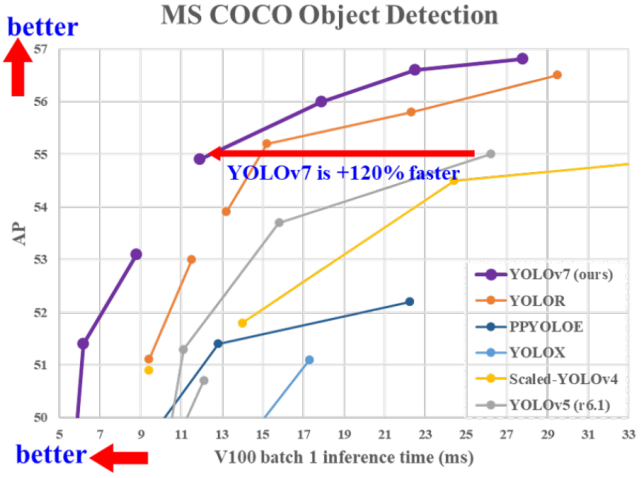


Figure 1: Comparison with other real-time object detectors, YOLOv7 achieves state-of-the-art performance.

YOLO is able to process images in real-time, making it suitable for use in a variety of applications such as self-driving cars and security systems, actually it has been widely used and has achieved state-of-the-art performance on a number of object detection benchmarks. It has also been implemented in a variety of applications including strawberry detection. There is a study which developed a SDNet strawberry growth detection algorithm (Strawberry Detect Net) for detecting strawberry growth and maturity for unmanned farms. The feature extraction module C3HB block, which was self-designed, substitutes the original CSP block in the backbone in the algorithm, which is based on

the YOLOX model, to increase the detection method's spatial interaction capacity and monitoring precision. The precision, accuracy, and recall of SDNet reached 94.26%, 93.15%, and 90.72%, and its monitoring speed is 30.5 ms [3]. And there is another study that describes a novel harvesting robot for strawberries and a fruit pose estimator called rotated YOLO (R-YOLO). R-YOLO was used to predict the rotation of the bounding boxes around the target fruits. The model was tested on a set of 100 strawberry images and had an average recognition rate of 94.43% and a recall rate of 93.46%, with a processing speed of 18 frames per second. The robot had a harvesting success rate of 84.35% in modified field conditions [4].

2.2 K-Nearest Neighbour

The research paper [6] mentioned is about using machine learning to identify the ripeness of strawberries using RGB colour values and the K-Nearest Neighbour (KNN) algorithm. The authors collected a dataset of strawberries that were labelled as either "ripe" or "unripe," and extracted three RGB colour features (red, green, and blue) from each strawberry in the dataset. They then used the KNN algorithm to classify the strawberries as either ripe or unripe based on these colour features.
The authors found that their approach was able to accurately classify the ripeness of strawberries with an overall accuracy of 95.2%. They also found that the KNN algorithm performed better at classifying ripe strawberries than unripe ones, with an accuracy of 97.5% for ripe strawberries and 93.0% for unripe strawberries.Overall, the authors conclude that using RGB colour features and the KNN algorithm is a promising approach for accurately identifying the ripeness of strawberries.

2.3 Single-Shot Detector (SSD)

In paper [7], the authors present a machine learning model that uses a fine-tuned version of the MobileNet classifier to classify images of strawberries and cherries into different fruit types. The authors used a dataset of annotated images of strawberries and cherries, and fine-tuned the MobileNet classifier on this dataset using transfer learning.

The authors found that the fine-tuned MobileNet classifier achieved an accuracy of 96.7% on the test set, which consisted of images of fruit types that were not included in the training set. The authors also compared the performance of the fine-tuned MobileNet classifier to a baseline model that was trained from scratch on the same dataset, and found that the fine-tuned MobileNet classifier outperformed the baseline model by a significant margin.
The authors concluded that the fine-tuned MobileNet classifier is an effective and efficient tool for classifying images of strawberries and cherries into different fruit types. They suggested that the model could be used in applications such as fruit sorting and quality control in the agriculture industry.

The paper[8] mentioned is about using machine learning algorithms for the task of classifying the ripeness of oil palm fresh fruit bunches (FFBs). The authors collected a dataset of images of oil palm FFBs that were labelled as either "ripe" or "unripe," and used this dataset to train and evaluate the performance of several different object detection algorithms. These algorithms included the Single-Shot Detector (SSD), the Faster R-CNN, and the RetinaNet.

