YOLO Models for Fresh Fruit Classification from Digital Videos

Yinzhe Xue

A thesis submitted to the Auckland University of Technology

in partial fulfillment of the requirements for the degree of

Master of Computer and Information Sciences (MCIS)

2023

School of Engineering, Computer & Mathematical Sciences

Abstract

With the development of deep learning and computer vision, daily activities are rapidly being replaced by computers or robots, for example, unmanned vehicles. Recently, with the advanced Microsoft Copilot, even coding work can be replaced by machines, there's no doubt that our lives become more and more convenient based on that technological development.

In this thesis, we conduct research work on fruit freshness detection from digital images. In the first step, we take use of YOLOv6, YOLOv7, and YOLOv8 to detect a variety of fruits from digital images, this can incredibly improve the efficiency and accuracy of fruit classification compared with human work. To generate the final outcomes, we train the three architectures individually, and we adopt a majority vote to get a better performance for fresh fruit detection. Compared with the previous work, our clustering method has higher accuracy and is faster than the previous architectures. Because we are use of the clustering method to generate our final results, it will be easy for us to change the backbone and get a better result in the future.

Pertaining to the dataset, we selected a dataset from Kaggle which includes 6 classes for the fruits, namely, "fresh apple", "fresh banana", "fresh orange", "rotten apple", "rotten banana", and "rotten orange". Based on this dataset, the proposed deep learning models are able to detect visual objects from digital images taken by ourselves, which can be easily fine-tuned with a larger dataset in the near future.

Keywords: Freshness detection, YOLOv6, YOLOv7, YOLOv8, orange, apple, banana

Table of Contents

List of Figures

List of Tables

List of Algorithms

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgments), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

[illegible] Date: 18 October 2023

Acknowledgment

First of all, I would like to thank my parents for their financial support. Owing to their unselfish and generous sponsors from them, I have this invaluable opportunity to complete my Master's study with the Auckland University of Technology (AUT), New Zealand.

I would like to express my deepest gratitude to my supervisor Dr Wei Qi Yan. In this study, he not only provided me with professional knowledge support and careful guidance, but also helped me enrich my learning experience. I believe I could not complete my project without Dr Yan's supervision and instructions. Meanwhile, I also appreciate the school's administrators of AUT for their invaluable guidance.

Yinzhe Xue

Auckland, New Zealand

October 2023

Chapter 1

Introduction

This chapter is composed of five parts: The first part encompasses the background and motivations, the second part includes the research question, followed by the contributions, objectives, and structure of this thesis.

1.1 Background and Motivation

Fruit freshness detection is a topic that attracts a lot of attentions in computer science which is also a very important task for human lives, since in our daily life, we need to know which food is safe, which is pathogenicity. Rotten foods may cause terrible food poisoning and always have a very bad taste, eating those food may cause some serious health issue, such as acute gastroenteritis or even death, and normally it is hard for a person to identify if a fruit is rotten or not. Our human developed many ways to detect or make a prediction of freshness, we conduct visual object detection based on the colors of fruits. It is also possible for people to detect watermelon quality just by tapping on the watermelon that looks like magic (Zou, 2023). In a factory, it is impossible to hire a lot of "fruit experts" to classify a substantial number of fruits every day , but identify the freshness of the fruit is very important for society, selling rotten food will not only loss the brand credibility but also will cause some lawsuit from users and resulting in large amounts of compensation, for the above reasons, our community really needs an automated method to identify fruit freshness.

The first choice of pattern classification (Yu, 2019) was that we make use of a computer to detect fruit freshness, but for fruit freshness detection, the method may be restricted, because for now, a computer can't smell or taste the fruit (maybe in the future, it is possible). In this thesis, we plan to use the contactless method to conduct fruit freshness classification, thus the most straightforward way is to conduct object detection based on digital images of fruit. Through these image data, we expect a computer can classify fresh or rotten fruits based on the given fruit images by using computer vision (Zhao, 2019; LeCun, 2013; Matsugu, 2003; Sa, 2016; Wu, 2020). From the previous method, the traditional way is to use the shape or color to classify the rotten or fresh fruit, these method is running fast and performs well for some kind of fruit, like banana, grapes and so on, because those fruits will change their color or shape when they are rotten, but for some other fruit, like apple or watermelon, it is hard to identify if the fruit is rotten only based on color or shape; recently, with the development of convolution neural network, we can use a new method to detect the rotten fruit based on the features from the image, features include color and shape, and will also include some information hard to describe from the image by our human.

In this thesis, we collect a considerable number of deep learning models for fruit freshness detection. YOLO is a very famous architecture that can be proffered for almost all types of pattern classification and object detection tasks (Zhao, 2019, Krizhevsky, 2017). Meanwhile, we also introduce those potential methods for this thesis (for example, the hottest “Transformer”) (Chen,2021). In this project, we have advanced methods (Chen, 2022; He, 2021; Moghaddam, 1995; Krizhevsky, 2012) for fruit classification.

1.2 Research Questions

In this thesis, we mainly concentrate on deep learning methods (like YOLO) (Li, 2022) to classify digital images for fresh or rotten fruits, we will apply three YOLO architectures and compare their outcomes, our research questions are:

(1) Is it possible to use YOLO to detect fruit freshness?

(2) Which architecture is the fastest one?

(3) Which model has the highest accuracy in fruit freshness detection?

(4) Which kind of majority vote method has the best performance?

The main idea of this thesis is to verify whether YOLO models can be employed to detect the freshness of the given fruits and find out the YOLO version with the best performance. There are multitude of the state-of-the-art methods that can be proffered for this project, but caused by the time limit and background knowledge, we only take advantage of YOLO models.

1.3 Contributions

The focus of this thesis is to detect or classify fresh or rotten fruits from the input images. According to the most advanced YOLO models, it will be easy for us to get high precision and recall for fruit freshness detection even compared with the human labeling method. By the end of this thesis, we are able to:

(1) Collect a large dataset for 3 different types of fruit (apple, banana, and orange)
(2) Process each image with YOLOv6, YOLOv7, and YOLOv8.
(3) Have a general understanding of the Transformer.
(4) Prove machine learning methods to detect the freshness of the fruit.
(5) Find an ensemble method to combine the detection result from different architectures and obtain the best clustering weights for different architectures.

Meanwhile, the core method for this thesis will be using different YOLO models to classify a variety of fruits, we take use of different types and different augmentation images for our experiments.

1.4 Objectives of This Thesis

Thanks to the development of machine vision, there are a large volume of methods to conduct fruit freshness detection based on digital images, for example, AlexNet (Krizhevsky, 2017), Fast R-CNN (Girshick, 2015), Faster R-CNN (Ren, 2015), Autoencoder (Zhang, 2016), Transformer (Chen, 2021, Cheng, 2022), the models already got the state-of-the-art outcomes. Considering the utility and convenience in the future, we plan to use YOLO (Redmon, 2016) models for this project. The reason is that YOLO is a light-weight architecture (can be used on a mobile device) and has a similar performance compared with Faster R-CNN or transformer, training YOLO models is also much faster and easier than training an R-CNN or transformer model. The most important thing is that YOLO is not only an architecture, but also a toolbox for image classification, detection, or segmentation. There are millions of instructions and applications, which can help us train and test YOLO models with a diversity of image datasets easily.

In this thesis, we apply YOLOv6 (Li, 2022), YOLOv7 (Wang, 2022), and YOLOv8 to conduct fruit freshness detection and classification based on digital images, all these architectures will have a similar structure and can process the same dataset, we will discuss the details about the dataset and YOLO models in the later chapters.

1.5 Structure of This Thesis

The structure of this thesis is described as follows:

(1) In Chapter 2, we show a literature review and discuss the relevant studies of visual object detection and classification (like autoencoder). Meanwhile, we also introduce the details about encoder-decoder and majority vote method.

(2) In Chapter 3, we detail our research methods and depict the details of our dataset.

(3) In Chapter 4, we implement the proposed algorithms, collect experimental data, and demonstrate the research outcomes in the form of figures and tables. Additionally, the limitations of these proposed methods will be described.

(4) In Chapter 5, we summarize and analyze the experimental results.

(5) We will the conclusion and vision of our future work in Chapter 6.

Chapter 2 Literature Review

The focus of this thesis is on fruit freshness detection based on YOLO models, this chapter will introduce plenty of methods (conventional and advanced method) and the relevant knowledge in visual object classification and detection.

2.1 Visual Object Classification

Visual object classification is a long-history issue and attracts much attention in the research area. Classification is a complex task for machine but detecting a visual object is a natural instinct for humans and all animals around the world. We have sight, smell, hearing, taste and touch feeling, but for a computer, the mainstream and available way of doing object detection is to make a classification through computer vision and image processing. The computer will decide the class of the given images based on visual features from the image (Ren, 2015; Sun, 2021); the most famous example of this kind of method is AlexNet (Krizhevsky, 2017). In the following section, we will introduce the details of AlexNet, the model has surpassed our human vision.

2.2 AlexNet

In AlexNet, convolutional layers (Galvez R L, 2018) were deployed to make visual object detection for the first time in history. In eq. (2.1), convolution is always shown as '*', *f* is the input image, *g* is the kernel,

$$[f * g](t) = \int_0^t f(\tau)g(t-\tau)d\tau \tag{2.1}$$

where *f* is the input image, *g* (·) is the convolutional kernel.

Convolutions on the time domain are a multiplication of kernels on the given windows. In digital image processing, a convolution with a kernel that will offer us special information about the image, we call the information as "features".

Before AlexNet, we only apply MLP (i.e., multilayer perception) to classify digital images, and always has low accuracy which is much lower than human vision. With convolutional layers, the net can extract visual features from the original images, as a human looks at the image from local details to the global image, this strategy incredibly increases the accuracy around 20%, finally getting a classification accuracy higher than human in 2018 based on convolutional neural network.

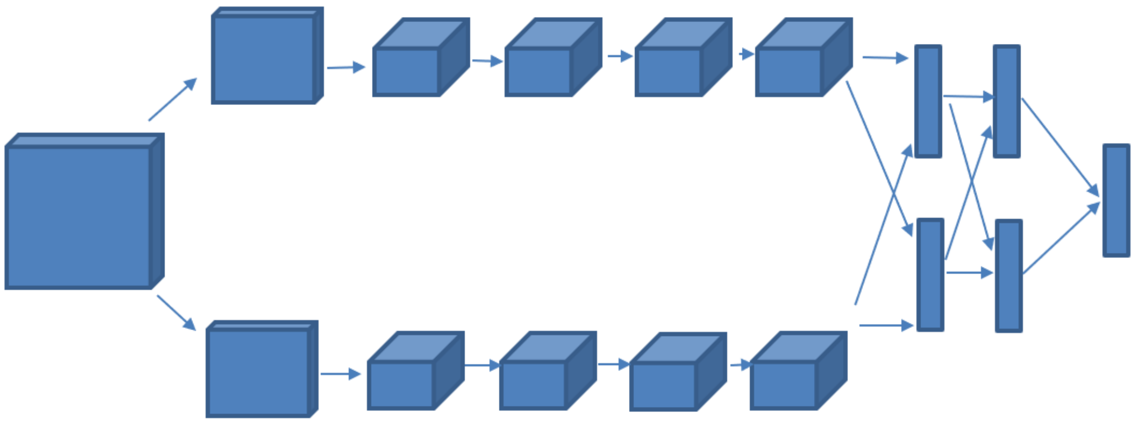


Figure 2.1 An example of the AlexNet model

In this model, the large-size boxes mean the convolutional layers. In each layer, there are multiple kernels. On the right side of Figure 2.1, the 2D squares refer to multilayer perceptron which is used to classify the features output from the previous convolution layer.

2.3 Autoencoder

After the convolutional neural networks, a new model was designed and created based on a convolutional strategy for visual object detection, which is named Autoencoder (Chong, 2017). Autoencoder is a very powerful object detection architecture, instead of using MLP to conduct the final classification, autoencoder replaces the later MLP layers to convolutional layers. The convolutional layers always look like an inverse of the input convolutional layers, we call the input convolutional layers as encoder and the convolutional layers as decoder. The encoder always conducts downsampling operations for the original input image to get visual features, the decoder always works an upsampling for the visual features to reconstruct an image with the same size as the input, the final outputs include the bounding box for visual object identification.

In general, autoencoder is employ of encoder to extract the features from the image, and then takes advantage of the decoder to conduct the classification.

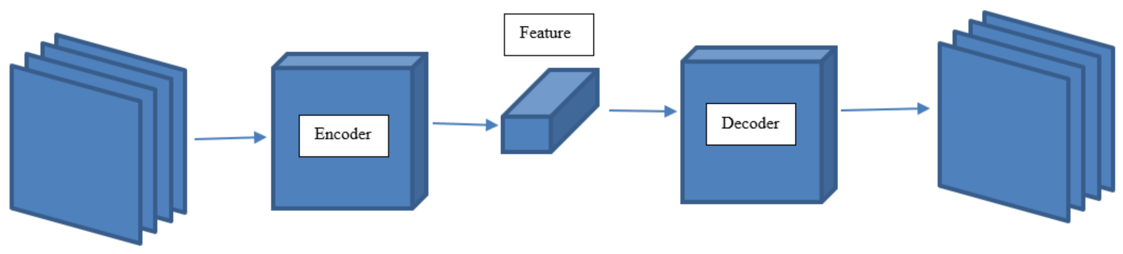


Figure 2.2 The example of autoencoder

In Figure 2.2, the multiple planes mean pre and post-convolutional layers, the blue box indicates the features we extract from the input, then we will employ it as an input for the decoder layer. Figure 2.2 shows the general architecture of the autoencoder, normally it only includes the encoder layer, feature, and decoder layer. For autoencoder, we always express the encoder as **E** and decoder as **D**, an estimation function *d* (•) to measure the error between the prediction and ground truth, then the loss function of autoencoder is,

$$L(\theta, \phi) = Ex \sim \mu_{ref} \left[d\left(x, D\theta\left(E\phi(x)\right)\right)\right] \tag{2.2}$$

where x is the input, μ is the mean function and L (•) is the loss function. The function $E(\cdot)$ indicates the expectation of the error. To train the auto-encoder, we need to minimize the loss.

2.4 Majority Vote

Majority vote is a method with very long history, compared with transformer or encoder-decoder, majority vote is very easy to use and understand, instead of use deep learning architecture, majority vote will directly combine different result, as we shown in Figure 2.6.

As shown in Figure 2.6, firstly we will have different models, and then we will use a same input as the input for each of the model, then we will collect the output from each of these models, the final result will be same as the most classes from all our results, for example, if we have 4 different outputs, then 3 of them is apple, only one of them is banana, then our final result will be apple, because over 75% predictions are apple. Through this way, if we add a new model result, we can only add this result to our result collection and directly generate the final result instead of adding the new architecture to our previous model.

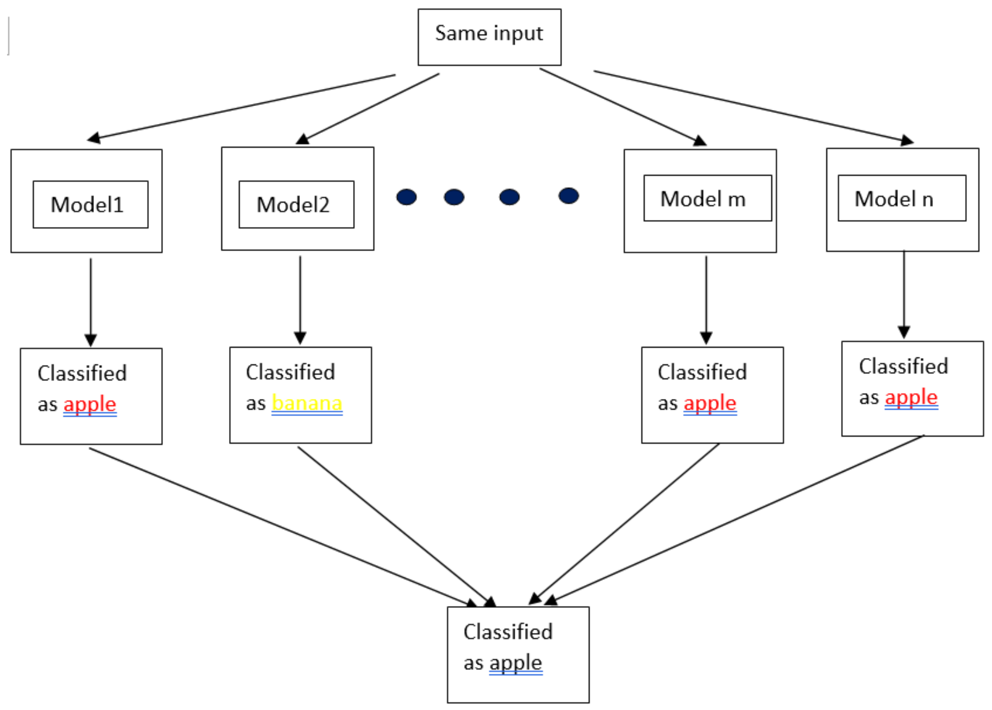


Figure 2.3 The majority vote

Chapter 3 Methodology

The main content of this chapter is to clearly articulate research methods, which satisfy the objectives of this thesis. The chapter mainly covers the details of the research methodology for fruit freshness detection by using YOLOv6, YOLOv7 and YOLOv8 models.

3.1 YOLO

YOLO (Fang, 2019; Fang, 2021; Redmon, 2021; Parico, 2021) is the abbreviation of “You Only Look Once”, which is a very famous and popularly spread deep learning model. There are a lot of advantages of YOLO models, firstly, YOLO is a “lightweight” architecture, which indicates it will be trained very fast, and the trained weights will not be stored in a large space. Secondly, YOLO can provide real-time detection of visual objects, what we mean in real-time is the network can conduct object detections based on the input image much faster and more accurate than human reaction. The network “suddenly” generates the output, though compared with Faster R-CNN network, the accuracy of YOLO is not very high, and the differences in the performance between YOLO and other architecture are pretty small. For the visual object detection task in this project, it is perfect to have a little bit lower accuracy but much faster during training and testing time.

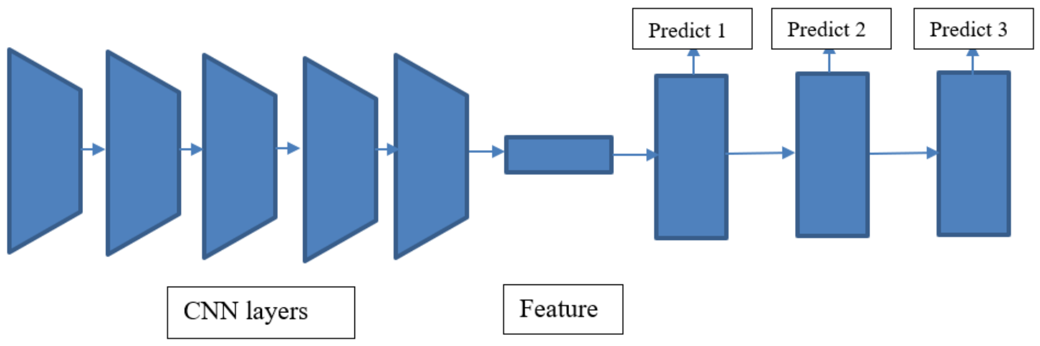


Figure 3.1 YOLO architecture

In Figure 3.1, the sequential trapezoidal is convolutional layers, the middlebox is residual layers, the right three boxes are the upsampling layer, which is employed to increase the size of the feature, the white box with “Predict” is the concatenate layer, this layer is used to concatenate the output from the previous layers. There are four very important components of YOLO:

Part 1. Residual blocks.

Part 2. Bounding box regression.

Part 3. Intersection over Union (IOU).

Part 4. Non-maximum suppression (NMS)

We will discuss each main component individually.

3.1.1 Residual Block

In the residual block, the main idea is to find the correct position of a visual object. In order to accomplish this, firstly, it will segment the given image into various regions, each region will have a dimension $N\times N$. The network will conduct visual object detection based on each grid. If an object appears in one grid, then the network will label this grid as a candidate for the final detection.

3.1.2 Bounding Box Regression

A bounding box is an indicator that always tells us where visual objects are, every bounding box has the following four most important components:

Part 1. Width of the box.

Part 2. Height of the box.

Part 3. Center of the box.

Part 4. Class information for the box.

YOLO is the network with a single-box regression to predict the four pieces of information for a bounding box. Figure 3.2 is shown as an example.

