

# Deep Learning for Fruit Ripeness Determination

**Ziang Zhao**

School of Engineering  
Cardiff University

A thesis submitted for the degree of  
*Doctor of Philosophy*

June 2025



# Abstract

Fruits play a fundamental role in human nutrition, serving as a key source of essential vitamins and minerals. However, the global fruit industry is facing a significant challenge: the shortage of labour for harvesting, which remains a predominantly manual task. Automated fruit-harvesting robots present a promising solution to address this labour gap and maintain stable production. These robots can operate continuously without fatigue, yet they struggle to accurately assess fruit ripeness, which is a critical factor influencing harvest quality and timing. While numerous laboratory-based techniques for evaluating ripeness have been developed, their application in field settings is limited due to the complex and variable conditions of real-world orchards.

To address these challenges, this thesis explores deep learning for determining fruit ripeness through vision models and high-quality fruit image datasets. Specifically, this thesis introduces NinePeach, a large dataset of peach images, and PeachSOLO, a one-stage model designed for peach instance segmentation. PeachSOLO achieves an average precision (AP) of 72.12, surpassing Mask R-CNN (69.91 AP). This thesis then proposes LightStraw, a lightweight model for strawberry instance segmentation. It requires considerably fewer parameters (17.42M) and floating-point operations (78.3G) than Mask R-CNN (35.08M/877.4G). This thesis also combines peach and strawberry images into a single dataset and proposes a query-based segmentation model FruitQuery. FruitQuery achieves the best AP of 67.02 with only 14.08M parameters, outperforming 13 other models with 33 variants, including three series of YOLO. Finally, this thesis develops AppleSSL, a self-supervised method for assessing in-field apple ripeness under occlusion. Using less than 1% labelled images, AppleSSL reconstructs obscured parts and provides ripeness scores from 0.0 to 1.0, surpassing 15 other self-supervised methods and one supervised method.

Overall, this thesis demonstrates that deep learning can enable practical, accurate, and efficient ripeness estimation in real-world environments, supporting robotic fruit picking and contributing to smart, precision agriculture.

# Acknowledgement

If the multiverse truly exists, then I must be the luckiest version of myself, the one who has made it this far. For everything I have been given, been taught, learned, gained and achieved, I express my sincerest gratitude. I am also grateful for all the kindness I have been fortunate to receive throughout this journey.

First and foremost, my deepest gratitude goes to my supervisors, Dr Yulia Hicks and Dr Xianfang Sun, for their continuous guidance and support. This work would not be possible without their instruction and patience. Special thanks to Yulia for her invaluable efforts in funding acquisition, which made my visiting study in New Zealand come true.

Second, I would like to thank Prof Chaoxi Luo, Dr Benjamin McGuinness, and Dr Hin Lim for their help in data collection and great professional ideas for this research. Working with them helped me understand the importance of cooperating with people from different disciplines.

Then, I would like to thank my family and friends for their unwavering love and encouragement. Without their constant support, I would not have been able to complete this research and present this thesis.

Last but not least, I sincerely appreciate the China Scholarship Council and Cardiff University, whose contributions have laid a solid foundation for this research.



# Contents

<b>Acknowledgement</b>	<b>ii</b>
<b>List of Figures</b>	<b>viii</b>
<b>List of Tables</b>	<b>x</b>
<b>Nomenclature</b>	<b>xiii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Research Motivation . . . . .	1
1.2 Problem Statement . . . . .	2
1.3 Research Contributions . . . . .	2
1.4 Publications . . . . .	4
1.5 Thesis Outline . . . . .	5
<b>2 Literature Review</b>	<b>7</b>
2.1 Background of Fruit Ripeness . . . . .	7
2.1.1 Introduction . . . . .	7
2.1.2 Biology of Fruit Ripening . . . . .	8
2.2 Sensing Methods for Ripeness Determination . . . . .	9
2.2.1 Colour Inspection . . . . .	9
2.2.2 Spectral Imaging . . . . .	10
2.2.3 Spectroscopy . . . . .	11
2.2.4 Fluorescence . . . . .	11
2.2.5 Aroma . . . . .	14
2.2.6 Summary . . . . .	14
2.3 Machine Learning for Ripeness Determination . . . . .	15
2.3.1 Introduction . . . . .	15
2.3.2 Concepts and Algorithms . . . . .	15
2.3.3 Applications . . . . .	16
2.3.4 Summary . . . . .	17
2.4 Deep Learning for Ripeness Determination . . . . .	18