2.4 Faster-R-Convolutional Neural Network (RCNN)

Faster R-CNN is based on the 2015 paper "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", which improves on the basis of R-CNN (Regions with CNN features). R-CNN is a method for object detection that finds the object by sliding a window in the image, and then uses CNN to extract the features of the object. However, R-CNN is less efficient because sliding a window over the entire image is computationally intensive. The main contribution of Faster R-CNN is the introduction of a new network structure called Region Proposal Network (RPN). It can extract the candidate target area in the input image, so that it can avoid sliding the window on the whole image to detect the target, which greatly improves the speed of target detection. And on this basis, the structure of Faster R-CNN is also more concise and clear, easy to understand. Due to the high efficiency and accuracy of Faster R-CNN, it has become one of the classic and widely used algorithms in the current target detection field, and is widely used in pedestrian detection, vehicle detection, target tracking, face detection and other scenarios. It also has many variants and improved versions such as FPN (Feature Pyramid Network), Cascade R-CNN, more efficient R-FCN, RetinaNet with hierarchical division, etc., which are widely used and researched. These algorithms are improved and improved on the basis of Faster R-CNN. Currently, these algorithms are published in many top conferences and journals, and have achieved remarkable results in practical applications. As shown as Figure 2 below:

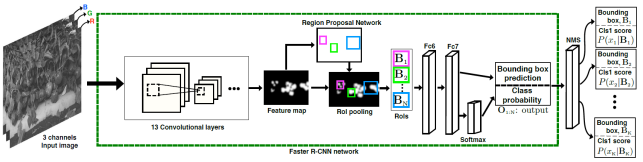


Figure 2: Faster R-CNN network

Fruit detection using Faster R-CNN is a common task in computer vision for the agricultural industry. The goal of this task is to detect and classify different types of fruits in images or videos, which can help with crop monitoring, yield estimation, and automated harvesting.
Faster R-CNN is well-suited for this task because it can detect objects of various sizes, shapes, and orientations in an image. The Region Proposal Network (RPN) in Faster R-CNN can generate candidate object regions, which is useful for detecting fruits of different shapes and sizes. Once the regions are generated, the CNN features are extracted from the regions and passed through a classifier to perform the classification.
To train a Faster R-CNN model for fruit detection, a dataset with annotated images of fruits is required. These images should be labelled with bounding boxes and class labels for each fruit. Once the model is trained, it can be applied to new images to detect and classify fruits.
Fruit detection using Faster R-CNN has been widely applied in practice and has demonstrated good results. This can greatly improve the efficiency and accuracy of fruit detection and help farmers increase their yield and save costs.

With the rapid development of Faster R-CNN in fruit detection, many scholars have improved and improved the detection effect to varying degrees based on Faster R-CNN, all showing the applicability of the model in the field of fruit detection. Researchers like Inkyu Sa[9] have employed transfer learning to adapt the Faster R-CNN model for fruit detection using images captured using two different modalities, colour (RGB) and Near-Infrared (NIR). They investigated different methods of integrating the information from both modalities by experimenting with early and late fusion techniques. This resulted in the development of a new multi-modal Faster R-CNN model, which outperformed previous methods in terms of F1 score (which considers both precision and recall) for sweet pepper detection, achieving an improvement from 0.807 to 0.838. In the field of coconut picking, scholars such as L.G. Divyanth [10] incorporates an attention mechanism into the Faster R-CNN model. The images captured were of tree crowns under different lighting, backgrounds, and coconut varieties. The proposed model was trained, validated and tested using 900 images which were manually acquired and augmented. When evaluating the model on the test dataset, the proposed model achieved a mean average precision (mAP) of 0.886 and a weighted mean intersection over union (wmIoU) of 0.827. Nikolas Lamb[11] scholar proposed a low-cost strawberry detection system based on convolutional neural networks. Ablation studies are presented to validate the choice of hyperparameters, framework and network structure. Other modifications to the training data and network structure are discussed to improve accuracy and execution speed, such as input compressed image tiling, colour masking, and network compression. Finally, we show the final network implementation on a Raspberry Pi 3B, demonstrating a detection speed of 1.63 frames per second and an average precision of 0.842.