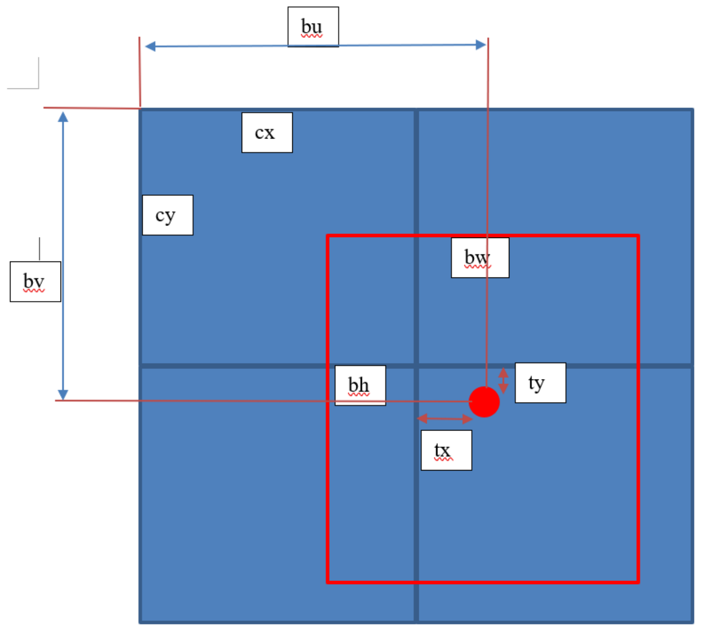


Figure 3.2 An example of bounding box.

In Figure 3.2, σ indicates the offset of bounding box, with the parameters for bounding box, we will have the following equations:

$$bu = \sigma(tx) + cx \tag{3.1}$$

$$bv = \sigma(ty) + cy \tag{3.2}$$

$$bw = p_w e^{t_w} \tag{3.3}$$

$$bh = p_h e^{t_h} \tag{3.4}$$

$$\sigma(t_o) = \mathrm{P\ r(object)IOU\ (b, object)} \tag{3.5}$$

$$B = box\ (b_u, b_v, b_w, b_h) \tag{3.6}$$

where **B** indicates bounding boxes, *IOU* is the intersection over union, we will introduce it in the next section.

3.1.3 Intersection over Union (IOU)

Intersection over union (IOU) is a concept in visual object detection that describes how the bounding boxes are overlapped. In the YOLO model, we employ IOU to provide an output box that surrounds the visual objects perfectly.

Each grid cell in Fig.3.3 is responsible for predicting bounding boxes and confidence scores. The IOU is equal to 1.0 if the predicted bounding box is as same as the real box. If we assume the blue square presents the prediction, and the red one is the ground truth. This mechanism eliminates bounding boxes that are not equal to the real box. we display an example of IOU in Figure 3.4.

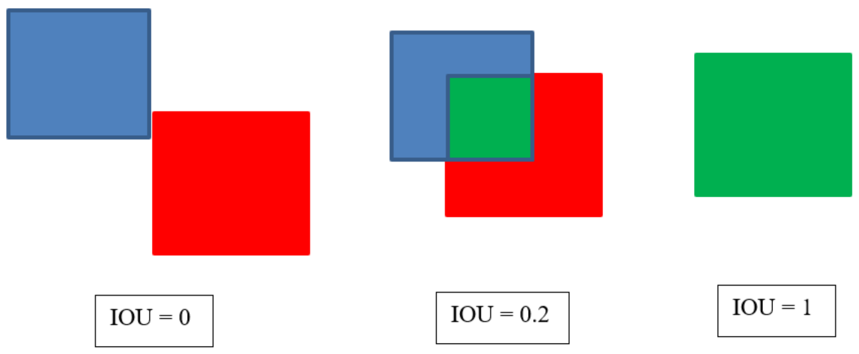


Figure 3.3 An example of intersection of union (IOU)

In Figure 3.3, the red bounding box is the ground truth, we need to conduct object detection, and the blue square is the prediction that the architecture works on the image. On the right side, the green box indicates that our prediction is overlapped with the ground truth (GT). In order to calculate IOU, we are use of equation (3.7),

$$IOU = \frac{Area\ of\ overlap}{Area\ of\ union} \tag{3.7}$$

After calculating the IOU, we set a threshold *t* to determine whether we will keep both of these detectors or only keep one of them:

Algorithm 1 Auto-labeling steps

```
if $i \le t$ then
    Keep all of the bounding boxes
else
    only keep the bounding box with the highest confidence value
end if
```

3.1.4 Non-maximum Suppression (NMS)

Normal object detection always is described by using a lot of bounding boxes around a visual object, but this kind of result will confuse the estimation functions. In order to eliminate this kind of false alarms, a method named Non-Maximum Suppression (NMS) was developed. Firstly, it will calculate the IOU between each pair of the detection bounding box, then we will eliminate the bounding boxes based on the IOU. If the IOU score is high, this indicates two bounding boxes are very close, this indicates they may detect the same object, so we can then neglect the bounding box with a lower score. An example of NMS is shown in Figure 3.4.

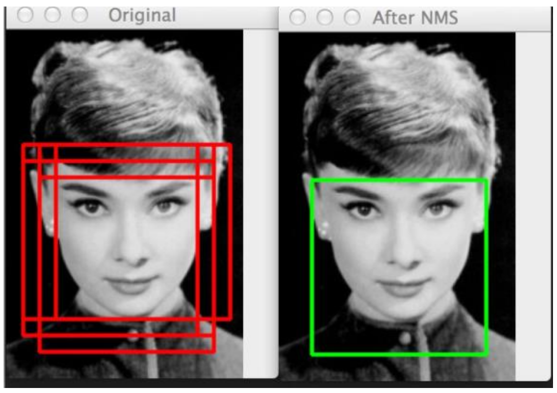


Figure 3.4 Example of NMS. (left is before NMS, right is after NMS)

The pseudocode for non-maximum suppression is shown as follows.

Algorithm 2 Non-maximum suppression

```
function $NMS(B, c)$
    $B_{nms} \leftarrow \theta$
    for $b_i$ $from$ $B$ do
        $discard \leftarrow False$
        for $b_j$ $from$ $B$ do
            if $same(b_i, b_j) \geq \lambda_{nms}$ then
                if $score(c, b_j) \geq score(c, b_i)$ then
                    $discard \leftarrow True$
                end if end
            if
            if $discard$ then
                $B_{nms} \leftarrow B_{nms} b_i$
            end if end
        for
    end for
    return $B_{nms}$
```

For the pseudocode about NMS, *B* indicate the input bounding boxes, and *I* is the ground truth. Hence, we can find the bounding box of visual objects for each image.

3.1.5 The Whole Pipeline

The whole pipeline of YOLO model is not very complicated, in this section, we introduce the architecture for the general YOLO model, because all the others just add more parts or change the backbone compared with the general one. The general architecture of YOLO is just a combination of the previous methods with deep learning models, there are four main steps in the whole architecture.

Firstly, we should resize any arbitrary image to a size of 448×448, then we forthput the second part which is the residual block. This model will help us segment the whole input image into blocks, normally, we will split the whole image into 16×16 blocks. The third part has two branches, the first one is to predict the classes for each block, we call it as a classification network. The second branch is the bounding box regression network, this part will predict the confidence for each bounding box. The final result will only keep the bounding box with high confidence. After we get the prediction bounding boxes, then we can utilize IOU to calculate the score for each two bounding boxes, we apply NMS to only keep the bounding box with high confidence and less overlap.

The final result from YOLO architecture will have three parts, the first one is the bounding box, and the second one is the confident value for the bounding box. The confidence value is from 0 to 1.0 which indicates how confident we trust there is a target in the bounding box, the last part of the output is the classification result, which will indicate the class of visual objects in the bounding box. The details about the network are listed:

(1) 24 convolutional layers, 2 multilayer perceptron

(2) The size of kernels in the first layer is 7×7, with 64 kernels.

(3) The second layer has 192 3×3 kernels.

(4) The third is 1×1×128, 3×3×256, 1×1×256, 3×3×512.

(5) The fourth is 1×1×256, 3×3×512, 1×1×512, and 3×3×1024.

(6) Then is 1×1×512, 3×3×1024, and 3×3×1024

(7) The last convolution layer is two 3×3×1024

(8) Then is the linear layer, with a size 4,096

3.2 YOLOv6

Compared with the previous architecture like YOLO or YOLOv5, YOLOv6 is utilizes almost the whole same dataset to be pre-trained, but the backbone has been changed to EfficientRep, and the neck has been changed to Rep-PAN.

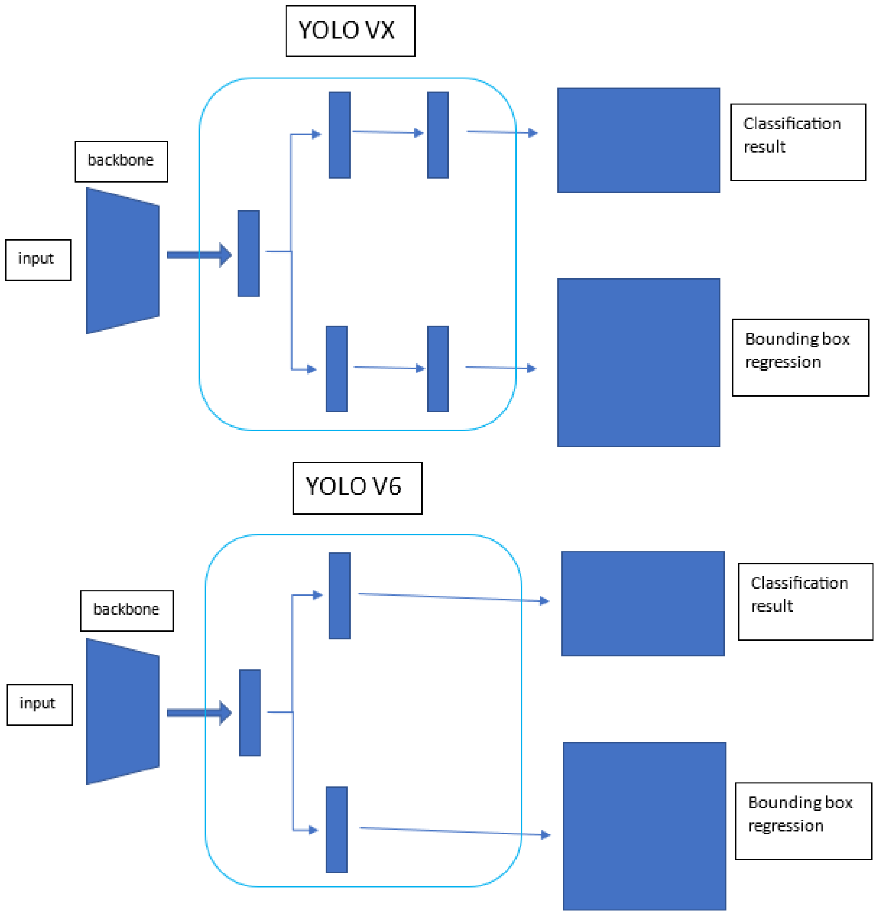


Figure 3.5 Architecture of YOLOv6 model

Figure 3.5 shows the model of YOLOv6, the input of this model is the normal RGB image, the efficient rep backbone is just multiple convolutional layers, the rep-PAN is the layer to combine the outputs from the backbone. After the rep PAN step, the network will utilize a pyramid architecture to conduct the classification and apply a regression network to generate a regression score estimation.

Compared with the previous network, YOLOv6 model added one more term in the loss function, which is a KL divergence between the predictions from teacher model and the student model, the KL divergence is,

$$LKD = KL(pals||pcls) + KL(preg||preg) \tag{3.8}$$

where *pals* is the part to make the classification, and *preg* is the part to make a bounding box prediction.

From In equation (3.8), *pcls* is the part to conduct the classification, *preg* is the part to be designed for bounding box prediction and confidence generation. KL is the abbreviation of the KL divergence, which can estimate the similarity between two different distributions, with the KL divergence, the final loss function is,

$$L_{total} = L_{det} + \alpha L_{KD} \tag{3.9}$$

where *Alpha* is a hyperparameter to leverage the KL loss, L_{det} is the loss function as same as the previous YOLO. If we add them all together, we can get the final loss. In this thesis, we use of this loss function to train the whole network.

3.3 YOLOv7

Compared with the previous YOLO architecture, like YOLOv6, YOLOv7 has a higher accuracy for multiple tasks shown in the thesis. The main difference is that YOLOv7 modifies the ELAN architecture (i.e., efficient layer aggregation network). The modified ELAN network is called E-Ellan, and it is just a combination of multiple convolutional layers and two more concatenation layers, as shown in Figure 3.6.

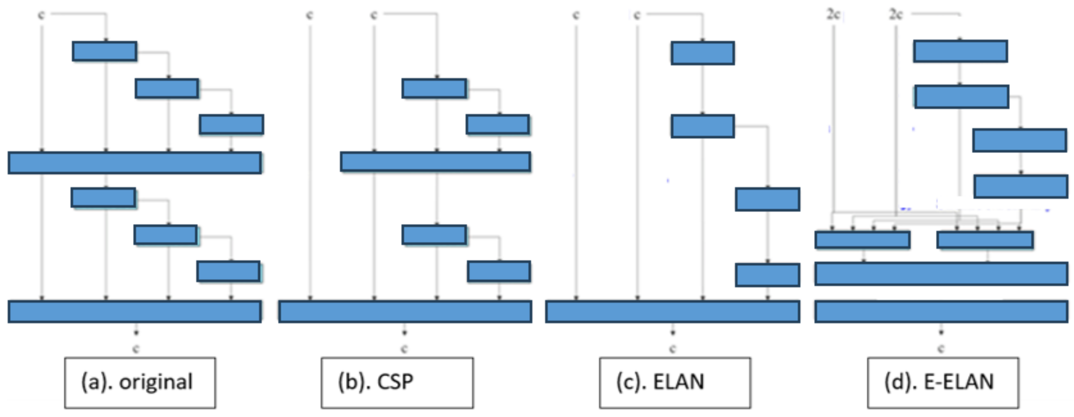


Figure 3.6 Architecture of YOLOv7

Figure 3.6 shows the new architecture added to YOLOv7 compared with the previous YOLO models., Figure 3.6(a) is the illustrates the original architecture, and CSP, ELAN and E-ELAN are the new architecture. ELAN and E-ELAN will have the best performance, compared with the previous architecture, In ELAN, a convolutional layer is deleted, so it has a faster training and testing ability. E-ELAN splits the architecture into multiple convolutional layers. This will reduce the number of parameters in the network which will help the network make a better prediction.

Compared with the previous ELAN architecture used in YOLOv6, E-Ellan changes the architecture in the computational block, but the transition layer is just kept as same as before. Although the architecture of YOLOv7 seems more complex than the general architecture of YOLO, the core components keep as same as before. The general architecture is still a backbone network, classification network and bounding box regression network. The weights for the backbone network will also be kept as same as before. In this way, the whole architecture will be pretty easy to be understood if we already know the YOLOv6 architecture, this is also an advantage for YOLO models.

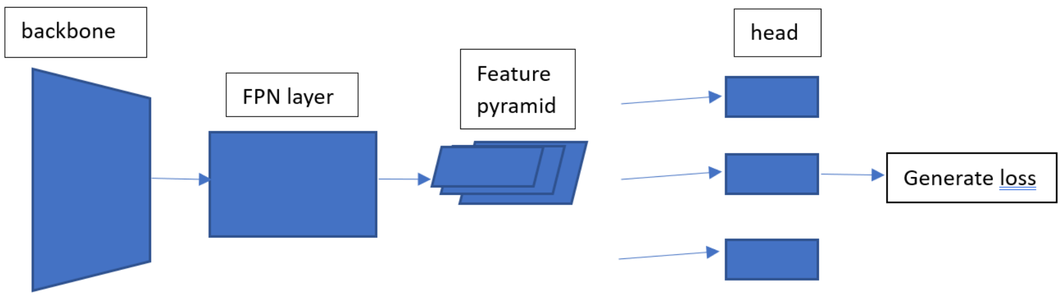


Figure 3.7 Architecture of YOLOv7 model

Figure 3.7 shows the details of the architecture of YOLOv7 model, the first backbone part is multiple convolutional layers that have a residual connection. The second part is called the “head” part of the architecture, it is just a combination of multiple MLP and convolutional layers.

3.4 YOLOv8

YOLOv8 published in 2022 is the newest version of YOLO models. Compared with YOLOv6 and YOLOv7, YOLOv8 can be installed as a Python toolbox which can directly be used employed as a Python package. YOLOv6 and YOLOv7 were created by some individuals various organizations, but YOLOv8 is was created built by a company name is ultralytics. At present, YOLOv8 is an open-source architecture model for general architecture. The architecture of YOLOv8 looks similar to YOLOv7, the only difference is YOLOv8 replaces C3 architecture with C2f architecture (Lin, 2017), compared with C3, C2f has two backbones, C3 only has one backbone. In YOLOv8, we will use different backbones for different tasks such as object detection or pattern classification, so compared with YOLOv7 or YOLOv6, YOLOv8 can be used for more different tasks. In general, YOLOv8 looks like a combination of the previous YOLO architecture, but with the C2f architecture, YOLOv8 is running much faster than the previous version of YOLO models.

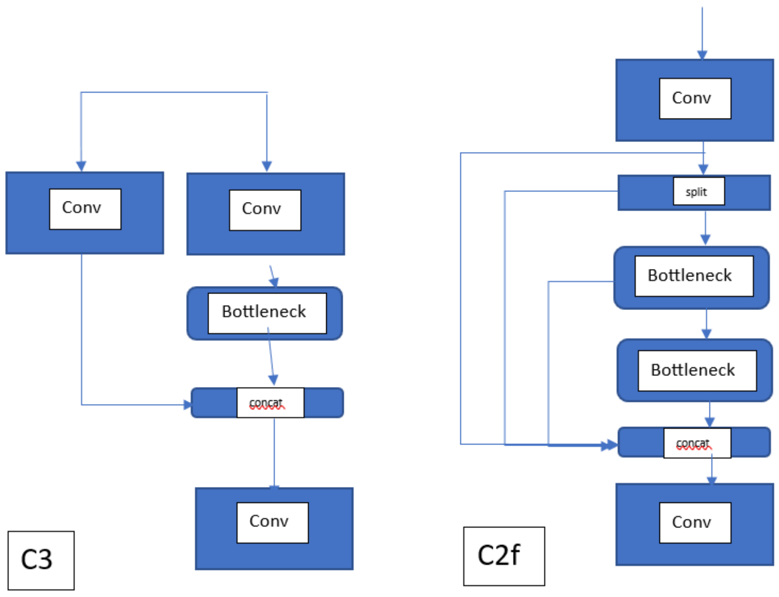


Figure 3.8 Architectures of C2 and C3f

Compared to the architecture of C3 and C2f, there are a few changes:

(1) The kernel size of the first convolution layer change to 6x6 instead of 3x3;

(2) Two convolutional layers are deleted in the neck;

(3) The number of blocks in the backbone is changed to 3-6-6-3;

(4) A split part was added before the bottleneck;

(5) The residual connections between the input and output are added;

(6) The kernel parameters have been changed;

(7) Instead of using the parallel bottleneck layers, the serial layers are connected;

(8) The head of YOLOv8 was changed to the anchor-free architecture, but the previous YOLO is anchor-based architecture.

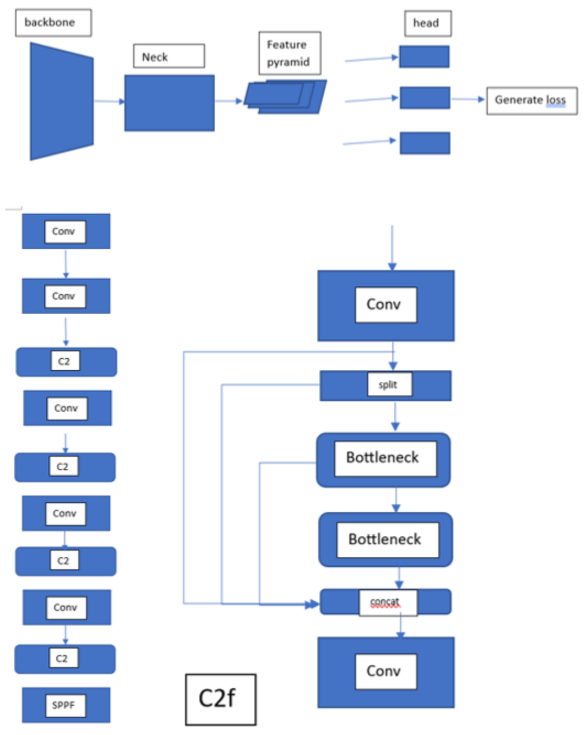


Figure 3.9 Architecture of YOLOv8 model

3.5 Dataset Preprocessing

From the original fruit dataset, we find that it only has image data for different types of images, but there's no information about labels for each image. This is a big problem, sincethe label is the most important part, without a label, the network will never know where the correct position is for the visual object. There are multiple ways to label the original image, the first method is to label it by ourselves, this is the method that will have no wrong labels, but it costs a very long time. In the whole dataset, there are around 15,000 image samples, it is impossible for us to label the whole dataset manually.