2.4.1	Introduction . . . . .	18
2.4.2	Deep Learning Structure . . . . .	19
2.4.3	Computer Vision Tasks . . . . .	24
2.4.4	Self-Supervised Learning . . . . .	38
2.5	Summary . . . . .	42
<b>3</b>	<b>PeachSOLO: A One-stage Instance Segmentation Model for Peach Ripeness Classification</b>	<b>43</b>
3.1	Introduction . . . . .	43
3.2	Dataset . . . . .	44
3.2.1	Image Collection . . . . .	44
3.2.2	Image Preprocessing . . . . .	44
3.2.3	Dataset Summary . . . . .	47
3.3	Method . . . . .	47
3.3.1	Model Structure . . . . .	47
3.3.2	Model Training . . . . .	50
3.3.3	Model Inference . . . . .	52
3.4	Experiments and Results . . . . .	53
3.4.1	Experiments . . . . .	53
3.4.2	Main Results . . . . .	54
3.4.3	Ablation Results . . . . .	55
3.4.4	Combined and Separate Training . . . . .	57
3.4.5	Visualisation . . . . .	57
3.4.6	Model Complexity . . . . .	59
3.5	Discussion . . . . .	60
3.5.1	The Details of NinePeach . . . . .	60
3.5.2	Limitations . . . . .	60
3.6	Summary . . . . .	61
<b>4</b>	<b>LightStraw: Lightweight CNN-based Strawberry Instance Segmentation Models</b>	<b>62</b>
4.1	Introduction . . . . .	62
4.2	Dataset . . . . .	63
4.2.1	StrawDI_Db1 Overview . . . . .	63
4.3	Method . . . . .	63
4.3.1	Model Structure . . . . .	63
4.3.2	Encoder . . . . .	64
4.3.3	Decoder . . . . .	67
4.3.4	Loss Function . . . . .	69

4.4	Experiments and Results . . . . .	70
4.4.1	Experiments . . . . .	70
4.4.2	Results . . . . .	70
4.4.3	Visualisation . . . . .	73
4.4.4	Deployment . . . . .	73
4.5	Discussion . . . . .	74
4.5.1	Limitations . . . . .	74
4.5.2	Future Work . . . . .	74
4.6	Summary . . . . .	75
<b>5</b>	<b>FruitQuery: Lightweight Query-based Segmentation Models for In-field Fruit Ripeness Determination</b>	<b>76</b>
5.1	Introduction . . . . .	76
5.2	Dataset . . . . .	77
5.2.1	Overview . . . . .	77
5.2.2	StrawDI_Db1 Ripeness Annotation . . . . .	77
5.2.3	Dataset Summary . . . . .	78
5.3	Method . . . . .	79
5.3.1	Model Structure . . . . .	79
5.3.2	Loss Function . . . . .	85
5.4	Experiments and Results . . . . .	86
5.4.1	Experiments . . . . .	86
5.4.2	Main Results . . . . .	87
5.4.3	Ablation Experiments . . . . .	92
5.4.4	Combined and Separate Training . . . . .	93
5.4.5	Model Parameter Distribution . . . . .	94
5.4.6	Visualisation . . . . .	95
5.4.7	Class Activation Map Analysis . . . . .	97
5.4.8	Deployment . . . . .	98
5.5	Discussion . . . . .	99
5.5.1	Comparison to PeachSOLO . . . . .	99
5.5.2	Limitations . . . . .	99
5.5.3	Future Work . . . . .	100
5.6	Summary . . . . .	100
<b>6</b>	<b>AppleSSL: A Novel Self-supervised Method for In-field Occluded Apple Ripeness Determination</b>	<b>102</b>
6.1	Introduction . . . . .	102
6.1.1	Ripeness labelling . . . . .	102