3. Implementation

3.1 YOLOv7

YOLO is a popular real-time object detection system. It can detect a wide variety of objects, and is known for being fast and accurate. Having gone through several iterations, YOLOv7 is the latest version of the YOLO system, and includes improvements such as a larger network, improved training techniques, and a new object detection architecture. So, here I chose YOLOv7 to count and detect the ripeness of strawberries. The source code of the YOLOv7 paper (2) was directly used for this project (github link: https://github.com/WongKinYiu/yolov7).

3.1.1 Dataset preparation

First of all, the original image size is too large to uploading and calculation, so I used OpenCV to resize the image size to 502*380, the images are still very clear to detect. Then I divided the images into 3 parts including 1930 train data, 486 validation data, and others are test data. The corresponding binding boxes were divided as labels. For YOLO, we need to put the images and labels in the same file, then the algorithm could find the images and labels. And the dataset was uploaded to google drive (link: https://drive.google.com/drive/folders/1EGbp4Mk4_x72fdcdeKs_e5pIGwXw4EyR?usp=share_link). Google Colab provides free GPU, so I'll run YOLOv7 on Google Colab. This Colab notebook link is (link: https://colab.research.google.com/drive/1Gs87dvaPgEQoRqB8S53bXvXuNNSi11I9?usp=share_link).

3.1.2 Train dataset

We also need to modify some files before we train our own dataset. In file "yolov7/data/coco.yaml", we need to set the train and val data path, number of classes, and class names. In file "yolov7/cfg/training/yolov7.yaml", we need to change the number of classes. Then we would download the smallest original weights "yolov7-tiny.pt" used for training. After that, we can input our parameters to train our dataset, here the recommended epochs number is 300 but we set it to 100 because of the run time limitation. The parameters were shown below.

```
!python train.py --workers 8 --device 0 --batch-size 32
--epochs 100 --data /content/yolov7/data/coco.yaml --img
384 384 --cfg /content/yolov7/cfg/training/yolov7.yaml
--weights /content/yolov7/yolov7-tiny.pt --name yolov7
--hyp data/hyp.scratch.p5.yaml
```

3.1.3 Test model

After the training process, we got our model with weights "best.pt". And we can use this to test our test data. After inputting these parameters below, we would get the test results.

```
!python detect.py --device 0 --weights
/content/yolov7/runs/train/yolov72/weights/best.pt --conf
0.25 --img-size 384 --source
/content/drive/MyDrive/myData/images/test2
```

3.2 SSD MobilenetV2

The Single-Shot Detector (SSD) is a popular choice for object detection tasks because it is fast and accurate. SSD is a type of deep neural network that is designed to be able to detect objects in images in a single pass, without the need for any additional processing. This makes it well-suited for real-time applications, where it is important to be able to process images quickly.
Additionally, SSD is relatively simple to implement and does not require a lot of computational resources, making it a good choice for resource-constrained environments. It is also able to handle a wide range of object sizes and aspect ratios, making it a versatile choice for a variety of object detection tasks.Overall, the main advantage of SSD for object detection is its speed and accuracy, making it a good choice for real-time applications and situations where computational resources are limited. (Link: https://techzizou.com/training-an-ssd-model-for-a-custom-object-using-tensorflow-2-x/)

3.2.1 Dataset preparation

For running SSD we need the annotations in the PASCAL VOC format where each annotation is expressed in a XML format. The bounding box format of the file is converted into the XML format. For using the tensorflow, these records are converted into TF Records. (Link to dataset:

https://drive.google.com/drive/folders/1tWQN2G7zPbUOci8-WhE8979W380tusqs?usp=share_link)

3.2.3 Training dataset and Testing Model

We also need to modify some files before we train our own dataset. In the "configuration file", we need to set the train record and test record data path, number of classes, and class names and we need to change the number of classes. After that, we can input our parameters to train our dataset, here the recommended epochs number 500 .Also, we have used only 100 images to train the model due to environment and hardware constraints.After the training process, we got our model and we can use this to test our test data. https://colab.research.google.com/drive/1GkNjwmbfjJlIbgnSkuOi5-4ozOOCLJaF?usp=sharing

3.3 Faster R-CNN

Faster R-CNN is a target detection model that mainly consists of two parts: one is Region Proposal Network (RPN) and the other is Fast R-CNN.