Instead of labeling manually, we write source code for programming to automatically label the images. Fortunately, all images in the dataset just have a clean background, and the foreground objects always have a color compared with the background, so we can easily utilize the color difference to find the contour of this object. We mark a bounding box for the contour as the position of the object , then we extract the average color in this area, and we assume a same kind of fruit should have a similar mean color. We will show the automatically labeled results in Figure 3.10, where the rectangle is the automatically labeled result.

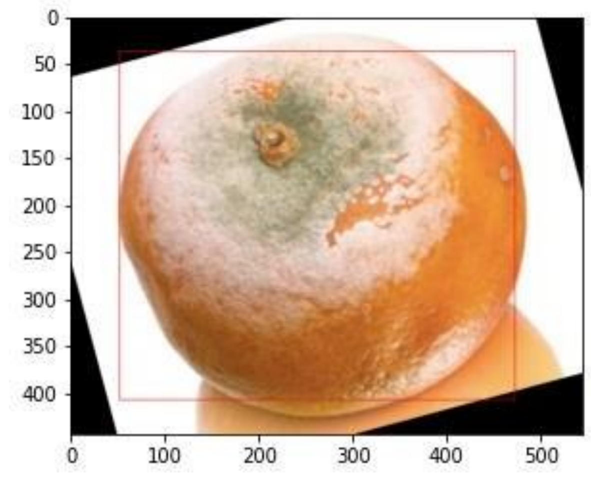


Figure 3.10 An example of auto-labeling

After the labeling, the preprocessing has been accomplished. We save all the bounding box information. This includes the coordinates of the left corner and the right corner as well as the labeling result in a ".txt" file. During the training and testing time, we can directly load this .txt file to find the information on the position of the bounding box as well as the labeling information for the object in the image. Our result correctly crops the object area inside the image, and the pseudocode for auto-labeling is shown in Algorithm 3.

Algorithm 3 IOU steps

```
Range the image
blur the image
Set a threshold to filter the contour
use cv2.findContours to find contour in the image
Draw bounding box for the contour =0
```

3.6 Combination of Different Results

We already have three different backbones, using their result individually will be a waste of the results. Pertaining to a single input image, YOLOv6 and YOLOv8 give us a correct detection result, but YOLOv7 provides us with a wrong result. For another image, the results from YOLOv7 and YOLOv8 are correct, but the results from YOLOv6 are incorrect. For the third image, YOLOv8 is wrong, but YOLOv6 and YOLOv7 make use of correct detection. If we only estimate the result individually, then the performance for each architecture will not be good. If we can combine their result, for example, we can put the three outcomes together for voting the final result, then the object detection will have a great result. Using a combination method to combine the detection result will be a good idea for object detection and pattern classification.

Through model combination, there have a lot of different ways to ensemble multiple outputs, but the most general one is the voting method, the theory of voting method is pretty, there are three steps for it:

Step 1. We get the prediction result from all different architectures,

Step 2. Based on these results, for each record, we apply the result from different architectures to vote for the classification.

Step 3. We choose the highest vote class as our final classification decision.

For example, YOLOv6 classified the first record to fresh apples, YOLOv7 classified the first record as fresh apple, but YOLOv8 classified the first record as fresh banana. Then for the first record, we choose fresh apple for 2 and fresh banana for 1, all the other types of fruit will have a vote that equals zero.

We will finally classify record one as fresh apple. Through this method, there's no doubt that we will have a better result during the testing. And in order not to make confuse, this combination method will only be used during testing.

3.7 Data Augmentation

Data argumentation is a popular method to improve network performance, it is used to add more types or more different appearance training data in the training set based on previous training datasets. We add more different types of training data that will make the network more robust. If we can get more different information from an object, it will help us better to understand the object.

For image classification with data arguments, the normal method is always related to rotating or flipping the original images. In that of the previous experiments, these two methods are very simple and can increase the performance of the proposed method. But for the network we adopted, only two kinds of argumentations may not be enough, on account of our fruit detection task, the fruit will not be put with an arbitrary angle in the image.

This case will always be happened in our daily lives, detecting fruit freshness will be happened at any time. In order to deal with this problem, we make sure that our network will have a good performance on the real dataset, we also add a blurring method in our augmentation. For this blurring method, we just easily apply a Gaussian kernel to convolute with the original image, then we will get a blurred image with the same fruit location and classes as the original input. Because we are use of the kernel to process the image, it is also easy for us to apply different blurry degrees to the image. We can find out that the best degree to help us increase the network performance as more as possible, we will show our results in Figure 3.11.

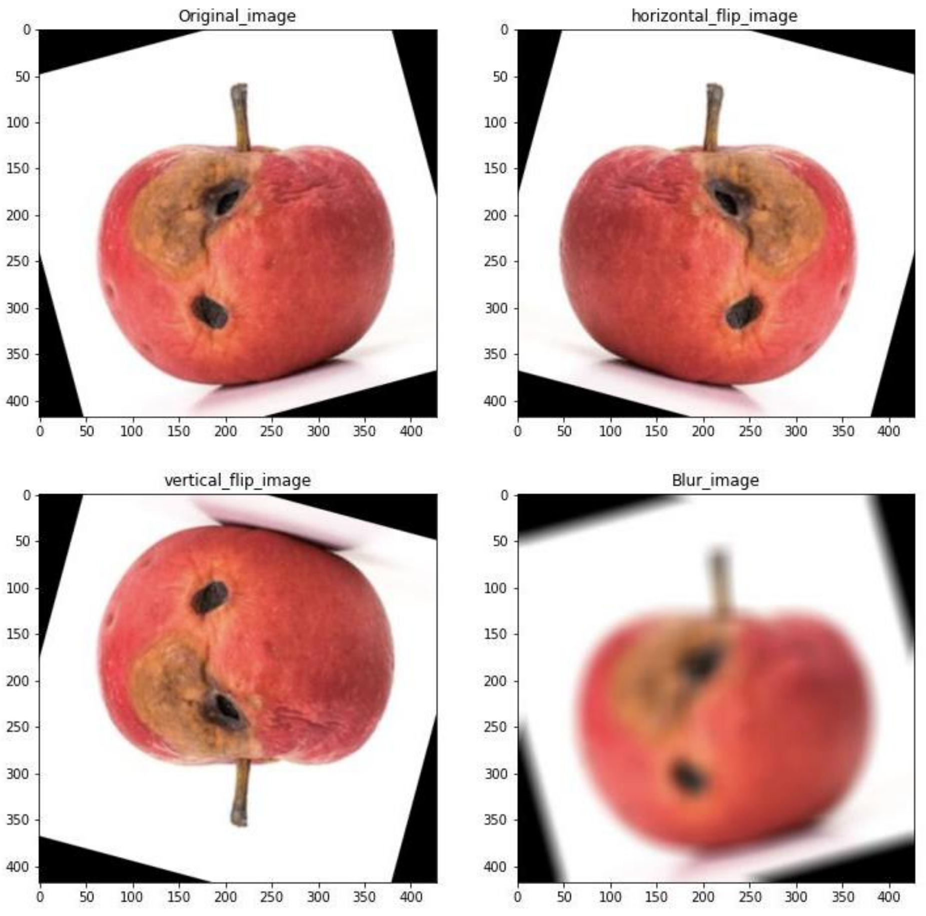


Figure 3.11 An example of augmentation

In Figure 3.11, the original input image is shown on the top left, we add three different methods in data augmentation, namely, horizontal flipping, vertical flipping, and image blurring. These two kinds of flipping methods will add more features to the network. While we take photos for the objects with various view angles, it is okay for us to add rotated images (not only 180 degrees or 90 degrees).

The step will add "ambiguous" features to the architecture, then the network will have the ability to process the images with various shapes and colors, instead of just detecting a specific type of the fruits we have. From our experiments, we add image blurring for data augmentation that will always help our models a lot increase the performance in precision and accuracy.

Chapter 4

Experimental Results

The main content of this chapter is to collect image data and demonstrate the experimental results. At the end of this chapter, we will discuss the limitations of this project.

4.1 Dataset Collection

4.1.1 Dataset Distribution

The dataset is the most important part of the model training. With a good dataset, the model will perform much better even with the same architecture. Regarding a good dataset, it should not include invalid input, the input image should be clear and correctly labeled. After all of this, a good dataset should also be easily utilized. what we mean easily used is we can conveniently preprocess the input image to the format which our network can directly adopt to model training and testing.

For the sake of time limitations, we find a dataset from Kaggle, since datasets on Kaggle always have very superior quality, no invalid data and with annotations and explanations. There have a lot of datasets correlated with fruit freshness on Kaggle, in order to make the network become more powerful and practical, we finally choose to select a dataset with three different kinds of fruits, namely, apple, banana, and orange. Each class of the fruits will have two sub-folders including the fresh or rotten image data for the fruits. Fortunately, they already split the dataset into training and test datasets, the whole data structure is shown in Figure 4.1.

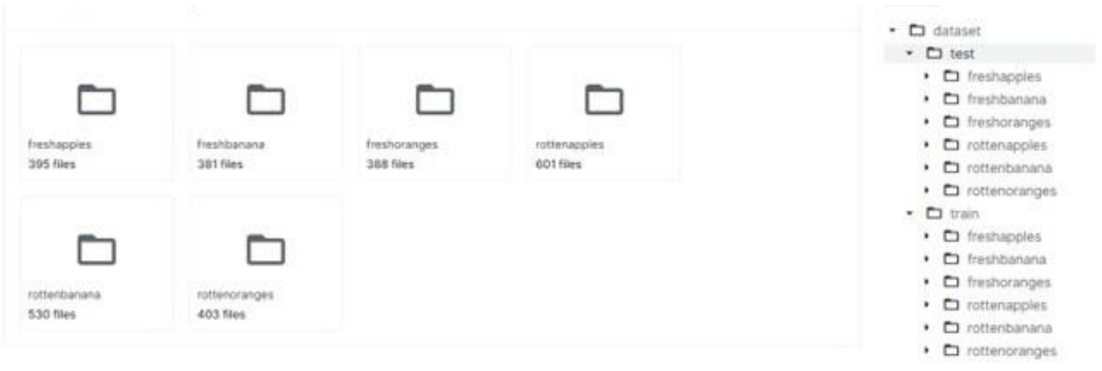


Figure 4.1 Structure of the dataset

Regarding the details of the dataset, in the training folder, there is a total of 10,901 images (all types). In the test folder, there are 2,698 images in total. In the training data folder, we have 1,693 images for fresh apples, 1,581 images for fresh bananas, 1,466 for fresh oranges, 2,342 images for rotten apples, 2,224 images for rotten bananas, and 1,595 images for rotten oranges. For the test data folder, we have 395 images for fresh apples, 381 images for fresh bananas, 388 images for fresh oranges, 601 images for rotten apples, 530 images for rotten bananas, and 403 images for rotten oranges. The distribution of the data samples is shown in Figure 4.2.

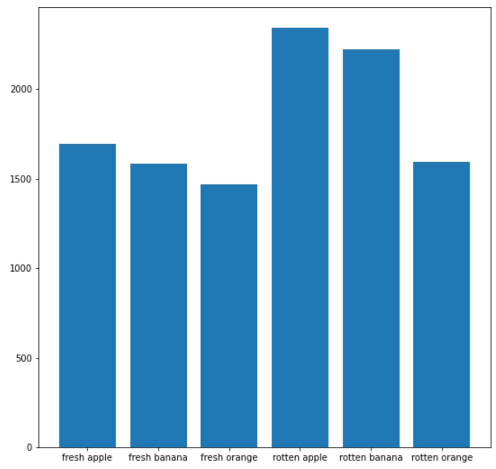


Figure 4.2 The distributions of train dataset

Figure 4.2 shows the distribution of the samples in the training dataset. From Figure 4.2, we know, on average, we have an almost equally number of images for fresh apples, fresh bananas, and fresh oranges. But for rotten fruits, the condition is a little bit different, we have a lot of rotten bananas and rotten apples, and we only have a few rotten orange images.

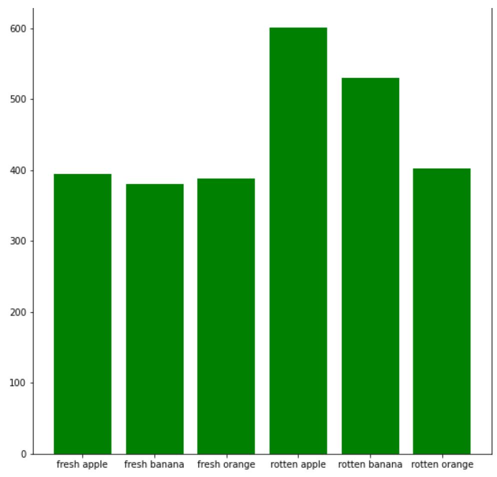


Figure 4.3 The distributions of test dataset

Compared with the dataset distributions for model training, the test dataset almost has the same distributions as the training dataset, only the numbers of sample images in different datasets have a small difference, like the number of images in the training dataset is four times more than the number of images in the test dataset.

From the distributions, we see that the training and test datasets have similar distributions among different classes of images. This is good for our model testing, this dataset is a "nearly balanced" dataset, which indicates the datasets have similar numbers of sample images for different classes of fruits. If we train our network based on a "balanced" dataset, it will always perform better than an "unbalanced" dataset. We can find out this from the pie chart shown in Figure 4.4.

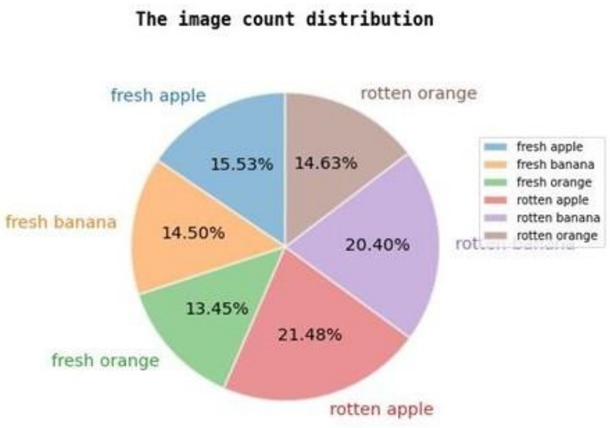


Figure 4.4 Pie graph for train data

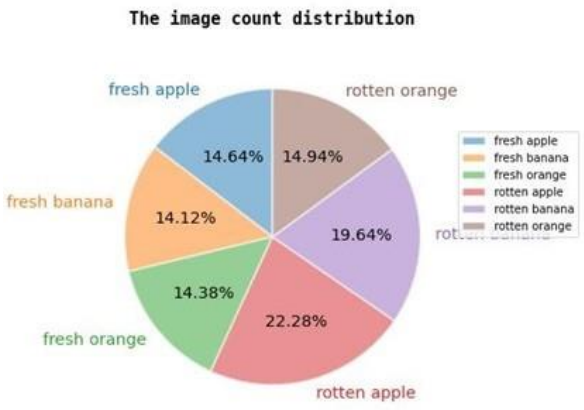


Figure 4.5 Pie graph for test data

It is also very important to determine the size difference between the training set and test set for the reason that if the training set is much larger than the test dataset or the test set is much larger than the training set, then the results could not tell us the real performance objectively. From our estimation, in this project, we choose to keep the original split ratio, which indicates the training data will occupy 80% of the whole dataset, and the test set only occupies 20%. Hence, we choose a great ratio between training samples and test samples is important.

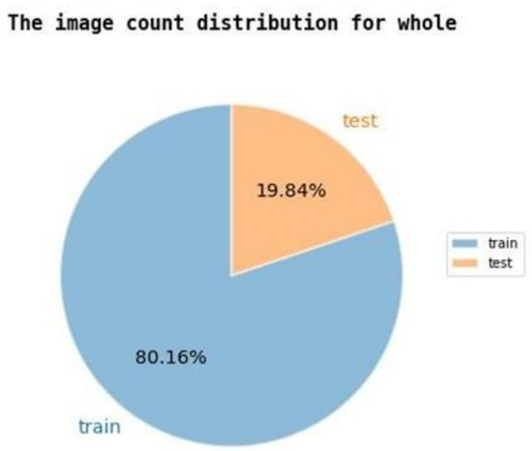


Figure 4.6 Pie graph for whole dataset

4.1.2 Dataset Visualization

In this section, we will show our image samples from the dataset. With these images, we can have a straightforward understanding of the dataset. In our dataset, our samples have been rotated at different angles. This is a normal augmentation for the original images. With this kind of augmentation methods, our network can become more robust. As we see from the image shown in Figure 4.7, almost all the “fresh fruit” images will have a very “beautiful” appearance. What we mean by “beautiful” is the fruits have no black dots or rotten regions in the image. In real reality, we know if there are flaws in fruit skin, it does not mean the fruits are 100% rotten.