6.1.2	In-field Occlusion . . . . .	104
6.1.3	Contributions . . . . .	105
6.2	Dataset . . . . .	106
6.2.1	Image Collection . . . . .	106
6.2.2	Image Preprocessing . . . . .	106
6.2.3	Image Augmentation . . . . .	107
6.3	Method . . . . .	108
6.3.1	Overview . . . . .	108
6.3.2	Reconstructor . . . . .	109
6.3.3	Extractor . . . . .	111
6.3.4	Predictor . . . . .	115
6.4	Experiments and Results . . . . .	116
6.4.1	Experiments . . . . .	116
6.4.2	Reconstructor . . . . .	116
6.4.3	Extractor . . . . .	120
6.4.4	Predictor . . . . .	122
6.5	Discussion . . . . .	127
6.5.1	Transfer to Other Fruits . . . . .	127
6.5.2	Limitations . . . . .	127
6.5.3	Future Work . . . . .	128
6.6	Summary . . . . .	129
<b>7</b>	<b>Conclusions and Future Work</b>	<b>130</b>
7.1	Research Summary . . . . .	130
7.2	Key Contributions . . . . .	130
7.3	Limitations . . . . .	132
7.4	Future Work . . . . .	133
	<b>References</b>	<b>135</b>
	<b>Appendix</b>	<b>163</b>

# List of Figures

2.1	Ripeness is a key factor throughout the whole process of fruit production.	7
2.2	Some sensing methods used for determining fruit ripeness. . . . .	10
2.3	The example of using K-means to cluster fruits. . . . .	16
2.4	The relation between deep learning, machine learning and artificial intelligence. . . . .	18
2.5	A biological neural network. . . . .	18
2.6	A deep learning network for image classification. . . . .	19
2.7	The typical structure of a deep learning model. . . . .	20
2.8	Some deep learning applications in agriculture. . . . .	25
2.9	The vision tasks of deep learning. . . . .	26
2.10	Some fruit harvesting robots. . . . .	36
2.11	The comparison of supervised, unsupervised, and self-supervised learning.	38
2.12	Contrastive learning (top) and masked image modelling (bottom). . . . .	39
3.1	The 9 cultivars of peach in NinePeach. . . . .	45
3.2	The overview of Label Studio. . . . .	45
3.3	Instance category distribution of NinePeach. . . . .	46
3.4	Original image (left), individual masks (middle), and annotated image (right). . . . .	46
3.5	The architecture of the proposed PeachSOLO. . . . .	48
3.6	The training loss (solid line, left ordinate) and evaluation AP (dash line, right ordinate) of the proposed PeachSOLO. . . . .	54
3.7	Segmentation visualisations of PeachSOLO, with accurate (left) and non-accurate (right) examples. . . . .	58
3.8	The segmentation comparison of Mask R-CNN (top) and the proposed PeachSOLO (below). . . . .	59
4.1	Sample images and annotations of StrawDI_Db1. . . . .	64
4.2	The architecture of the proposed LightStraw. . . . .	64
4.3	The backbone of the proposed LightStraw. . . . .	66
4.4	Non-maximum suppression and bipartite matching. . . . .	68

4.5	The segmentation visualisation of the proposed LightStraw. . . . .	72
4.6	NVIDIA Jetson Orin Nano. . . . .	73
5.1	The process of strawberry mask classification. . . . .	78
5.2	Examples of fruit instance annotation for the StrawDI_Db1. . . . .	79
5.3	The proposed backbone of FruitQuery. . . . .	81
5.4	The architecture of FruitQuery. . . . .	82
5.5	The illustration of the Pyramid Pooling Module. . . . .	84
5.6	Segmentation visualisations of FruitQuery on NinePeach (top) and StrawDI_Db1 (bottom). . . . .	95
5.7	The segmentation comparison of YOLOv9 and FruitQuery. . . . .	96
5.8	The CAM comparison of YOLOv9 and FruitQuery. . . . .	97
6.1	Apples with distinct ripeness difference can appear simultaneously. . . . .	103
6.2	Different users have different criteria for apple ripeness. . . . .	103
6.3	Example of modal and amodal masks [79]. . . . .	104
6.4	The apple orchard in New Zealand (left) and samples of apple images (right). . . . .	105
6.5	The workflow of image preprocessing. . . . .	106
6.6	The selected 20 fully unripe and 20 fully ripe apples. . . . .	107
6.7	The Fr distribution of the dataset. . . . .	107
6.8	The original image (left) and examples generated via augmentation (right). . . . .	108
6.9	The overall architecture of this study. . . . .	109
6.10	The architecture of the proposed AppleSSL. . . . .	110
6.11	The proposed distances for model performance evaluation. . . . .	113
6.12	The reconstruction comparison using different models and mask sizes. . . . .	118
6.13	The visualisation of reconstruction, the numbers in masked input indicate visible rates for the model. Detailed analysis of (a)~(l) are in 6.4.2. . . . .	119
6.14	Ripeness score $R$ predictions for complete apple instances, with intervals of 0.1 and at most 40 items displayed per score. . . . .	124
6.15	3D PCA visualisations of ripeness scores on extracted features. . . . .	126
6.16	Two prediction deficiencies in the proposed reconstructor. . . . .	127
6.17	The digital simulation of a large orchard, with apple locations and ripeness monitored. . . . .	128