Region Proposal Network (RPN): RPN is a convolutional neural network-based network structure that is used to generate a large number of a priori boxes (proposals), which are regions that may contain targets. Fast R-CNN: Fast R-CNN is a network structure used to classify and locate these prior boxes. The a priori frame and the original image after the RPN prediction enter Fast R-CNN, and in the Fast R-CNN, the a priori frame is intercepted from the original image through the RoIPooling layer as a feature map, and then the feature map is input to the multi-layer volume Classification and localization in the product neural network. Overall network structure: Faster R-CNN is a network structure that combines RPN and Fast R-CNN. When training Faster R-CNN, RPN and Fast R-CNN are trained together. RPN is responsible for generating a priori frame, Fast R-CNN is responsible for classification and positioning, and both contribute to the performance of the model.

3.3.1 Data Preparation

The same as Wan, I divided the images into 3 parts including 1593 train data, 183 validation data, and others are test data. But since the model I'm using converts the data into pascal_voc_data, which helps to train the model. So I wrote a script to convert the text of the bounding box of each strawberry instance provided into the corresponding xml file. The final dataset has been uploaded to Google Drive: https://drive.google.com/drive/folders/1--4FJ7KUvWsDUUEfC2MsJV0wRP6VeZ-G?usp=share_link
In addition, before the model training, I used some data (250 doubled pictures) to select the hyperparameters of the model training on my own computer. In order to speed up the training, I uploaded the trained hyperparameters and model codes to Google colab and used the limited-time free GPU for calculation.

3.3.2 Code Implementation

My code for Faster R-CNN and RPN is based on the 1st-place winning entries in several tracks in ILSVRC and COCO 2015 competitions. I have modified some functions based on the code, such as modifying the data import method that is more suitable for our project when importing data, and modifying the hyperparameters and image size. Finally, after uploading the code to Google's colab, you can get the result by calling the data and then running it. https://github.com/chenyuntc/simple-faster-rcnn-pytorch

3.3.3 Validation train model

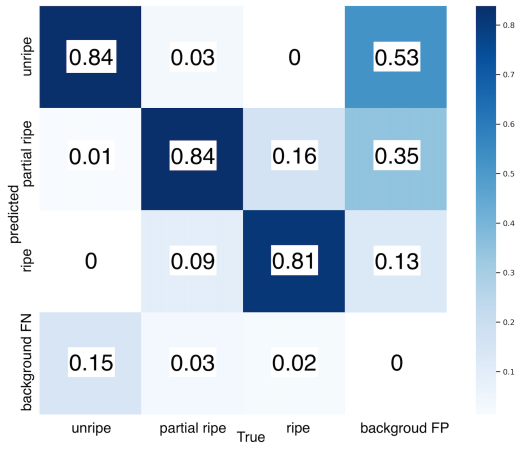
Since the training set has been allocated above, there is still a part of the verification set that will verify the fitting results of the model in this part.