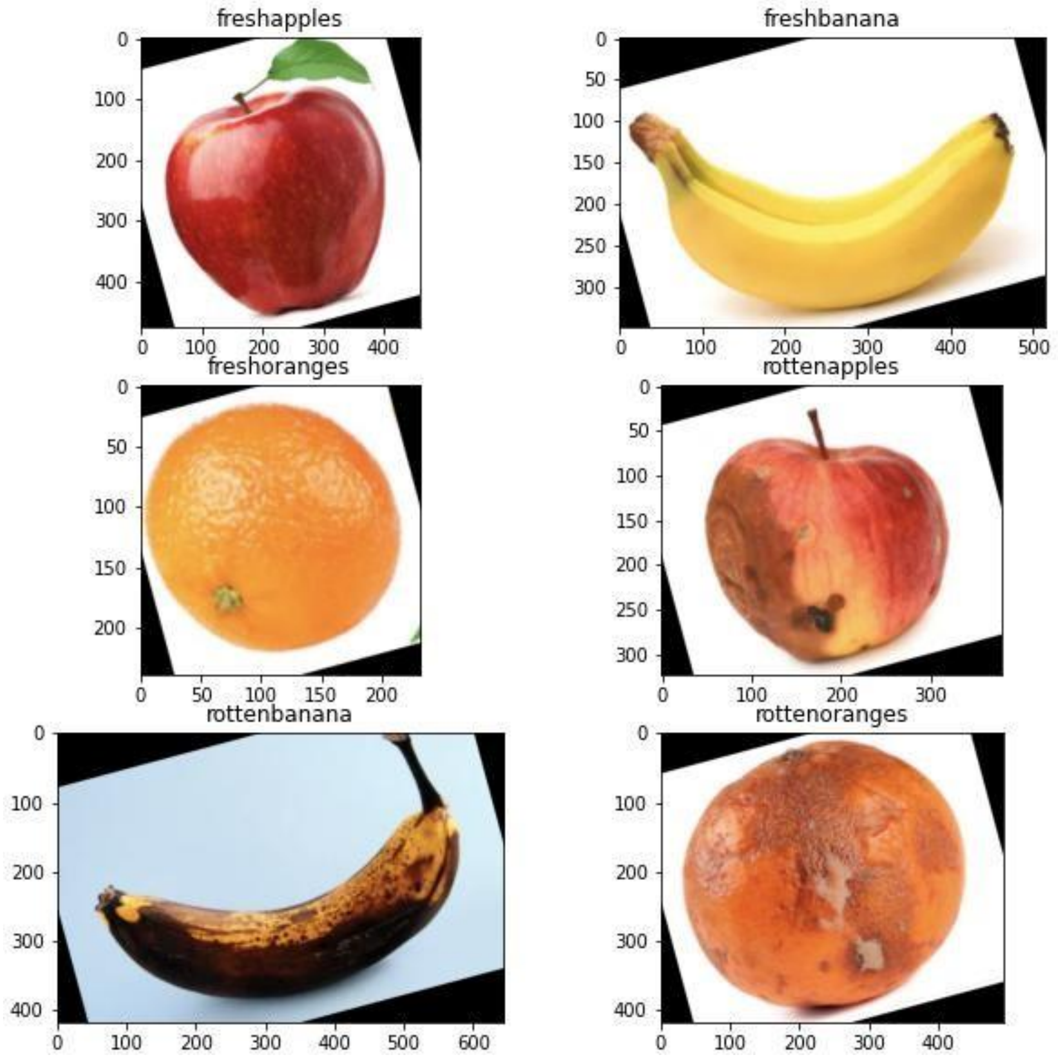


Figure 4.7 Example of dataset

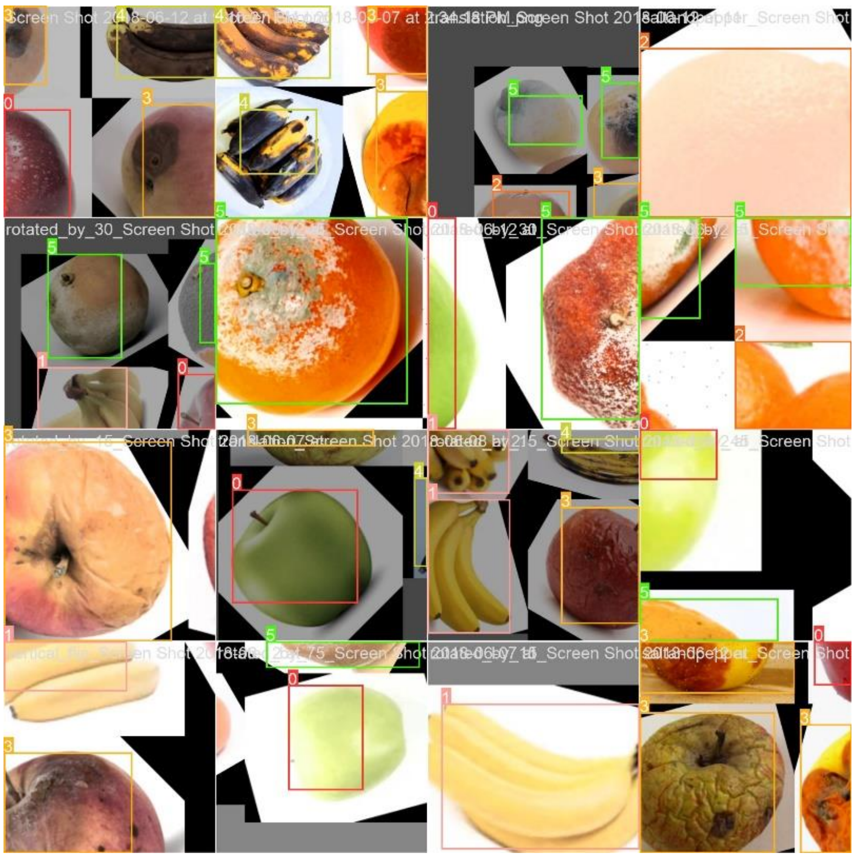


Figure 4.8 Combining different objects in the same image

Figure 4.8 shows the results from the training step, different color indicates prediction labels for the object, for example,

(1) The green box indicates the rotten orange, the red box is the fresh apple (red or green apple),
(2) The yellow box refers to the rotten apple.
(3) The orange box is to indicate the fresh banana.

In Figure 4.8, we also know that for model training, instead of just training the network with only one image, we will randomly combine images and employ these combined images to train the network. Our network will have the ability to detect multiple objects from the image, training the network in this way will also help the network become more robust.

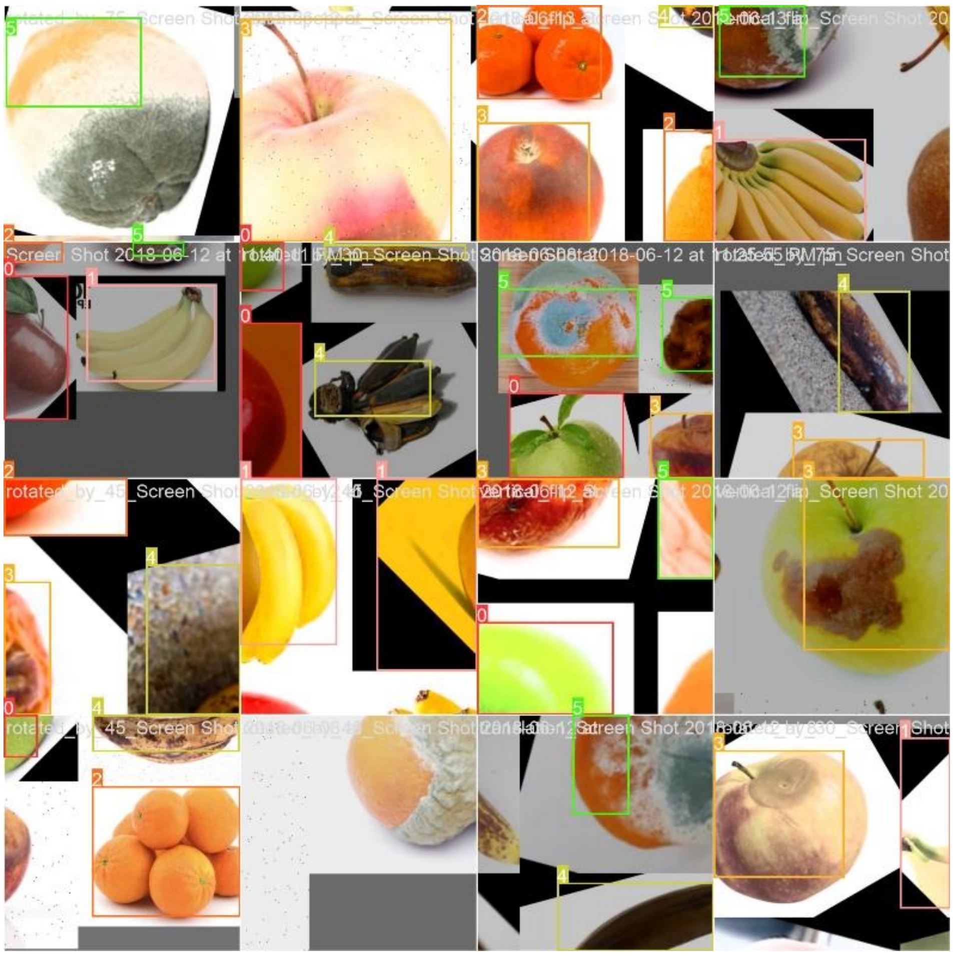


Figure 4.9 Combining different types of fruits in the same image

Figure 4.9 demonstrates the result from the training step, as same as the previous result. In Figure 4.9, different colors mean different prediction labels for the visual object, for example,

(1) The green box indicates the rotten orange,

(2) The red box is the fresh apple (red or green apple),

(3) The yellow box displays the rotten apple.

(4) The orange box indicates the fresh banana.

4.2 Experimental Results

4.2.1 Experiment Details

In order to accomplish the experiments, we are using of our test dataset to test three YOLO models (e.g., YOLOv6, YOLOv7 and YOLOv8), respectively. We will generate the precision and recall curves, F_1 curves and confusion matrices to showcase the results. Regarding precision and recall, we calculate them with:

$$precision = \frac{TP}{TP + FP} \tag{4.1}$$

$$recall = \frac{TP}{TP + FN} \tag{4.2}$$

where *TP* indicates True Positive (correct classification), *FP* indicates False Positive (background classified as a fruit), and *FN* indicates False Negative (fruit classified as a background).

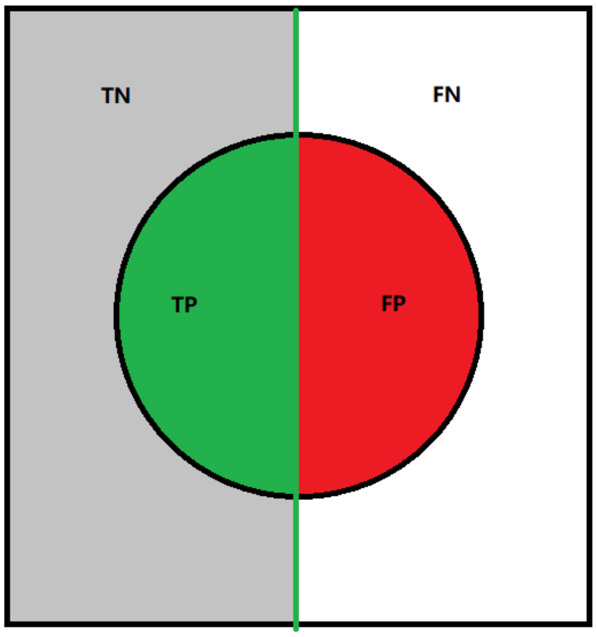


Figure 4.10 The explanation of FP, FN, TP, and TN

Pertaining to the experiments, we test YOLOv6, YOLOv7 and YOLOv8 models with the same dataset. For the performance estimation, instead of just calculating the precision and recall, we also count the F_1 score. The confidence curve and confusion matrix are to better estimate the performance. The equation to calculate F_1 is shown in equation (4.3),

$$F1 = \sum \frac{2 * precision}{precision + recall} \tag{4.3}$$

The summation sign in equation (4.3) indicates we will sum up all the products from each point related to the PR curve. The confusion matrix will show the error rate with a heat map, the color in the heat map will give us information about the value for each block.

4.2.2 YOLO Architecture Comparison

We test our test image on three different architectures, they are YOLOv6, YOLOv7 and YOLOv8, in order to get a better understanding of the performance between different architecture, we will use precision, recall and F1 score to estimate the network performance, first, we will compare the precision and recall result between different architecture.

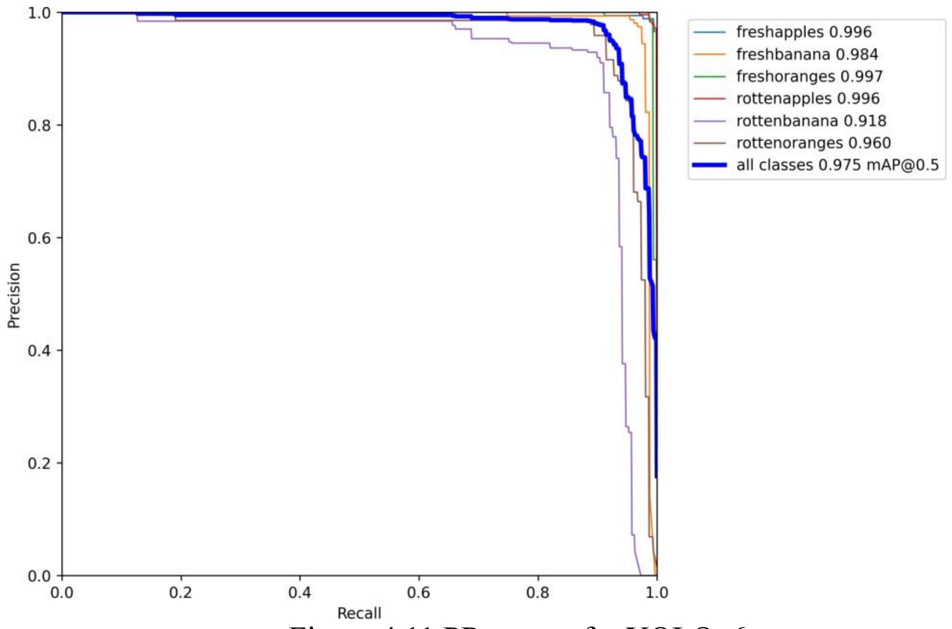


Figure 4.11 PR curves for YOLOv6

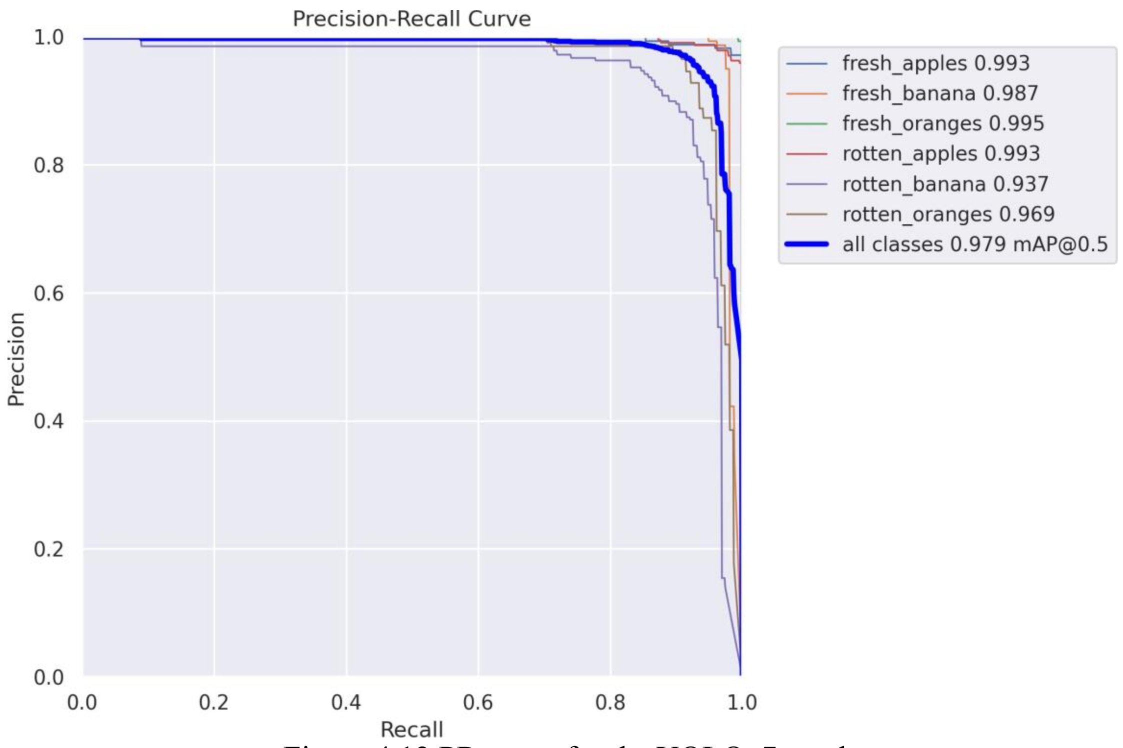


Figure 4.12 PR curve for the YOLOv7 result

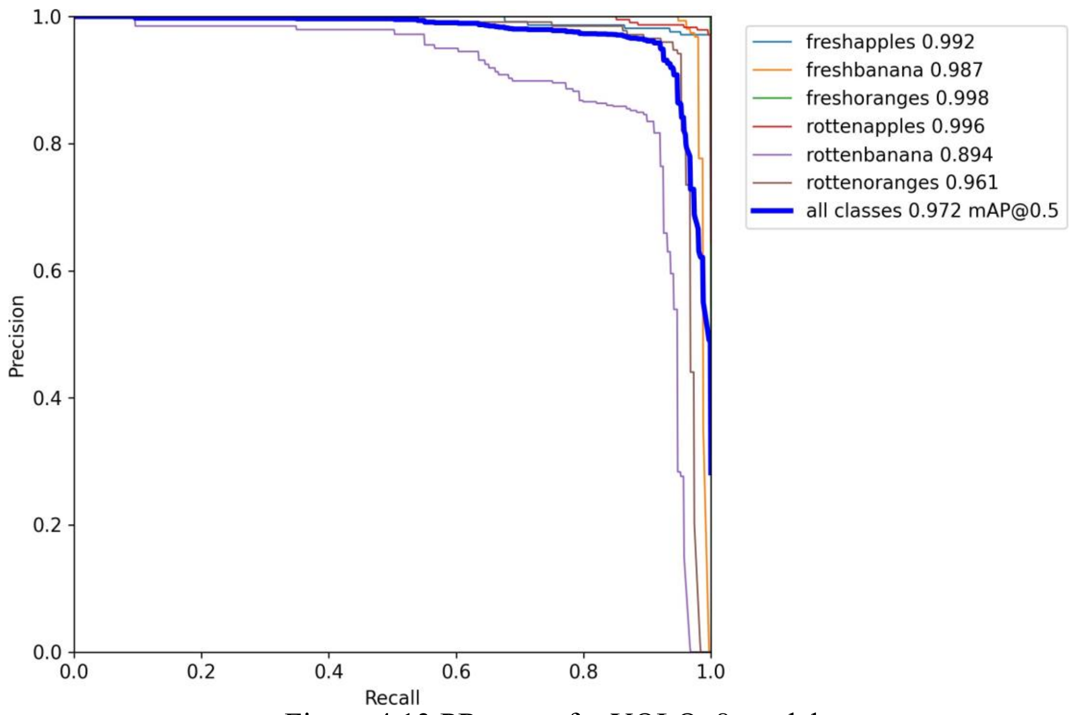


Figure 4.13 PR curve for YOLOv8 model

In Figure 4.11, the blue line shows the PR curve for fresh apples, the orange line shows the results for fresh bananas, and the green line shows the results for fresh oranges. Other three lines show the results for rotten fruits. The red line indicates apples, the purple line shows oranges, and the brown line reflects the results for bananas. The bold blue curve shows the result for the average performance of all classes. In YOLOv6 model, the worst prediction is rotten bananas. Predicting fresh fruits always has a satisfactory performance.

From the legend shown in the PR curve in Figure 4.11, the number following the classes of objects is the AUC (area under the curve). If a network has a good performance, we always want the network to have high precision and recall, which indicates a higher AUC indicates better performance. From the PR cure, we can know, normally, it will have an average precision higher than 0.9 and recall higher than 0.9 at the same time among all the different types of fruits. Detecting fresh apples and fresh bananas usually has the highest precision and recall. The blue bold line is the average curve among all the classes of fruits. As we can see, the average performance is very high, with an AUC of 0.975. In summary, YOLOv6 model has a better performance for fruit freshness classification, especially for apples.

Pertaining to the results of YOLOv7 model, the legend of lines in Figure 4.12 is as same as what we show in the YOLOv6 model. From the figure, based on the rotten bananas, it still is the worst result. But for YOLOv7 it is much worse than the results from YOLOv6. The bold blue curve still shows the results for the average performance for all classes.

From the PR curve, it is easy for us to know that the AUC curve of all classes for the YOLOv7 model is less than 0.972. Compared to the same type of AUC curve of YOLOv6 model, we know the performance of YOLOv7 and YOLOv6 is similar. From the curve for YOLOv7, it is obvious that the PR curve is the curve for YOLOv6, and the AUC for YOLOv6 is also large than the AUC for YOLOv7 mode. From both YOLOv6 results and YOLOv7 results, we know that detecting the rotten orange is the hardest project compared with detecting other types of fruits, this may be caused by the shapes and the colors of the fruits, because there are three types of fruits in our dataset.

From Figure 4.13 (PR curve for YOLOv8), the meaning for all lines is kept same as before. Compared with YOLOv6 andv7, YOLOv8 has much better performance, even for the worst case (rotten banana), YOLOv8 still has a high F1 score (0.937)

From the PR curve of YOLOv8, we know the AUC for YOLOv8 is higher than both YOLOv6 and YOLO v7, this indicates YOLOv8 may give a better classification result. From the average PR curve (the blue bold line), the performance of YOLOv8 really looks like the performance of YOLOv6, but, if we compare other types of fruit, for example, rotten banana or rotten orange, obviously, the YOLOv8 has a better detection result compared to YOLOv6, as for YOLOv6, the AUC for rotten orange is 0.96, and for rotten banana is 0.918, but for YOLOv8, the AUC for rotten banana is 0.937, rotten orange is 0.969; the performance for YOLOv8 increases a lot, this is great, which indicates for fruit detection, using YOLOv8 instead of using YOLOv6 should be a good idea, especially when the test image includes orange or banana; but if the user only wants to detect fresh apple or rotten apple, YOLOv6 or YOLOv8 will have a similar performance, if they want to put the network in a mobile machine, they are better to apply an architecture with fewer parameters.

After comparing the results from different architectures with precision and recall, we also compare the F1 score from different architectures.

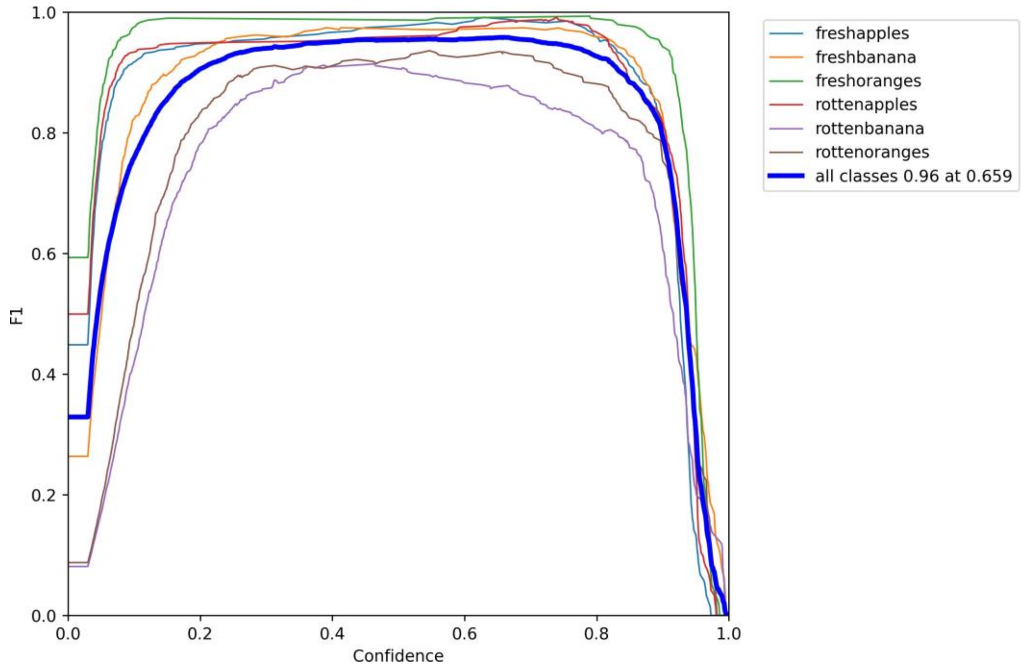


Figure 4.14 F1 versus confidence curves for YOLOv6

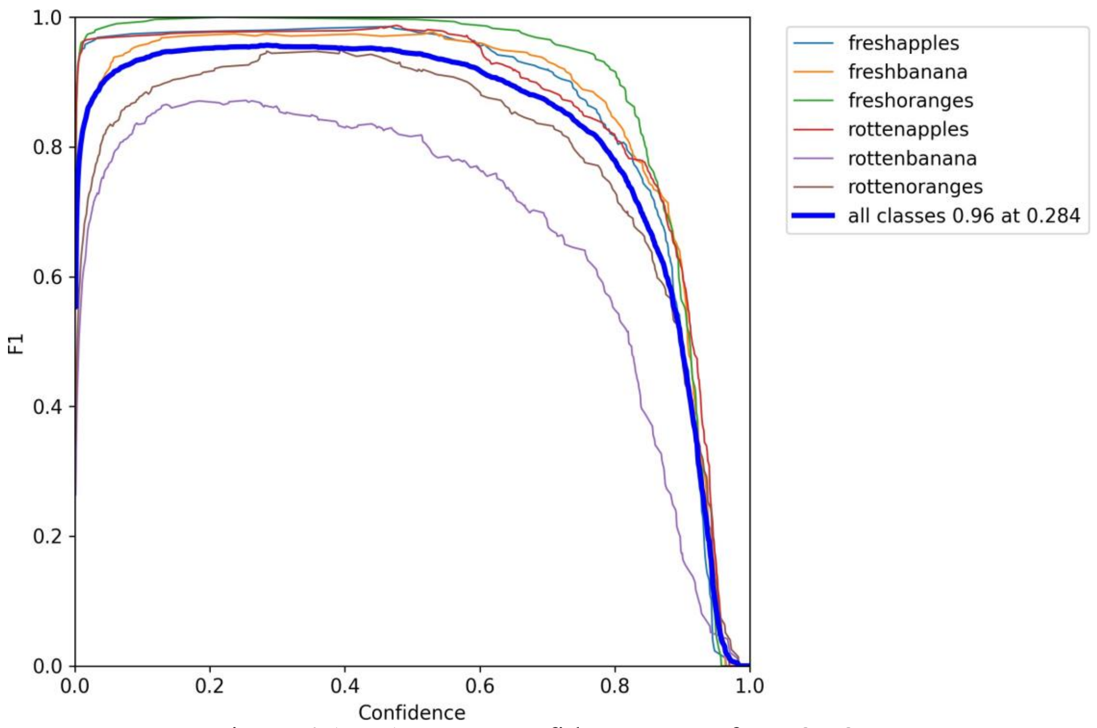


Figure 4.15 F1 versus confidence curve for YOLOv7

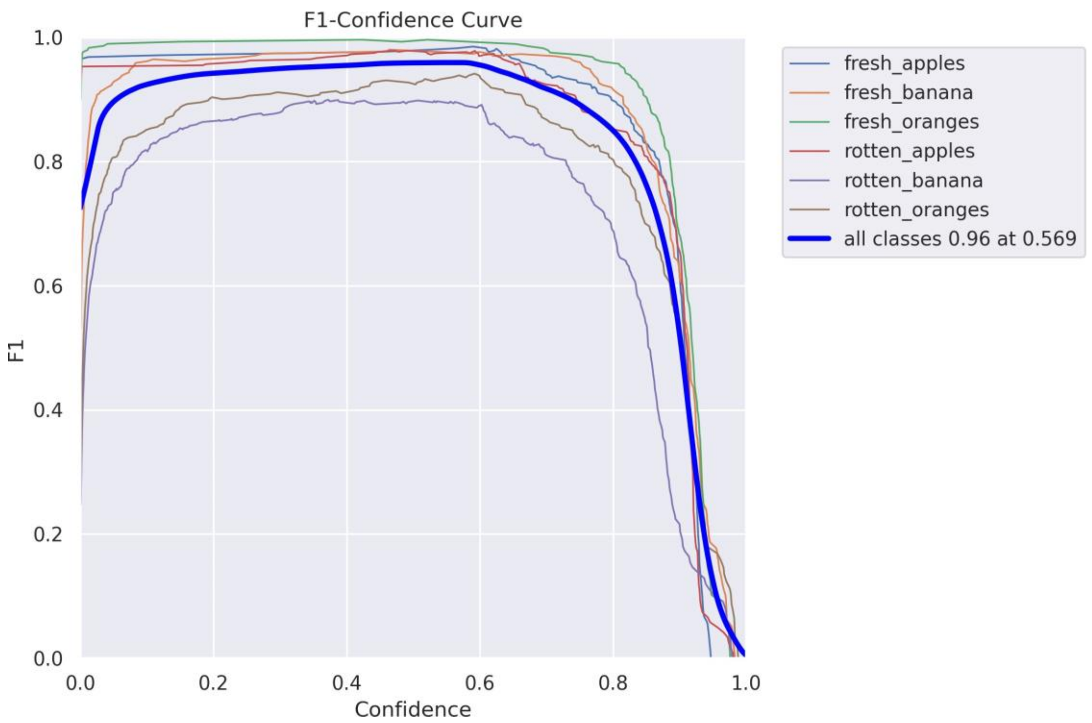


Figure 4.16 F1 versus confidence curve for YOLOv8

The orange line displays the result for fresh bananas, the green line shows the result for fresh oranges, the blue line shows the PR curve for fresh apples, and the legend for other lines is as same as the previous PR curve. The bold blue curve still shows the result for the average performance among all classes same as in Figure 4.11. As we find in the PR curve, rotten banana still has the worst result, we should label the rotten bananas as such a case.

In order to solve this problem, we can add more image samples for rotten bananas or train an individual network to detect rotten bananas in the future. We split the worst case and increased the performance a lot.

Figure 4.14 shows the curve for F_1 score versus the confidence of YOLOv6. F_1 score is a number that has a similar meaning as AUC, and confidence is a value from zero to one to show the confidence of the network prediction. Confidence always is employed as a threshold to make an accurate object detection, a low confidence always indicates the network is not very confident to make the detection, and a high confidence indicates the inversed thing (the network has a low confidence about the detection result), so it is better for us to choose a higher confident value as the threshold instead of a low value (best threshold should be 0.99).

What we want finally is a high F_1 score with a relatively high confidence. We determine the most important parameters to estimate the performance is the F1 score, then the confidence value. Regarding the F_1 score, we always want the network to have a high F1 score (the best should be 1) when we choose a confident score as a threshold. In YOLOv6 results, the highest F_1 score for the YOLOv6 model should be around 0.95 with a threshold of around 0.659. This is a relatively high confidence value and F_1 score. We believe with such a confidence threshold and more than 0.9 accuracy; we can directly apply it in the real scene.

For the rest of types of the fruits (for example, all fresh fruits etc.), they almost have similar performance, so it is not necessary to train them separately. Training a network with a variety of inputs and a lot of image samples will be better than the “trained single” network.

In Figure 4.15, from the confidence curve, the legend is as same as before. Compared with the YOLOv6 model, the performance of rotten oranges has been improved a little bit. For all the types of the fruits, the performance of YOLOv7 is a little bit worse than the performance of YOLOv6. The reason is that with the new architecture added to the YOLOv7, it is not good to process the fruit images. This always happened in deep learning, a much deeper network is not possible to get a better result.

From F_1 score curve and confidence curve, it is obvious that the highest F_1 for YOLOv7 model is similar as YOLOv6 model, but the confidence threshold for YOLOv7 is around 0.3, which is much smaller than the threshold of YOLOv6 model. A lower confidence threshold may involve more false alarms, which will have a bad effect on the object detection results. So, compared with YOLOv6, YOLOv7 seems not a good architecture for the fruit detection task, this is caused by the new components add to YOLOv7, for the reason that sometimes a simple architecture may be much easier to train and much easier to converge.

Figure 4.15 is the result for the YOLOv8 model. Compared with the previous architecture, YOLOv8 has a similar performance shown on the F1 score compared with YOLOv6, so we think YOLOv6 and v8 are all good to make a classification for fruit.

From the F1-confidence curve, the highest F_1 score is around 0.95 when we utilize of a confidence threshold of around 0.6. This is great that it indicates YOLOv8 has a high F_1 score with a relatively high threshold. This indicates it will have fewer false alarms. From the YOLOv6 result, when it gets the high F_1 score, the confidence value for YOLOv6 is 0.659, but for YOLOv8, the confidence is 0.569, the difference between these two confident scores is a little bit large, which indicates YOLOv6 has a higher confidence to make a correct detection, it looks like YOLOv6 is better than YOLOv8. YOLOv8 has a better performance, a higher confidence score may only mean YOLOv6 can detect more different types of fruit, in view of with more types of fruits, there’s no doubt that in order to increase the correct detection, we should decrease the confidence score to let the network accept more detections compare with less types of fruit detection.

Then we can also compare the results for different confusion matrices, confusion matrix is a great method to compare the prediction result between different architectures.

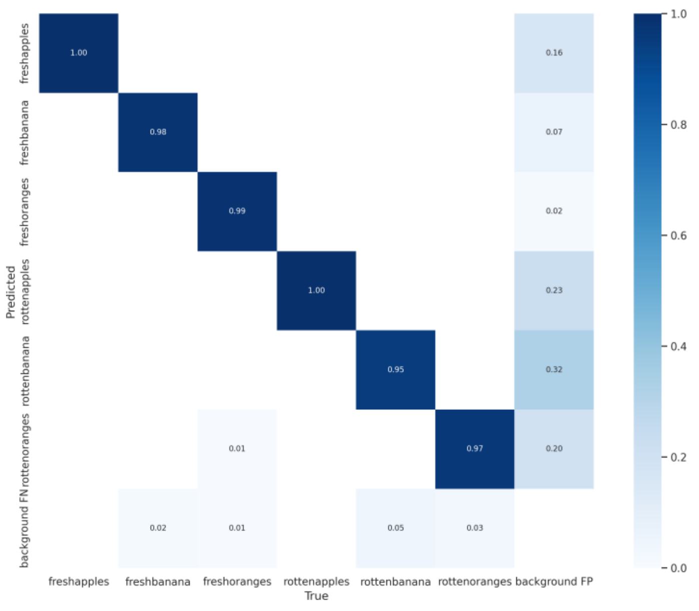


Figure 4.17 The confusion matrix for YOLOv6

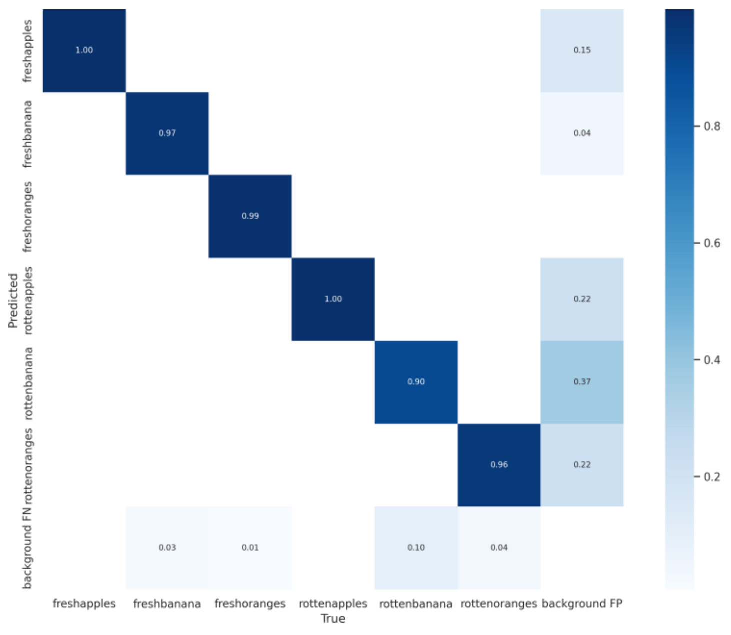


Figure 4.18 The confusion matrix for YOLOv7

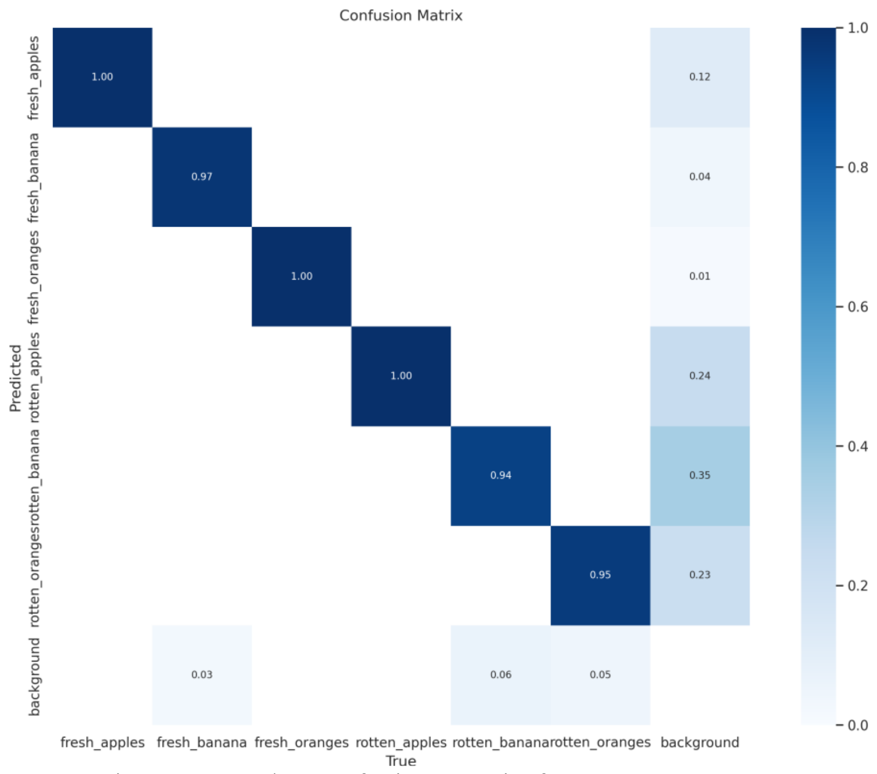


Figure 4.19 The confusion matrix for YOLOv8

The heat map shows a confusion matrix for YOLOv6, different colors mean different correlation scores. In Fig. 4.17, the white area indicates zero and the deep blue area refers to the one hundred percent. The rows and columns show different classes in our dataset. The rows mean the prediction results from our model, and the column refers to indicates the ground truth for our classification.

A confusion matrix always gives us general information about the performance of the detector. From the confusion matrix of YOLOv6 models, it is obvious that the network performs very well for apples, and a little bit worse on bananas. Classifying oranges will be the hardest task. From the last column, it seems like the network will also have error predictions for each type of fruit. For example, though the deep learning network successfully detects all rotten bananas, it also classified the background image (not the object image) as a rotten banana.

Figure 4.18 shows the confusion matrix for YOLOv7, from the confusion matrix of YOLOv7, we compared it with the confusion model of YOLOv6, YOLOv7 model has a better performance on orange classification. If we look at the details of the confusion matrix of YOLOv7 models, we find that for YOLOv7, the wrong classification is always related to the background, which indicates our error is always a “missing detection”. This is great as the real scenario, we will always apply the architecture to conduct a fruit classification for fruits, it indicates there must be a fruit, and we can notice that and detect it manually. For rotten oranges, it is still very hard for the network to make a classification, a lot of the backgrounds are classified as rotten oranges.

Figure 4.19 shows the confusion matrix, from the confusion matrix of YOLOv8, compared with YOLOv7, the classification accuracy of YOLOv8 increases a lot. With the YOLOv8 model, we can correctly classify apples with an error rate that is smaller than 1% to classify orange and banana. Compared with previous YOLOv7 or YOLOv8 results, regarding rotten bananas or rotten oranges, the network still has a high probability of detecting the background as a rotten orange, or rotten banana, this is a very interesting phenomenon.

We also show some of our result for precision and recall with confidence separately.