4. Results

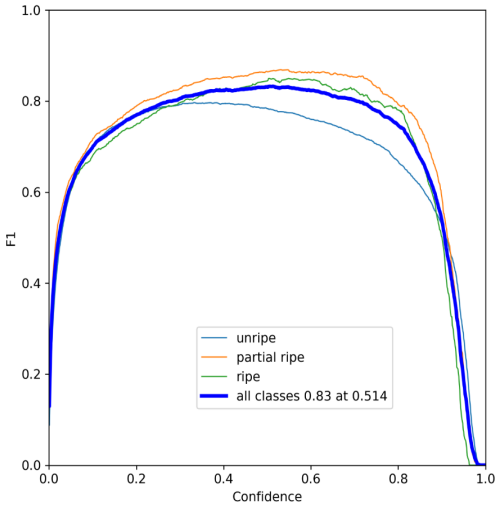
4.1 Model performance

4.1.1 YOLOv7

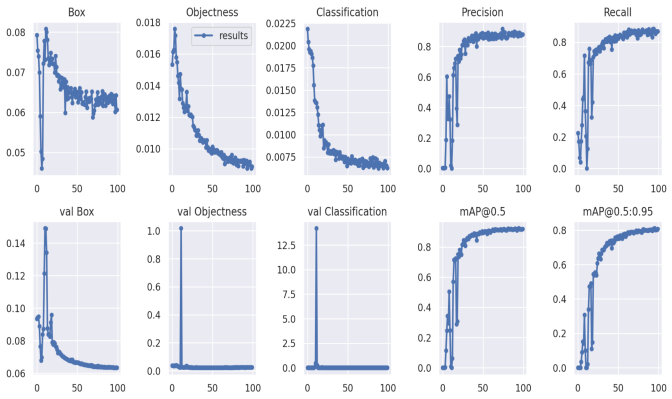
Our dataset training by YOLOv7 with 100 epochs completed in 2.989 hours. And the model we got contained 415 layers, 37207344 parameters, and 37207344 gradients. As figure 3 shows, the overall precision is above 0.8. Some mis-detection is wrong ripeness, with some false positive results (means background objects were misjudged as strawberry). It didn't reach the best performance of YOLOv7, maybe because of the limitation of GPU and epochs. But the speed and accuracy are satisfied here.



(a) Confusion matrix



(b) F1 score



(c) other results

Figure 3: The performance of YOLOv7 trained by our dataset

4.1.2 SSD

Our dataset training by SSD with 500 epochs completed in 4 hours and using only 100 images due to hardware constraints. As figure 3 shows, the overall precision is nearly 0.8. Some mis-detection is incorrectly identifying the ripeness along with some of the false positive results where the background objects were misjudged as a strawberry.

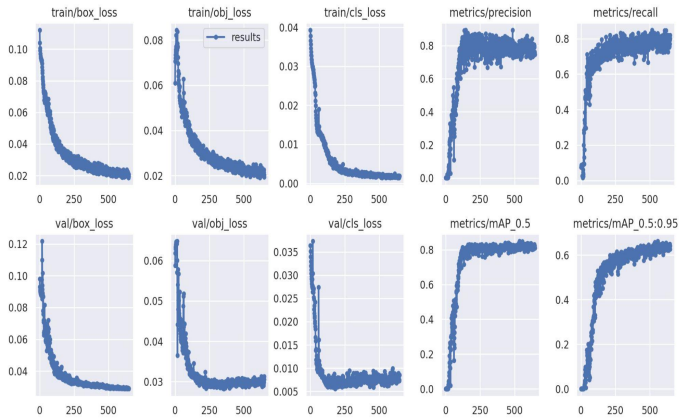


Figure 4: The performance of SSD trained by our dataset

4.1.3 Faster R-CNN

Our dataset training by Faster R-CNN with 60 epochs completed in 4.2 hours. As figure 5, 6 shows, The loss of the test set gradually decreases as the number of iterations increases. After 600 iterations, the result hovers around 0.11. Due to the time limit of google GPU, there is no way to continue training. The best results are not achieved here, and it can be seen that the effect of the validation set is not very good. But this result can also be used as a reference to a certain extent.

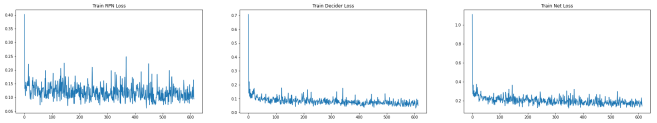


Figure 5: Loss of Training

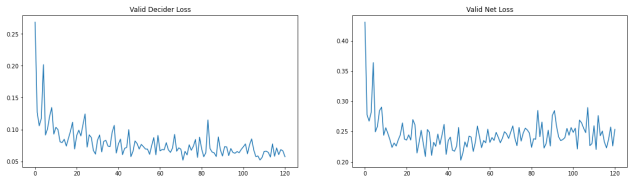


Figure 6: Loss of Validation

4.2 Performance on test images

The provided 5 test images for group 5 were tested by our three trained models, the results are shown in Figure 5, 6 and 7. Others are summarised in table and Individual Reports. It's obvious that, except YOLO, not all of the strawberries were found in these models and some overlapping strawberries were not well divided. The overall results of some models are satisfied.

4.2.1 YOLOv7

(a) 1360.png: 5 Unripe and 1 Fully Ripe

(b) 1367.png: 7 Unripe and Partially Ripe

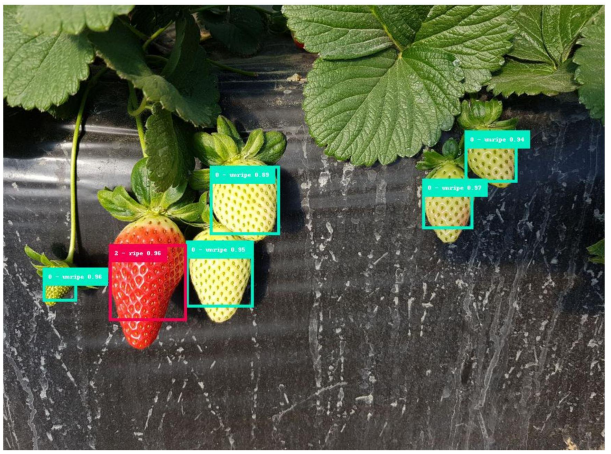
(c)1426.png: 4 Unripe, 2 Partially Ripe and 2 Fully Ripe

(d) 1478.png: 4 Unripe and 1 Partially Ripe

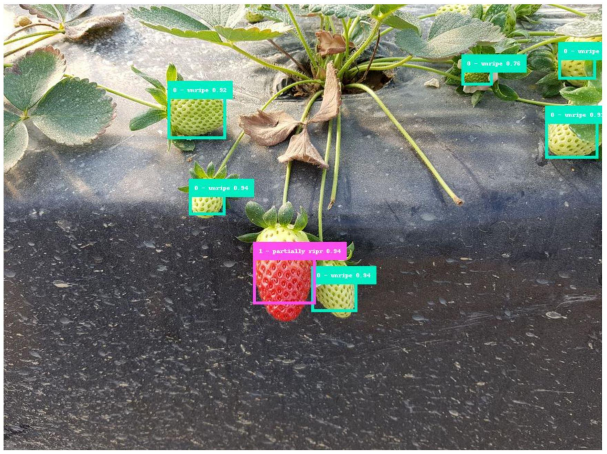
(e) 1531.png: 9 Unripe and 2 Fully Ripe

Figure 5: Performance of the YOLOv7 model on the 5 test images

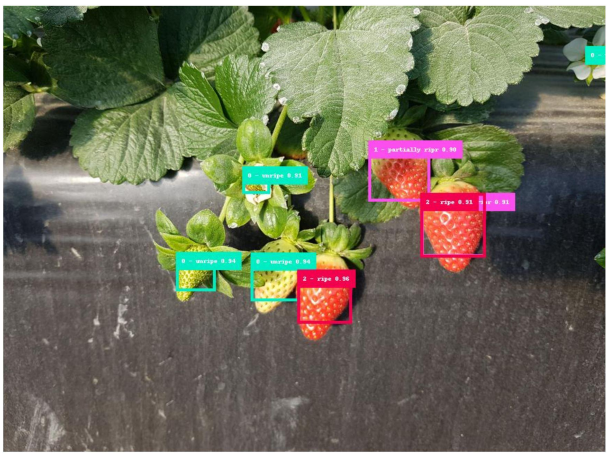
4.2.2 SSD



(a)1360.png: 5 Unripe and 1 Ripe



(b) 1367.png: 6 Unripe and 1 Partially ripe



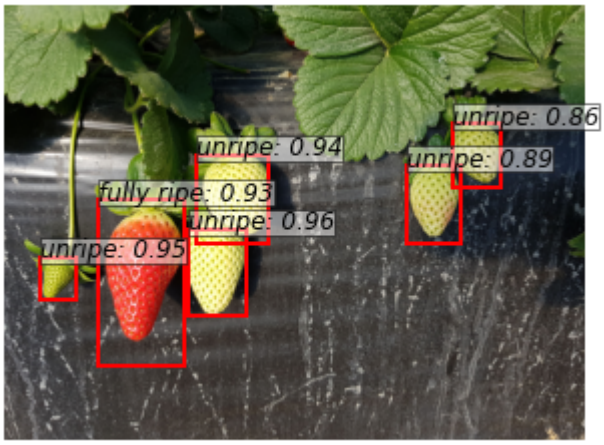
(c)1426.png: 3 Unripe, 1 Partially ripe and 2 Fully ripe