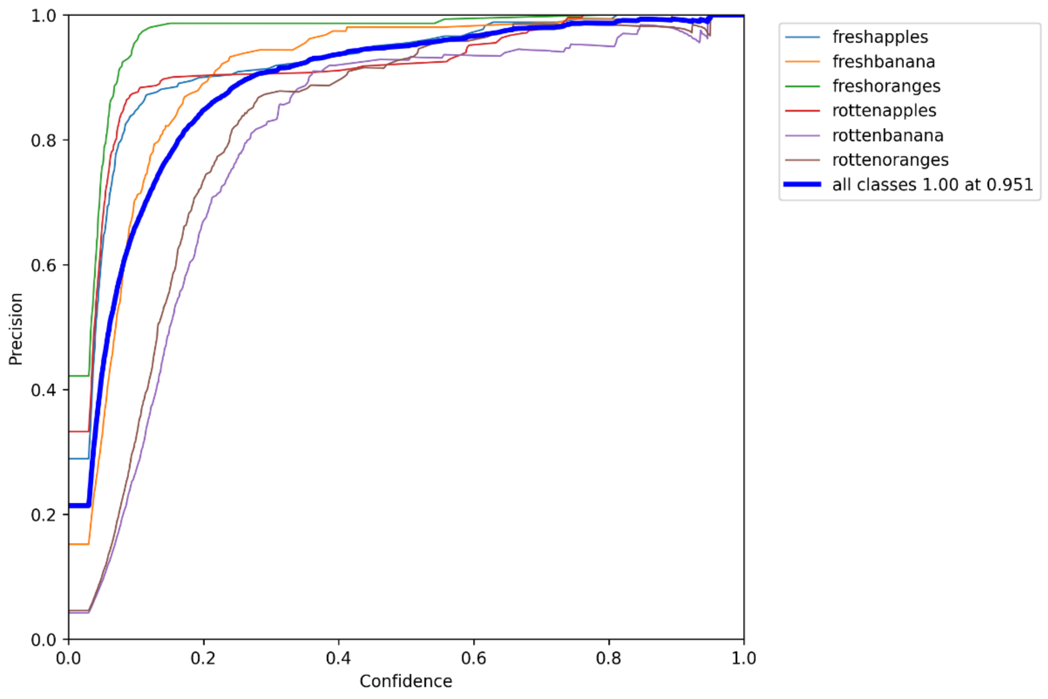


Figure 4.20 The precision vs confidence curve for YOLOv6

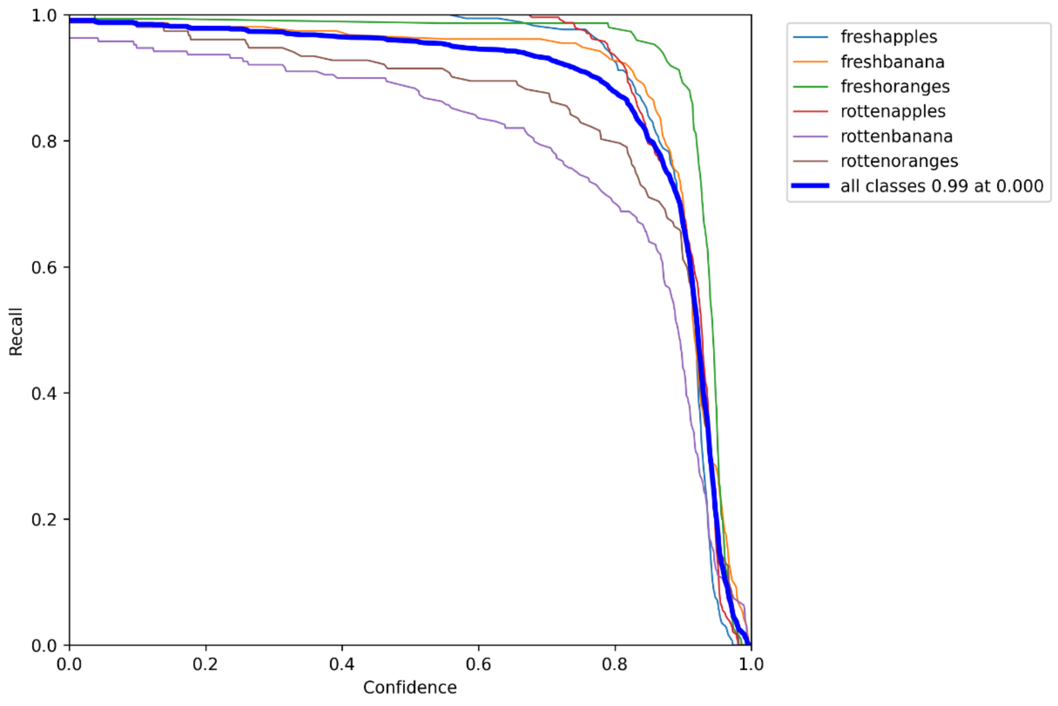


Figure 4.21 The recall vs confidence curve for the YOLOv6

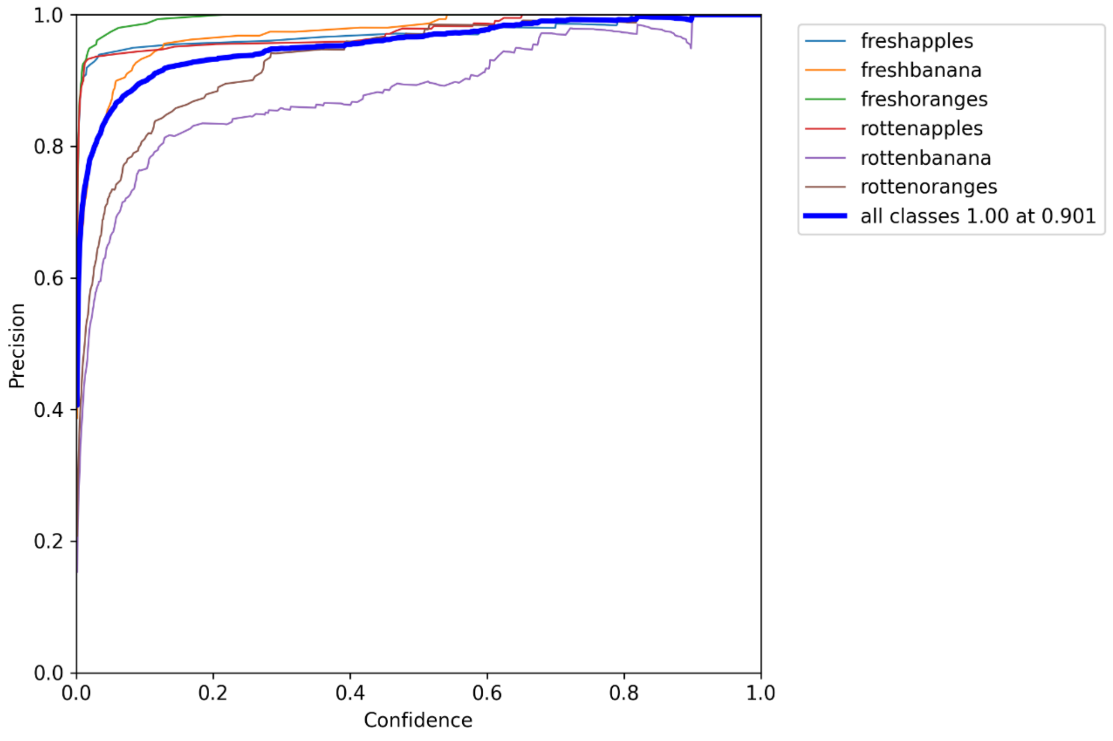


Figure 4.22 The precision vs confidence curve for YOLOv7

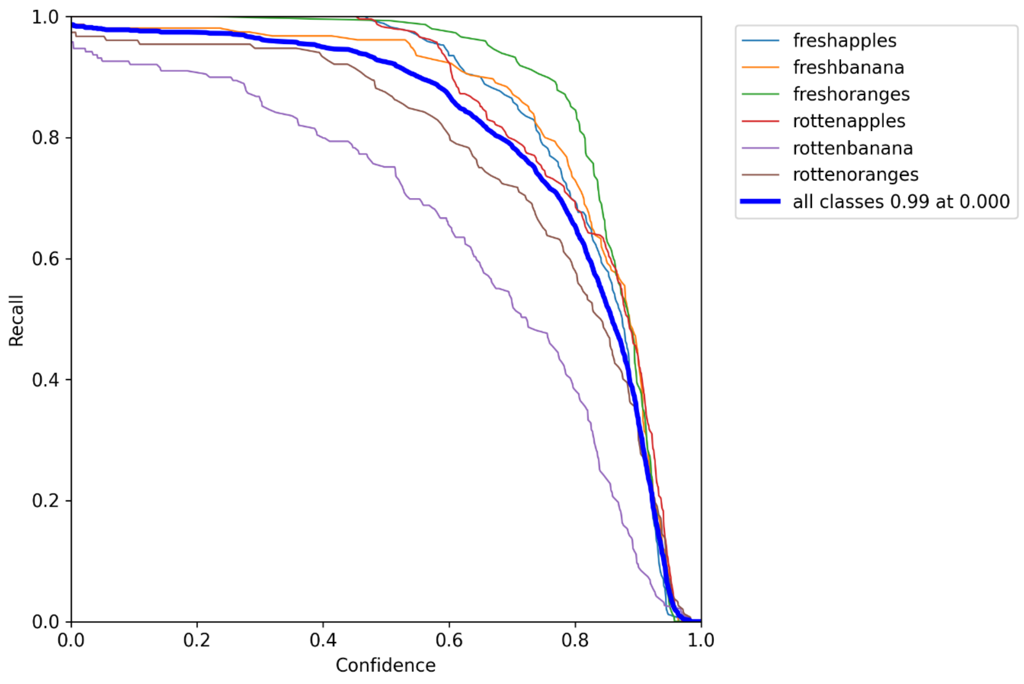


Figure 4.23 The recall vs confidence curve for YOLOv7

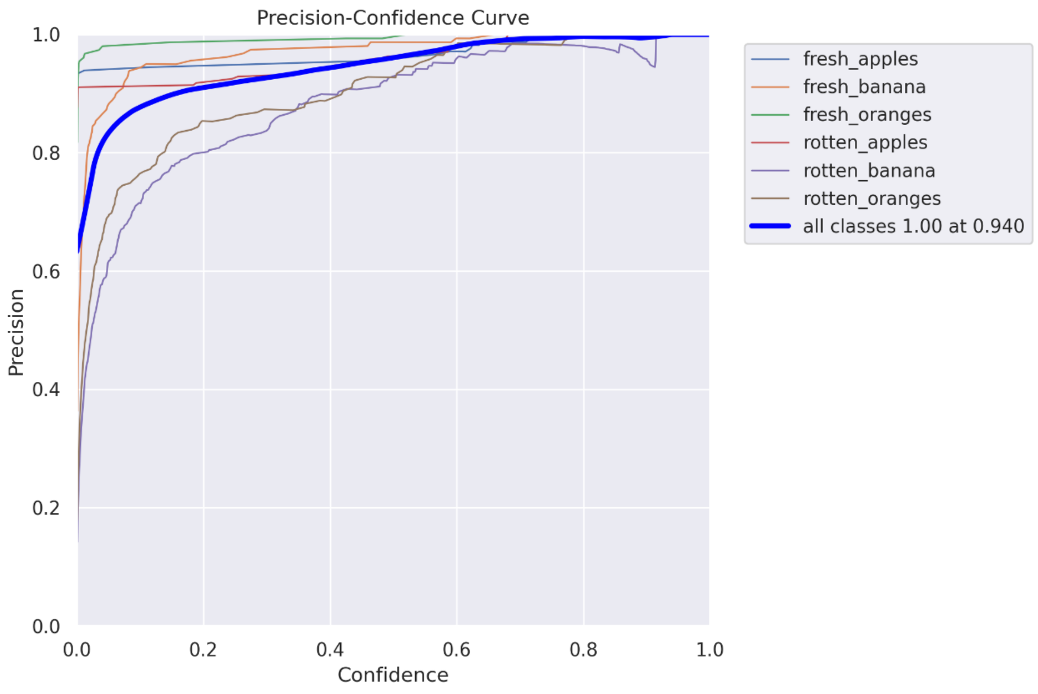


Figure 4.24 The precision vs confidence curve for YOLOv8

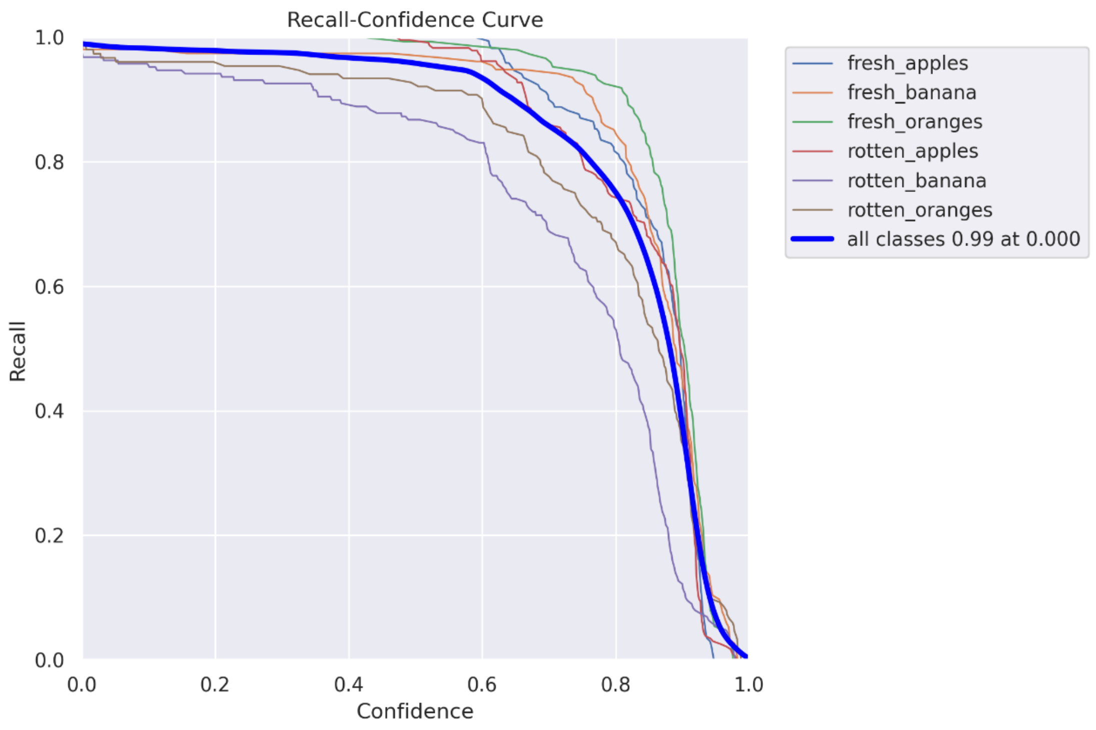


Figure 4.25 The recall vs confidence curve for YOLOv8

4.2.3 Majority Vote

A majority vote is a traditional method that has been adopted to improve the detection performance. Since the YOLOv8 model is very easy to be implemented, we plan to use it and get a better result. There are two main reasons why we take advantage of the majority vote as a method in this thesis.

The first reason is that the majority vote is a method that can be used “outside” of the architecture, what we mean by “outside” is the method that will only process the outputs from the architecture instead of manipulating the whole architecture. Compared with the method which will change the weights or add new layers to the original architecture, the majority vote is more much, because we always keep the architecture as same as before, so it will be easy for us to trace back. The best model weights are utilized to get the final results.

The second reason is with a majority vote, we can easily determine how many results we have, and which results we harness to generate the final outputs. This is a very useful idea to get a good result. Since with different weights, the network will have varying performance. With a majority vote, we can always choose the best performance as the input for the majority vote, there’s no doubt that we will always have the best performance. The majority vote method also has the best adaptability skill. Changing the backbone (like YOLOv8 we mentioned before) will not influence the architecture, we still can use the same code to change the input to the result we get from the new backbone, this will also help us easily change our architecture in the future.

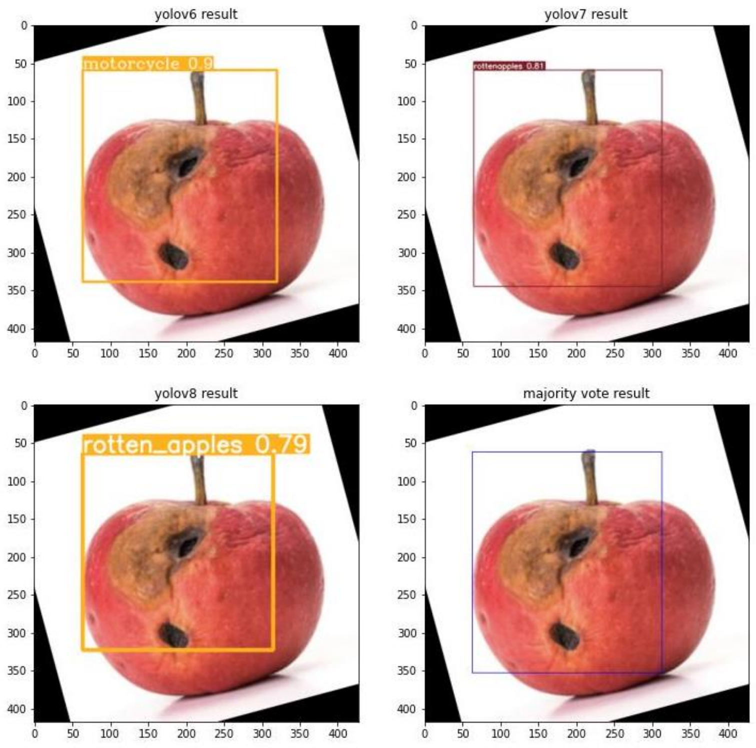


Figure 4.26 The major result of prediction

In Figure 4.26, the top row is the result of YOLOv6 and YOLOv7, and the bottom left is the result of YOLOv8. Based on the bottom right, we show the majority vote results different from all the previous results we have. As we see from the example, the result looks better than the previous results. We compare the previous result with the combination result as shown in Table 4.1.

For the majority vote method, by reason of we cannot calculate the combination probability, we cannot draw the precision vs recall curve when we vary the confidence for each network prediction result, so we just calculate the precision and recall for different methods. The first three rows are the best precision and recall for different types of fruits, the last row is the majority vote result. From Table 4.1, it is obvious with the majority vote method, our network performance increases by at least 1% regarding to both precision and recall. We also compare our final result with other network architectures, and our result is the best.

Table 4.1: Majority vote compared with other results

Models	Precision	Recall
YOLOv6	0.956	0.943
YOLOv7	0.933	0.92
YOLOv8	0.957	0.941
Majority vote	0.961	0.957

Table 4.2: our final result with other network architectures

Architecture	Accuracy	Loss
Convolution network	0.9417	0.1491
For the baseline on Kaggles	0.9492	0.1329
For our best result	0.96	0.113

4.2.4 Ablation Study

While we train the deep learning models, we are use of the original fruit images, because the network training with an augmented dataset will increase the performance of our proposed methods. In this section, we will show our ablation results if we train our proposed models with a variety of datasets. There are three types of datasets to compare, the first one is the original dataset, which includes the image without any manipulation of the image. The second dataset is that we add horizontal flipping and vertical flipping to the training dataset. The third dataset is to add flipping and blurring to the training dataset, the result is shown in Table 4.3.

From the results in Table 4.3, it is obvious that when we add flipping methods to the data augmentation, the performance increases a little bit. Compared with only flipping augmentation dataset, we add the blurred images to the dataset, that increases the performance a lot. Both precision and recall from different architectures increase at least one percent, this indicates compared with the flipped images, blurred images are much effective for model training, we should continue training the deep neural network with blurred images in the future.

Table 4.3: Ablation study results

	Model training based on the original dataset		Model training based on a dataset with flipping augmentation		Model training based on a dataset with flipping and blurring augmentation	
Architectures	Precision	Recall	Precision	Recall	Precision	Recall
YOLOv6	0.956	0.943	0.955	0.94	0.967	0.96
YOLOv7	0.933	0.92	0.937	0.921	0.952	0.931
YOLOv8	0.957	0.941	0.96	0.934	0.97	0.964

From the results we show in Section 4.2.5, it is obvious that with the majority vote method, our net can perform better than the individual net result, this is great for fruit freshness detection. As combining the results from different networks is very efficient and robust, we can always keep the high precision result and deny the worse results, the average performance will always be better than the previous single network result. In this section, we add our ablation study for this majority vote method, in general, the majority vote method is,

$$voting(w_1 * net_1, w_2 * net_2 \dots\dots w_n * net_n) \tag{4.4}$$

where $w_1$to w_n shown in equation (4.1) are the weights for different architecture, the net_1 to net_n are the networks from different network architectures. The weights will always be in range [0,1], 0 indicates we will not utilize the detection result from such a network, 1.00 indicates we will directly believe the detection results. If a network has a high weight (high confidence), then we will believe that the result is more accurate than other results. With different weights, we can make the majority vote results become much accurate and robust. In Table 4.4, we show different weights after we combine YOLOv6, YOLOv7 and YOLOv8 model together.

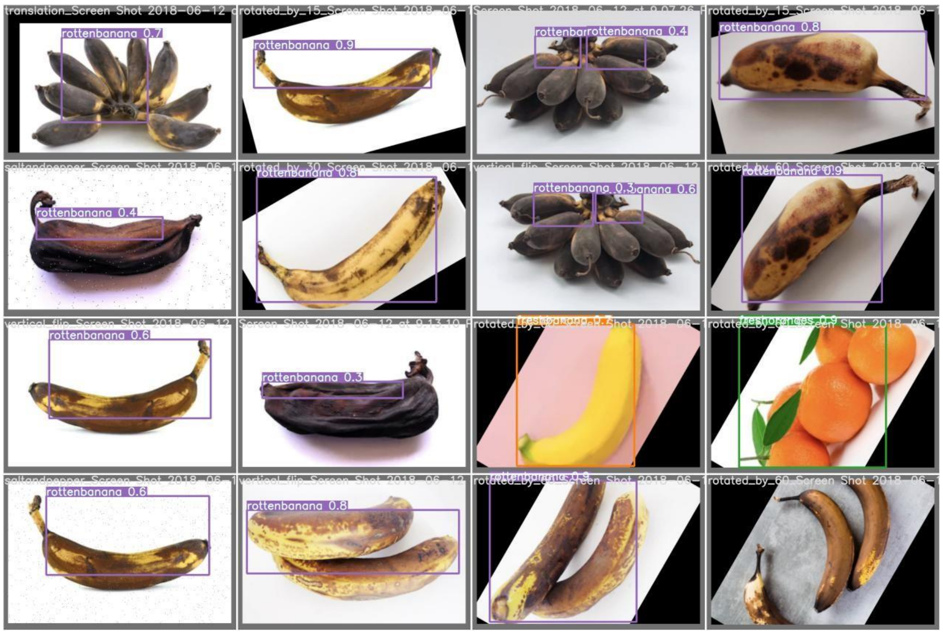
From Table 4.4, we easily make a conclusion that balancing the weights may not be the best way to generate satisfactory results. Given the best performance architecture, a relatively large weights will have a better performance, but increasing the weights too much will make the majority vote results get close to the single network result.