(d) 1478.png: 3 Unripe, 1 Partially Ripe

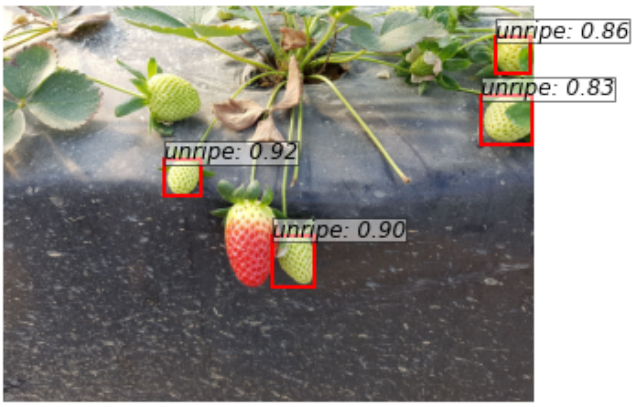
(e) 1531.png: 6 Unripe and 1 Fully ripe

Figure 6: Performance of the SSD model on the 5 test images

4.2.3 Faster R-CNN

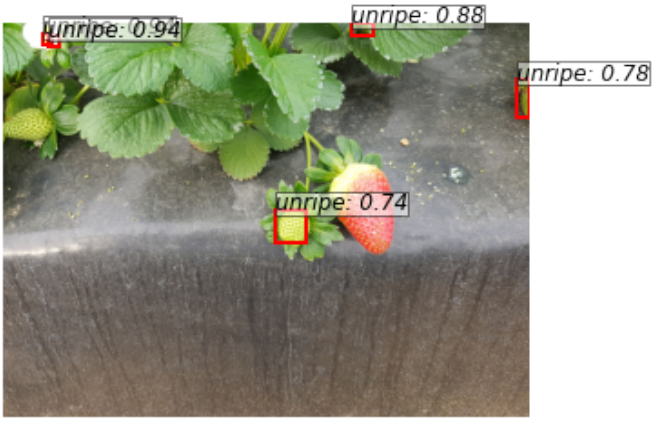


(a) 1360.png: 5 Unripe and 1 Fully Ripe

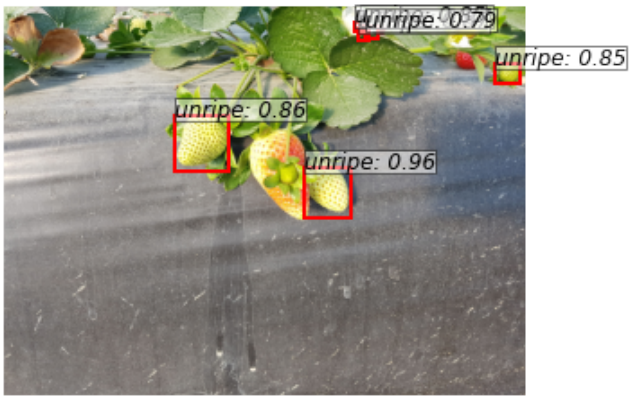


(b) 1367.png: 4 Unripe

(c)1426.png: 3 Unripe, 2 Fully Ripe



(d) 1478.png: 5 Unripe



(e) 1531.png: 5 Unripe

Figure 7: Performance of the Faster model on the 5 test images

4.2.4 Comparison of the results of the three models

The results of the comparison of the three models for this task are somewhat unexpected. From the perspective of time, the time for each iteration of the SSD is the shortest. From the training results, the results of SSD and YOLO are better. However, Faster R-CNN may not have reached the optimal moment due to insufficient training. From the training results, the results of SSD and YOLO are better. However, Faster R-CNN may not have reached the optimal moment due to insufficient training. The specific results for the five test charts assigned to Group 5 are shown in the table below:

Table 1: 5 test results of 3 different methods

Test Images	YOLO	SSD	Faster R-CNN
1360.png	5 unripe, 1 Fully ripe	5 unripe, 1 Fully ripe	5 unripe, 1 Fully ripe
1367.png	7 unripe, 1 partial ripe	6 unripe and 1 partially ripe	4 unripe
1426.png	4 unripes, 2 partial ripes, 2 ripes	3 unripe, 1 partially ripe and 2 Fully ripe	3 unripe, 2 fully ripe
1478.png	4 unripe, 1 partial ripe	3 unripe, 1 partially ripe	5 unripe
1531.png	9 unripe, 2 ripes	6 unripe and 1 Fully ripe	5 unripe