Table 4.4: The ablation study after we change the weights

Series	Weight for YOLOv6	Weight for YOLOv7	Weight for YOLOv8	Majority vote result	
				precision	recall
1	0.33	0.33	0.33	0.961	0.957
2	0.6	0.2	0.2	0.96	0.942
3	0.2	0.6	0.2	0.94	0.92
4	0.2	0.2	0.6	0.97	0.963
5	0.8	0.1	0.1	0.954	0.94
6	0.1	0.1	0.8	0.955	0.942

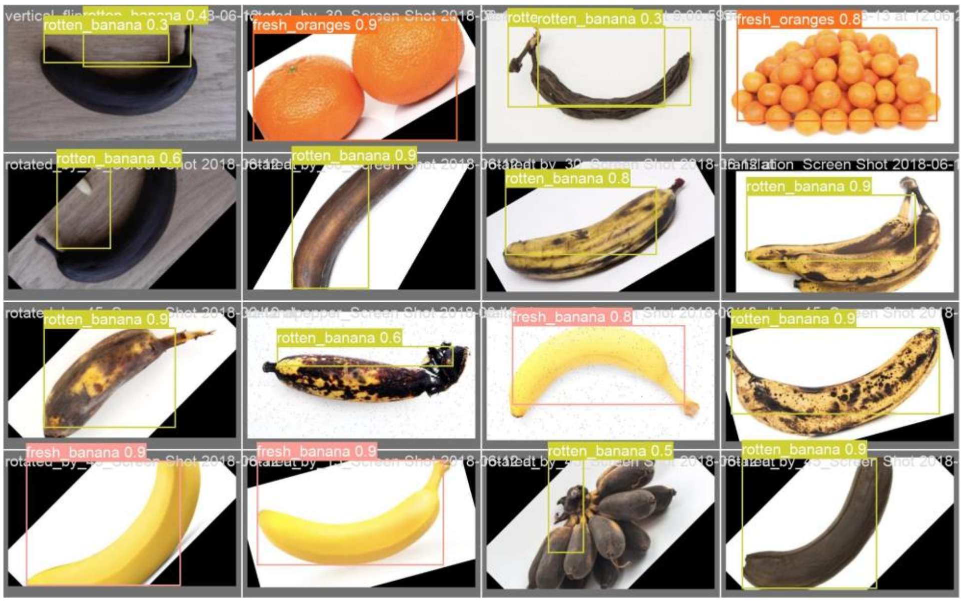
4.2.5 Prediction Results

In this section, we show the prediction results generated by using deep neural network. In our dataset, various color bounding boxes mean different classifications, purple bounding box indicates rotten banana, green bounding boxes refer to fresh oranges, and orange bounding box indicates fresh bananas. From the test images, the images also have a rotation around the center, but this kind of argumentation of the dataset can give us a better estimation of the performance of the network in the real scenario, since in our real world, it is impossible for us or for a machine to take a photo for a fruit from a constant view angle.



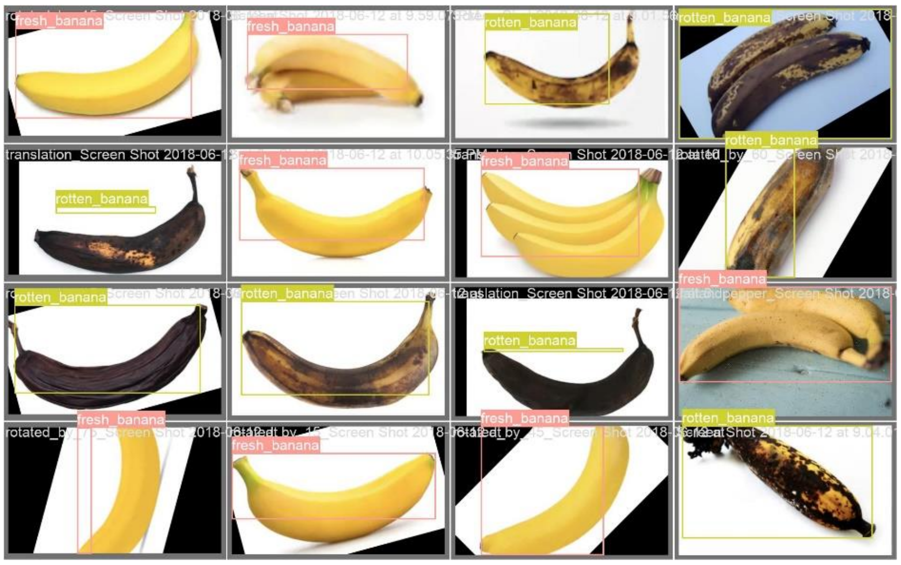
rottenbanana 0.7
rottenbanana 0.9
rottenbanana 0.4
rottenbanana 0.8
rottenbanana 0.4
rottenbanana 0.8
rottenbanana 0.3
rottenbanana 0.6
rottenbanana 0.9
rottenbanana 0.6
rottenbanana 0.3
rottenbanana 0.6
rottenbanana 0.8

(a)



rotten_banana 0.3
rotten_banana 0.4
fresh_oranges 0.9
rotten_banana 0.3
fresh_oranges 0.8
rotten_banana 0.6
rotten_banana 0.9
rotten_banana 0.8
rotten_banana 0.9
rotten_banana 0.9
rotten_banana 0.6
fresh_banana 0.8
rotten_banana 0.9
fresh_banana 0.9
fresh_banana 0.9
rotten_banana 0.5
rotten_banana 0.9

(b)


(c)

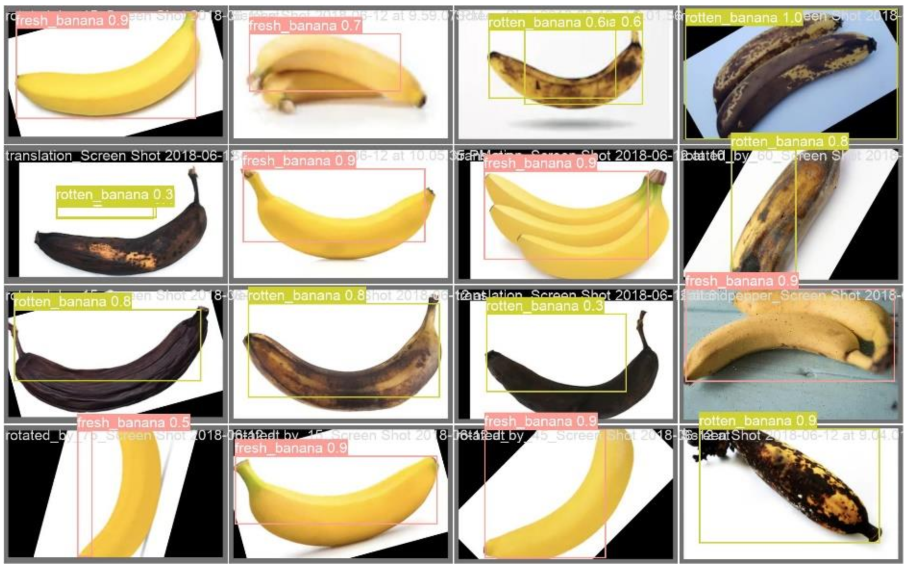

(d)

Figure 4.27 Results of YOLOv8 models

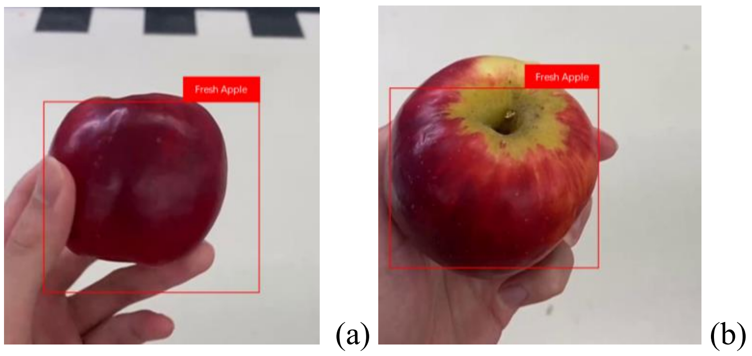


Figure 4.28 Prediction on the real video fresh apples

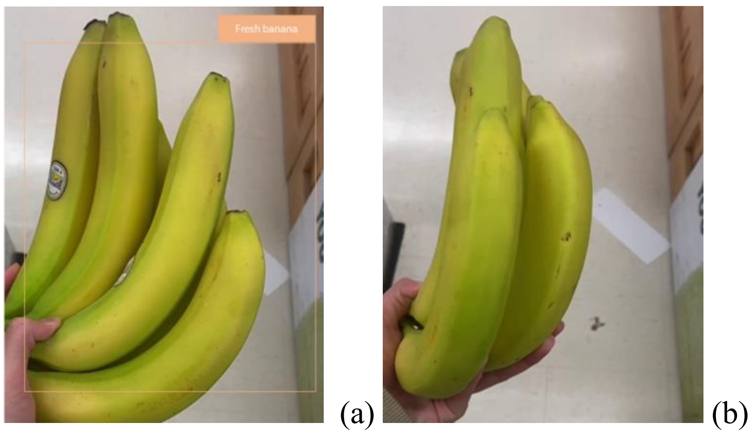


Figure 4.29 Prediction on the real video fresh bananas

Chapter 5

Analysis and Discussion

In this chapter, our experimental results are analyzed and compared. Comparisons of the results under various conditions will be detailed.

To better compare the result, we will show the final prediction results from three different model architectures (YOLOv6, YOLOv7, and YOLOv8). Because this is a classification task, we finally utilize precision and recall to predict the average performance, as we discussed in the previous chapters, precision indicates how many targets the classifier can detect with the correct label from the images. Recall is the score to show the ratio of our correct detection, a good classifier should have a high precision and a high recall at the same time. In this fresh fruit detection project, due toa rotten fruit may cause terrible disease. In this case, we decide a high recall is more important than a high precision, and the results from different deep learning models are shown in Table 5.1.

Table 5.1: The results for training our proposed models

Architectures	Avg precision	Avg recall
YOLOv6	0.846	0.884
YOLOv7	0.852	0.879
YOLOv8	0.861	0.892

The results in Table 5.1 are based on the trained model using the training dataset we discussed in Chapter 3 and Chapter 4. We get the training precision and recall. From the training results, we find that the precision for YOLO is always around 0.85, which indicates if there is a dataset including 100 different images, we can correctly label 85 of them, and the other 15 images may be classified as the wrong type or classified as a background. The recall for the YOLO model in the fruit detection is always around 0.88, which indicates if we detect fresh apples, finally the classifier can tell us that there are 50 images, then we can check manually there are 44 of them which are really the fresh apple images, the other 6 images are not fresh apples.

Table 5.2 shows the testing results. From Table 5.2, we know in the real scenario, if we can only apply a single classifier to conduct object detection on the fruit dataset, we highly recommend using the YOLOv8 model. If the YOLOv8 model doesn’t give a good result, then it is also worth trying YOLOv6.

Table 5. 2: The results for testing the proposed classifiers

Architecture	All classes IOU	All classes F1	The best threshold	The accumulated error rate
YOLOv6	0.975	0.96	0.659	12%
YOLOv7	0.972	0.96	0.284	9%
YOLOv8	0.979	0.96	0.569	14%

Table 5.2 shows the result for the test result, we see that YOLOv6 has a better result. From the confusion matrix, YOLOv8 has a better classification result. In YOLOv8, a large error only occurs for rotten orange detection, YOLOv8 has the highest average IOU, which indicates on average, the YOLOv8 model will have higher precision and recall. If we compare the best threshold, YOLOv7 has the lowest threshold, which indicates if we apply YOLOv7 to conduct classification, much more than the training set, the probability of wrong classification may increase. YOLOv7 tends to keep a lot of them as the detection result, this will automatically decrease the recall value.

YOLOv6 may have a much better performance compared with YOLOv8. In the future, if we test the network with fruit images, instead of using YOLOv8, we would like to forthputting YOLOv6 for object detection. In ensemble learning, we combine all the results from multiple architectures, it is better that YOLOv6 will be assigned a higher weight during the ensemble step.

In Table 5.2, it shows the accumulated error rate, this score is calculated from the confusion matrix we showed in the last section, for each network, we will sum up all the ratios for wrong classification blocks (blocks not on the confusion matrix diagonal), it looks like the error rate for YOLOv8 is a little bit higher than the error rate of YOLOv6, looks like YOLOv6 is better than YOLOv8, but YOLOv8 has a higher AUC, so combine the result from YOLOv6 and YOLOv8 may also be a good idea in the future work. If we combine the result we get from different architecture to the ensemble result, then we can get the following result:

Table 5.3: The average result

Architecture	Avg precision	Avg recall
Combine	0.912	0.924

Compared with the previous YOLOv6, YOLOv7, and YOLOv8 results, the combination of the result from three different architectures increases the precision and recall a lot, this is so great, which indicates our combination method really works for the fruit detection method, and the right now we only combine three different background to generate the final result. In the future, it is also worth trying to use more different architectures and combine all of their results together.

Chapter 6 Conclusion and Future Work

In this chapter, we will summarize the subject and method of this project and propose a new research direction according to the result and insufficiency of the experiment, preparing for future work.

6.1 Conclusion

In this thesis, we are using of fruit freshness dataset from Kaggle which includes six classes of images. We believe this dataset is general and widely used in the research area. In this dataset, fruit images were from fresh apples, fresh bananas, fresh oranges, rotten apples, rotten bananas, and rotten oranges. After obtaining the dataset, we split it into training and test datasets, following the normal preprocess steps. We split the dataset with as ratio of 8:2, which indicates we will randomly choose 80% data to train and the rest for testing our model. For the method to make the classification and detection, we also apply YOLOv6, YOLOv7, and YOLOv8 to generate the detection result automatically.

Compared to the results from different architectures, during the model training and testing, YOLOv8 will have the highest accuracy and training speed. Although just a 0.1 % increase, compared with YOLOv6 and a 0.5 % increase compared with YOLOv7. As we know YOLOv6 spent 24 hours to train the model with 300 epochs, YOLOv7 took more than 28 hours to train 300 epochs, but YOLOv8 spent 8 hours to train 300 epochs. Meanwhile, the YOLOv8 model is always very easy to employ as the whole architecture already has been built, we can easily apply the architecture with simple “import” command, but for YOLOv6 and YOLOv7 models, we need to build it by ourselves, but YOLO is very easy to employ compared with other types of architectures. And for we have multiple YOLO architectures that can detect the fruit freshness, we also creativity apply the majority vote method for those architectures result and generate a better result with higher precision and recall, what we find is when we use 20 percent of YOLOv6 and YOLOv7 individually and YOLOv8 result in 80 percent for the final result donations, then we will get precision with 0.97 and recall 0.963, which is the highest precision and recall we can get right now.

In the future, for base models, we are planning to apply YOLOv8 as a baseline of the architecture, we believe this will be a good choice, and we will also pay much attention to YOLOv6 detection results, since it can give us satisfactory results. For the YOLOv7 model, we will only employ it in the ensemble step if we need it in the future.

On the other side, in this project, the training and test datasets only include six different classes of visual objects, namely, “fresh apple”, “rotten apple”, “fresh banana”, “rotten banana”, “fresh orange", "rotten orange', which are far less compared with the real scenario. There are perhaps more than thousands of types of fruits around the world, and the appearance of rotten or freshness of fruits will be normal.

Regarding fruit skins, black may mean rotten, but for the datasets we use in this project, rotten apples will look black or yellow, rotten oranges appear grey or green, and rotten bananas will be black, it will also be pretty easy for us to classify them based on colors of the images.

In the next step, we will find a much larger dataset with more images and a greater number of classes of fruit images. We plan to have almost all types of fruits we can find in our daily life. For example, watermelon, grape, lemon, melon, strawberry and blueberry, and blueberry will be available in the image dataset.

From the previous result, we find that with the majority vote method, we can get better performance compared with the previous best result of the YOLOv8 model. Because of the majority vote, for each detection, we can always keep the best detection result to make our final decision, as the proverb goes "Many hands make the light work", if we add more detection results to make our final decision, the net performance can become better and better. So, we will continue working on this part and plan to add more powerful architectures to our final architecture.

We are also planning to use postprocessing methods. For example, the ensemble methods are used for different learners or classifiers, as we discussed in Chapter 5, with this new method, we can continue to increase the performance of our method, and we can harness different weights for different architectures to get a better result.

6.2 Our Future Work

In the future, we will collect more real fruit images for the model training which will be more effective to improve the accuracy of our proposed models, for a network, we can enlarge the dataset or add more classes of image data. For example, with a fruit dataset, we can add more fruits; or for each fruit class, we can add more classes, like "freshness", "not freshness", "a little bit rotten", and "totally rotten".

For the training set, it will always be a good idea to improve the performance of the proposed architecture. With those new datasets, we plan to add more daily fruits in the dataset such as “strawberries”, “blueberry”, “grapes”, and “watermelon”.

This will be very hard, in the reason of a rotten blueberry will have the same color and shape as a ripe blueberry, we will determine whether the blueberry is rotten or ripe is also very hard even for a human, there are a lot of different grapes, some are green and some are purple, it is better for us to split them as some sub training set.

With the new dataset, we will add real backgrounds for each image, through this way, our dataset will be like the real scene, and the network trained with this dataset will also become more robust.

Meanwhile, in our future work, we will not only focus on the YOLO models for image classification and detection, but also embark on more and more new architectures, like auto-encoder which are the hottest architecture in the research area. We will try even GPT architecture which is the most famous architecture right now. With GPT-4, it can also process the images. With the SAM (segmentation all model) network, we can segment all images for a specific demand, we will finetune the previous architecture.

Back to this project, for the next step, we will pay more attention to YOLOv6 and YOLOv8, we are use of YOLOv8 as our backbone, and YOLOv6 will be served as a reference for the result from YOLOv8. Throughout this project, we will conduct object detection with a higher accuracy. Although YOLOv7 does not have a good performance compared with YOLOv8 or YOLOv6, it will also be worth having a try, since YOLOv7 still has a very high detection accuracy, we will keep generating more results with the YOLOv7 model and make it as a part of our architecture during the ensemble step. We will also add more ensemble methods in the future, like voting with different weights, or even combine the network of YOLOv6, YOLOv7, and YOLOv8 directly to generate a new strategy for the fruit freshness detection (Lin, 2020).

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., & Ghemawat, S. (2016) TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467.*

Abdulrahman, A., & Iqbal, K. (2014) Capturing human body dynamics using RNN based on persistent excitation data generator. *International Symposium on Computer-Based Medical Systems (CBMS),* (pp. 221-226).

Al-Sarayreh, M., Reis, M., Yan, W., Klette, R. (2019) A sequential CNN approach for foreign object detection in hyperspectral images. International Conference on Information, Communications and Signal.

Al-Sarayreha, M., Reis, M., Yan, W., Klette, R. (2020) Potential of deep learning and snapshot hyperspectral imaging for classification of species in meat. Food Control.

Al-Sarayreha, M. (2020) Hyperspectral Imaging and Deep Learning for Food Safety. PhD Thesis. Auckland University of Technology, New Zealand.

An, N., Yan, W. (2021) Multitarget tracking using Siamese neural networks. ACM Transactions on Multimedia Computing, Communications and Applications.

An, N. (2020) Anomalies Detection and Tracking Using Siamese Neural Networks. Master's Thesis. Auckland University of Technology, New Zealand.

Anderson, C., Burt, P., & Van Der Wal, G. (1985). Change detection and tracking using pyramid transform techniques. *Cambridge Symposium* (pp. 72-78) International Society for Optics and Photonics.

Baum, L., E., & Sell, G. (1968). Growth transformations for functions on manifolds. *Pacific Journal of Mathematics*, *27*(2), 211-227.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157-166.

Barnea, E., Mairon, R., & Ben-Shahar, O. (2016). Colour-agnostic shape-based 3D fruit detection for crop harvesting robots. Biosystems Engineering, 146, 57-70.

Bulanon, D. M., Burks, T. F., & Alchanatis, V. (2009). Image fusion of visible and thermal images for fruit detection. Biosystems Engineering, 103(1), 12-22.