5. Limitations, Conclusions and Future Work

5.1 Limitations

Faster R-CNN is a two-stage approach, first it needs to generate a set of region proposals then classify each proposal to determine if it contains an object or not. YOLOv7 and SSD are one-stage approaches, they could directly predict the object bounding boxes and class probabilities in a single forward pass of the network. Due to the two-stage architecture, faster R-CNN requires more computational resources, so the efficiency is much lower compared to the one-stage architecture including YOLO and SSD. And YOLOv7 and SSD might struggle when detecting objects in images with significant occlusion, where some parts of the object are covered or hidden by other objects, it is also challenging when meeting changing lighting conditions or partial visibility of objects. As we know, YOLOv7 is a very fast detection model that is able to detect a wide range of objects in real-time. However, for SSD, it's fast but the large number of default boxes reduces its efficiency, making SSD less suitable for real-time applications or for use on devices with limited computational resources.

5.2 Conclusion

Based on the results achieved on the test images, we can conclude that YOLOv7 outperforms SSD and Faster-RCNN. It took the least amount of time to train the model and utilised the most of the provided dataset. The YOLOv7 model makes the least mis-detections and is able to identify the strawberries correctly.

SSD is also able to have good accuracy but the bounding boxes obtained in such cases are not as good as compared to the other two methods. Faster-RCNN performs better than R-CNN where it identifies more strawberries. Generally speaking, the accuracy rate of the Faster R-CNN model should be at least similar to that of the SSD model. The reason we received such a result may be that the model training of the Faster R-CNN is not complete, resulting in the model not reaching the optimal effect.

5.3 Future work

All three models have a certain degree of influence in the field of fruit detection. For the recognition effect, there are still some models that can achieve satisfactory results under the influence of many errors, which shows the development prospects of these three models in this field. Under this area more futur-istic work can be performed as there is a lot of scope under various parameters in this area which have not been taken under study till now. We believe that through better machine performance and more sufficient data volume, we will be able to achieve more accurate performance.

References:

[1] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
[2] Wang, Chien-Yao, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors." *arXiv preprint arXiv:2207.02696* (2022).
[3] An, Qilin, et al. "Real-Time Monitoring Method of Strawberry Fruit Growth State Based on YOLO Improved Model." *IEEE Access* 10 (2022): 124363-124372.
[5] Yu, Yang, et al. "Real-time visual localization of the picking points for a ridge-planting strawberry harvesting robot." *IEEE Access* 8 (2020): 116556-116568.
[6] Anraeni, Siska & Indra, Dolly & Adirahmadi, Desrial & Pomalingo, Suwito & Sugiarti, & Mansyur, St Hajrah. (2021). Strawberry Ripeness Identification Using Feature Extraction of RGB and K-Nearest Neighbor. 395-398. 10.1109/EIConCIT50028.2021.9431854.
[7] Venkatesh, N. Y, S. U. Hegde and S. S, "Fine-tuned MobileNet Classifier for Classification of Strawberry and Cherry Fruit Types," 2021 International Conference on Computer Communication and

Informatics (ICCCI), Coimbatore, India, 2021, pp. 1-8, doi: 10.1109/ICCCI50826.2021.9402444.
[8] Mansour, M.Y.M.A., Dambul, K.D., Choo, K.Y., 2022. Object Detection Algorithms for Ripeness Classification of Oil Palm Fresh Fruit Bunch. International Journal of Technology. Volume 13(6), pp. 1326-1335
[9] Sa I, Ge Z, Dayoub F, Upcroft B, Perez T, McCool C. DeepFruits: A Fruit Detection System Using Deep Neural Networks. Sensors. 2016; 16(8):1222. https://doi.org/10.3390/s16081222
[10] Divyanth LG, Soni P, Pareek CM, Machavaram R, Nadimi M, Paliwal J. Detection of Coconut Clusters Based on Occlusion Condition Using Attention-Guided Faster R-CNN for Robotic Harvesting. Foods. 2022; 11(23):3903. https://doi.org/10.3390/foods11233903
[11] N. Lamb and M. C. Chuah, "A Strawberry Detection System Using Convolutional Neural Networks," *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 2515-2520, doi: 10.1109/BigData.2018.8622466.