Chatzis, S. P., & Kosmopoulos, D. I. (2011). A variational Bayesian methodology for hidden Markov models utilizing Student's-*t* mixtures. Pattern Recognition, *44*(2), 295-306.

Chen, Y. N., Han, C. C., Wang, C. T., Jeng, B. S., & Fan, K. C. (2006) The application of a convolution neural network on the face and license plate detection. *International Conference on Pattern Recognition,* (Vol. 3, pp. 552-555). IEEE.

Collins, R. T., Lipton, A. J., Kanade, T., Fujiyoshi, H., Duggins, D., Tsin, Y., & Wixson, L. (2000) A System for video surveillance and monitoring. Research Report. The Robotics Institute, Carnegie Mellon University, Pittsburgh PA

Chen, S., Sun, P., Song, Y., et al. DiffusionDet: Diffusion model for object detection. arXiv preprint arXiv:2211.09788, 2022.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078.*

Chen, C., Fan, Q., Panda, R. (2021) CrossVit : Cross- attention multi-scale vision transformer for image classification. *IEEE/CVF International Conference on Computer Vision*, 357 – 366.

Cheng, B., Misra, I., Schwing, A., et al. Masked-attention mask transformer for universal image segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, 1290-1299.

Cummins, F., Gers, F., Schmidhuber, J. (1999) Language identification from prosody without explicit features. *EUROSPEECH'99,* pages 371–374, 1999

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv : 1412. 3555.*

Chong, Y., Tay, Y. (2017) Abnormal event detection in videos using spatiotemporal autoencoder. *Advances in Neural Networks-ISNN* pages 189 – 196. Springer.

Eickeler, S., & Muller, S. (1999). Content-based video indexing of TV broadcast news using hidden Markov models. *IEEE International Conference on Acoustics, Speech, and Signal Processing,* Vol. 6, pp. 2997-3000.

Fang, W., Lin, W., Ren, P. (2019) Tinier-YOLO: A real-time object detection method for constrained environments. IEEE Access, 8: 1935 - 1944.

Fang, Y., et al. (2021) You only look at one sequence: Rethinking transformer in vision through object detection. *Advances in Neural Information Processing Systems*, 34: 26183 – 26197.

Fu, R., Zhang, Z. and Li, L. (2016). Using LSTM and GRU neural network methods for traffic flow prediction. *Youth Academic Annual Conference of Chinese Association of Automation (YAC).*

Fu, Y., Nguyen, M., Yan, W. (2022) Grading methods for fruit freshness based on deep learning. Springer Nature Computer Science.

Fu, Y. (2020) Fruit Freshness Grading Using Deep Learning. Master's Thesis. Auckland University of Technology, New Zealand.

Galvez, R., Bandala, A., Dadios, E., et al. (2018) Object detection using convolutional neural networks. IEEE TENCON: 2023-2027.

Gers, F. A., & Schmidhuber, J. (2000). Recurrent nets that time and count. Neural Networks, pp. 189-194.

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural Computation, 12*(10), 2451 – 2471.

Gers, F. A., Schraudolph, N. N., & Schmidhuber, J. (2002). Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research,* 115 – 143.

Gers, F. A., & Schmidhuber, E. (2001). LSTM recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks, 12* (6), 1333 – 1340.

Gers, F. A., Schraudolph, N. N., & Schmidhuber, J. (2002). Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research, 3,* 115 – 143.

Girshick, R. Fast R-CNN. *IEEE International Conference on Computer Vision*, pages 1440 – 1448.

Gowdra, N., Sinha, R., MacDonell, S., Yan, W. (2021) Maximum Categorical Cross Entropy (MCCE): A noise-robust alternative loss function to mitigate racial bias in Convolutional Neural Networks (CNNs) by reducing overfitting. Pattern Recognition.

Gowdra, N. (2021) Entropy-Based Optimization Strategies for Convolutional Neural Networks. PhD Thesis, Auckland University of Technology, New Zealand.

Haritaoglu, I., Harwood, D., & Davis, L. S. (2000). W/sup 4: Real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22* (8), 809 – 830.

Han, X., Gao, Y., Lu, Z., Zhang, Z., & Niu, D. (2015). Research on moving object detection algorithm based on improved three frame difference method and optical flow. *International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC)* (pp. 580-584).

Heikkila, M., & Pietikainen, M. (2006). A texture-based method for modeling the background and detecting moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 28*(4), 657-662.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735 - 1780.

Hori, T., Kubo, Y., & Nakamura, A. (2014). Real-time one-pass decoding with recurrent neural network language model for speech recognition. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 6364 - 6368).

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology, 160* (1), 106 - 154.

Huang, X., Wang, X., Lv, W., et al. (2021) PP-YOLOv2: A practical object detector. arXiv preprint arXiv:2104.10419.

Joseph, R., Divvala, S., Girshick, R., Farhadi, A. (2021) You only look once: Unified, real-time object detection. *IEEE Conference on Computer Vision and Pattern Recognition*, 779 – 788.

Ji, H., Liu, Z., Yan, W., Klette, R. (2019) Early diagnosis of Alzheimer's disease based on selective kernel network with spatial attention. Asian Conference on Pattern Recognition.

Ji, H., Yan, W., Klette, R. (2019) Early diagnosis of Alzheimer’s disease using deep learning. ACM ICCCV.

Jiao, Y., Weir, J., Yan, W. (2011) Flame detection in surveillance. Journal of Multimedia 6 (1).

Khan, R., Debnath, R. (2015) Multi class fruit classification using efficient object detection and recognition techniques. International Journal of Image, Graphics and Signal Processing.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1725 - 1732).

Katagiri, S., & Lee, C. H. (1993). A new hybrid algorithm for speech recognition based on HMM segmentation and learning vector quantization. *IEEE Transactions on Speech and Audio Processing, 1* (4), 421 – 430.

Kang, H., & Chen, C. (2020). Fast implementation of real-time fruit detection in apple orchards using deep learning. Computers and Electronics in Agriculture, 168, 105108.

Kang, H., & Chen, C. (2020). Fruit detection, segmentation and 3D visualisation of environments in apple orchards. *Computers and Electronics in Agriculture*, 171, 105302.

Kang, H., & Chen, C. (2019). Fruit detection and segmentation for apple harvesting using visual sensor in orchards. *Sensors*, 19(20), 4599.

Kim, J., & Kim, H. (2016). Classification performance using gated recurrent unit recurrent neural network on energy disaggregation. International Conference *on Machine Learning and Cybernetics (ICMLC)* (pp.105–110).

Krizhevsky, A., Sutskever, I., Hinton, G. (2017) ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60(6):84–90.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems* (pp. 1097 – 1105).

Koirala, A., Walsh, K. B., Wang, Z., & McCarthy, C. (2019). Deep learning for real-time fruit detection and orchard fruit load estimation: Benchmarking of "MangoYOLO". Precision Agriculture, 20, 1107-1135.

Kuang, H., Liu, C., Chan, L. L. H., & Yan, H. (2018). Multi-class fruit detection based on image region selection and improved object proposals. Neurocomputing, 283, 241-255.

Le, R., Nguyen, M., Yan, W. (2018) A vision aid for the visually impaired using commodity dual-rear-camera smartphones. International Conference on Mechatronics and Machine Vision.

Le, R., Nguyen, M., Yan, W. (2019) A web-based augmented reality approach to instantly view and display 4D medical images. Asian Conference on Pattern Recognition.

Le, R., Nguyen, M., Nguyen, Q., Nguyen, H., Yan, W. (2020) Automatic data generation for deep learning model training of image classification used for augmented reality on pre-school books. International Conference on Multimedia Analysis and Pattern Recognition.

Le, R., Nguyen, M., Yan, W. (2020) Machine learning with synthetic data – A new way to learn and classify the pictorial augmented reality markers in real-time. International Conference on Image and Vision Computing New Zealand.

Le, R., Nguyen, M., Yan, W., Nguyen, H. (2021) Augmented reality and machine learning incorporation using YOLOv3 and ARKit. Applied Sciences.

Le, R. (2022) Synthetic Data Annotation for Enhancing the Experiences of Augmented Reality Application Based on Machine Learning (PhD Thesis). Auckland University of Technology, New Zealand.

Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., Ke, Z., Li, Q., Cheng, M., Nie, W., et al. (2022) YOLOv6: A single-stage object detection framework for industrial applications. arXiv preprint arXiv: 2209. 02976.

Lin, G., et al. (2020) Color-, depth-, and shape-based 3D fruit detection. Precision Agriculture, 21: 1 – 17.

Lin, T., Dollár, P., Girshick, R., et al. (2017) Feature pyramid networks for object detection. IEEE Conference on Computer Vision and Pattern Recognition, 2117 - 2125.

Liu, Z., Yan, W., Yang, B. (2018) Image denoising based on a CNN model. International Conference on Control, Automation and Robotics.

LeCun, Y. (1989). Generalization and network design strategies. Connectionism in Perspective, 143 – 155.

LeCun, Y., & Ranzato, M. (2013). Deep learning tutorial. *Tutorials in International Conference on Machine Learning (ICML'13).*

Liu, Y., Sun, P., Wergeles, N., et al. (2021) A survey and performance evaluation of deep learning methods for small object detection. Expert Systems with Applications, 172: 114602.

Liu, H., & Hou, X. (2012). Moving detection research of background frame difference based on Gaussian model. *International Conference on Computer Science & Service System (CSSS),* (pp. 258 - 261). IEEE.

Lin, G., Tang, Y., Zou, X., Cheng, J., & Xiong, J. (2020). Fruit detection in natural environment using partial shape matching and probabilistic Hough transform. Precision Agriculture, 21, 160-177.

Moghaddam, B., Pentland, A. (1995) Probabilistic visual learning for object detection. *IEEE International Conference on Computer Vision*, 786-793.

Matsugu, M., Mori, K., Mitari, Y., & Kaneda, Y. (2003). Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural Networks, 16* (5), 555 - 559.

Maryam, K., & Reza, K. M. (2012). An analytical framework for event mining in video data. *Artificial Intelligence Review, 41* (3), pp. 401 – 413.

Mikolov, T., Kombrink, S., Burget, L., Černocký, J., & Khudanpur, S. (2011). Extensions of recurrent neural network language model. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* (pp. 5528 - 5531).

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. *Interspeech* (Vol. 2, p. 3).

Narendra, K. S., & Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks, 1* (1), 4 – 27.

Noorit, N., & Suvonvorn, N. (2014). Human activity recognition from basic actions using finite state machine. *International Conference on Advanced Data and Information Engineering (DaEng - 2013)* (pp. 379 - 386). Springer.

Ng, J., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., & Toderici, G. (2015). Beyond short snippets: Deep networks for video classification. *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4694 – 4702).

Pan, C., Yan, W. (2018) A learning-based positive feedback in salient object detection. International Conference on Image and Vision Computing New Zealand.

Pan, C., Yan, W. (2020) Object detection based on saturation of visual perception. Multimedia Tools and Applications, 79 (27-28), 19925-19944.

Pan, C., Liu, J., Yan, W., Zhou, Y. (2021) Salient object detection based on visual perceptual saturation and two-stream hybrid networks. IEEE Transactions on Image Processing.

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. *ICML (3), 28,* 1310 – 1318.

Petrushin, V. A. (2005). Mining rare and frequent events in multi-camera surveillance video using self-organizing maps. *ACM SIGKDD international Conference on Knowledge Discovery in Data Mining,* pp. 794 – 800.

Popoola, O. P., & Wang, K. (2012). Video-based abnormal human behavior recognition—A review. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42* (6), 865 – 878.

Parico, A., Borja, A., Ahamed, T. (2021) Real time pear fruit detection and counting using YOLOv4 models and deep SORT. Sensors, 21(14): 4803.

Parico, A., Ahamed, T. (2021) Real time pear fruit detection and counting using YOLOv4 models and deep SORT. Sensors, 21(14): 4803.

Qi, J., Nguyen, M., Yan, W. (2022) Small visual object detection in smart waste classification using Transformers with deep learning. International Conference on Image and Vision Computing New Zealand (IVCNZ).

Qi, J., Nguyen, M., Yan, W. (2022) Waste classification from digital images using ConvNeXt. Pacific-Rim Symposium on Image and Video Technology (PSIVT).

Qi, J., Nguyen, M., Yan, W. (2023) CISO: Co-iteration semi-supervised learning for visual object detection. Multimedia Tools and Applications.

Qi, J., Nguyen, M., Yan, W. (2024) NUNI-Waste: Novel semi-supervised semantic segmentation for waste classification with non-uniform data augmentation. Multimedia Tools and Applications.

Rabiner, L., & Juang, B. (1986). An introduction to hidden Markov models. *IEEE ASSP, 3* (1), 4 – 16.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE, 77* (2), 257 – 286.

Remagnino, P., Monekosso, D. N., & Jain, L. C. (Eds.). (2011). *Innovations in Defence Support Systems-3: Intelligent Paradigms in Security* (Vol. 336). Springer Science & Business Media.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 779-788).

Ren, S., He, K., Girshick, R., and Sun, J. (2015) Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems,* 28.

Sa, Inkyu, et al. (2016) Deepfruits: A fruit detection system using deep neural networks. Sensors, 16 (8): 1222.

Sak, H., Senior, A., & Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. *Annual Conference of the International Speech Communication Association.*

Schwenk, H. (2007). Continuous space language models. *Computer Speech & Language*, *21* (3), 492 - 518.

Shen, D., Xin, C., Nguyen, M., Yan, W. (2018) Flame detection using deep learning. International Conference on Control, Automation and Robotics.

Sisson, J. H., Stoner, J. A., Ammons, B. A., & Wyatt, T. A. (2003). All-digital image capture and whole-field analysis of ciliary beat frequency. *Journal of Microscopy, 211* (2), 103 – 111.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). DropOut: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, *15* (1), 1929 - 1958.

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

Sun, Z., Cao, S., Yang, Y., & Kitani, K. M. (2021). Rethinking transformer-based set prediction for object detection. *IEEE International Conference on Computer Vision* (pp. 3611 - 3620).

Tang, Y., Huang, Y., Wu, Z., Meng, H., Xu, M., & Cai, L. (2016). Question detection from acoustic features using recurrent neural network with gated recurrent unit. *International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 6125 - 6129).

Trinh, H., Fan, Q., Jiyan, P., Gabbur, P., Miyazawa, S., & Pankanti, S. (2011). Detecting human activities in retail surveillance using hierarchical finite state machine. *International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1337 - 1340).

Ukwuoma, C. C., Zhiguang, Q., Bin Heyat, M. B., Ali, L., Almaspoor, Z., & Monday, H. N. (2022). Recent advancements in fruit detection and classification using deep learning techniques. Mathematical Problems in Engineering, 2022, 1-29.

Vallayil, M., Nand, P., Yan, W., Allende-Cid, H. (2023) Explainability of automated fact verification systems: A comprehensive review. Applied Science, 13(23) 1260

Varol, G., Laptev, I., & Schmid, C. (2016). Long-term temporal convolutions for action recognition. *arXiv preprint arXiv:1604. 04494.*

Vasconez, J., et al. (2020) Comparison of convolutional neural networks in fruit detection and counting: A comprehensive evaluation. Computers and Electronics in Agriculture, 173: 105348.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., and Polosukhin, I. (2017) Attention is all you need. *Advances in Neural Information Processing Systems*, 30.Vig, J. (2019) A multiscale visualization of attention in the transformer model. arXiv preprint arXiv : 1906. 05714.

Wan, S., Goudos, S. (2020) Faster R-CNN for multi-class fruit detection using a robotic vision system. Computer Networks, 168: 107036.

Wan, S., & Goudos, S. (2020). Faster R-CNN for multi-class fruit detection using a robotic vision system. Computer Networks, 168, 107036.

Wang, J., Yan, W., Kankanhalli, M., Jain, R., Reinders, M. (2003) Adaptive monitoring for video surveillance. International Conference on Information, Communications and Signal Processing.

Wang, J., Kankanhalli, M., Yan, W., Jain, R. (2003) Experiential sampling for video surveillance. ACM SIGMM International Workshop on Video surveillance (pp.77-86).

Wu, X., Sahoo, D., Hoi, S. (2020) Recent advances in deep learning for object detection. Neurocomputing, 396: 39-64.

Wildes, R. P. (1998). A measure of motion salience for surveillance applications. *International Conference on Image Processing* (pp. 183 - 187). IEEE.

Xia, Y., Nguyen, M., Yan, W. (2022) A real-time Kiwifruit detection based on improved YOLOv7. International Conference on Image and Vision Computing New Zealand (IVCNZ)

Xia, Y., Nguyen, M., Yan, W. (2023) Kiwifruit counting using KiwiDetector and KiwiTracker. IntelliSys conference.

Xia, Y., Nguyen, M., Yan, W. (2023) Multiscale Kiwifruit detection from digital images. PSIVT.

Xiang, Y., Yan, W. (2021) Fast-moving coin recognition using deep learning. Springer Multimedia Tools and Applications.

Xiao, B., Nguyen, M., Yan, W. (2021) Apple ripeness identification using deep learning. International Symposium on Geometry and Vision.

Xiao, B., Nguyen, M., Yan, W. (2023) Apple ripeness identification from digital images using transformers. Multimedia Tools and Applications, Springer Science and Business Media LLC.

Xiao, B., Nguyen, M., Yan, W. (2023) Fruit ripeness identification using transformers. Applied Intelligence, Springer Science and Business Media LLC.

Xiao, B., Nguyen, M., Yan, W. (2023) A mixture model for fruit ripeness identification in deep learning. Handbook of Research on AI and ML for Intelligent Machines and Systems, pp.1-16, Chapter 16, IGI Global.

Xiao, B., Nguyen, M., Yan, W. (2023) Fruit ripeness identification using YOLOv8 model. Multimedia Tools and Applications.

Xin, C. (2020) Detection and Recognition for Multiple Flames Using Deep Learning. Master's Auckland University of Technology, New Zealand.

Xin, C., Nguyen, M., Yan, W. (2020) Multiple flames recognition using deep learning. Handbook of Research on Multimedia Cyber Security, 296-307.

Yan, W. (2019) Introduction to Intelligent Surveillance: Surveillance Data Capture, Transmission, and Analytics. Springer Nature.

Yan, W. (2023) Computational Methods for Deep Learning: Theory, Algorithms, and Implementations. Springer Nature.

Yan, W., Nguyen, M., Stommel, M. (2023) International Conference on Image and Vision Computing (IVCNZ 2022), Springer Nature LNCS 13836

Yan, W., Nguyen, M., Stommel, M. (2024) Pacific Conference on Image and Video Technology (PSIVT 2023), Springer Nature LNCS 14403

Yang, Y., et al. (2013). A novel motion object detection method based on improved frame difference and improved Gaussian mixture model. *International Conference on Measurement, Information and Control* (Vol. 1, pp. 309 – 313).

Yu, Y., Zhang, K., Yang, L., et al. (2019) Fruit detection for strawberry harvesting robot in non-structural environment based on Mask R-CNN. Computers and Electronics in Agriculture, 163: 104846.

Yu, Y., Zhang, K., Yang, L., & Zhang, D. (2019). Fruit detection for strawberry harvesting robot in non-structural environment based on Mask R-CNN. Computers and Electronics in Agriculture, 163, 104846.

Zaremba, W., Sutskever, I., & Vinyals, O. (2014). Recurrent neural network regularization. *arXiv preprint arXiv:1409. 2329.*

Zaremba, W. (2015). An empirical exploration of recurrent network architectures. *International Conference on Machine Learning, Lille, France.*

Zhang, Y., Er, M. J., Venkatesan, R., Wang, N., & Pratama, M. (2016). Sentiment classification using comprehensive attention recurrent models. *International Joint Conference on Neural Networks (IJCNN)* (pp. 1562 - 1569).

Zhang, Y., et al. (2022) Complete and accurate holly fruits counting using YOLOX object detection. Computers and Electronics in Agriculture 198: 107062.

Zhang, Y., Zhang, E., Chen, W. (2016) Deep neural network for halftone image classification based on sparse auto-encoder. Engineering Applications of Artificial Intelligence, 50:245 – 255.

Zhao, Q., Zheng, P., Xu, S., et al. (2019) Object detection with deep learning: A review. IEEE Transactions on Neural Networks and Learning Systems, 30(11): 3212-3232.

Zou, Z., et al. (2023) Object detection in 20 years: A survey. Proceedings of the IEEE.