# List of Tables

2.1	Key aspects and tools for assessing fruit ripening characteristics. . . . .	9
2.2	Fruit ripeness analysis based on sensing methods. . . . .	12
2.3	Machine learning for fruit ripeness determination. . . . .	17
2.4	Common deep learning networks. . . . .	21
2.5	Common deep learning activation functions. . . . .	21
2.6	Common deep learning loss functions. . . . .	22
2.7	Common deep learning optimisers. . . . .	22
2.8	Common deep learning regularisation methods. . . . .	23
2.9	Common deep learning rate schedulers. . . . .	23
2.10	Deep learning for various fruit recognition. . . . .	28
3.1	Specifications of the mobile device used. . . . .	44
3.2	The instance category distribution of NinePeach dataset. . . . .	47
3.3	Statistics of the NinePeach dataset. . . . .	47
3.4	The computer specifications. . . . .	54
3.5	Instance segmentation results on the NinePeach validation set. . . . .	55
3.6	Ablation on different components of the detection head. . . . .	56
3.7	Ablation on different weights for class loss. . . . .	56
3.8	The comparison of training on separate and combined NinePeach dataset. . . . .	57
3.9	The complexity comparison of PeachSOLO and Mask R-CNN. . . . .	59
4.1	Statistics of the StrawDI_Db1 dataset. . . . .	63
4.2	LightStraw backbone architecture variants. . . . .	67
4.3	Instance segmentation results on the StrawDI_Db1 testing set. . . . .	71
4.4	Params and FLOPs of the models. . . . .	71
4.5	The model FPS across different devices. . . . .	73
5.1	Four ripeness stages of strawberry. . . . .	78
5.2	The category distribution of the combined dataset. . . . .	80
5.3	The specification of the proposed backbones. . . . .	83
5.4	Instance segmentation results on the combined dataset. . . . .	88

5.5	Ablation on the pixel decoder. . . . .	91
5.6	Results of ablation experiments based on FruitQuery-xs. . . . .	92
5.7	The AP comparison of training on separate and combined datasets. . . .	93
5.8	The parameters comparison of YOLOv9 and FruitQuery. . . . .	94
5.9	The FPS of FruitQuery across different devices. . . . .	98
6.1	The results of reconstruction. . . . .	116
6.2	The distance results of different extractors. . . . .	121
6.3	The results of predictor using features from extractors. . . . .	123
7.1	The overview of this research. . . . .	131



# Nomenclature

$\sigma(x)$	Sigmoid function
AI	Artificial Intelligence
ANN	Artificial Neural Networks
AP	Average Precision
AR	Average Recall
CAM	Convolutional Activation Map
CBAM	Convolutional Block Attention Module
CCD	Charge-Coupled Devices
CMOS	Complementary Metal-Oxide-Semiconductor
CNN	Convolutional Neural Network
DCN	Deformable Convolution Layer
DL	Deep Learning
DM	Dry Matter
ESMA	Efficient Multi-head Self-Attention
FFN	Feed-Forward Network
FLOPs	Floating-point Operations
FPN	Feature Pyramid Network
FPS	Frames per second
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit

HSI	Hyperspectral imaging
IAM	Instance Activation Map
IoU	Intersection over Union
KNN	K-Nearest Neighbours
LSTM	Long Short-Term Memory
mAP	mean Average Precision
MC	Moisture Content
ML	Machine Learning
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
MSI	Multispectral imaging
NMS	Non-Maximum Suppression
Params	Learnable Parameters
PCA	Principal Component Analysis
PPM	Pyramid Pooling Module
PSNR	Peak-Signal-to-Noise Ratio
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
RPN	Region Proposal Network
SAM	Segment Anything Model
SRSA	Spatial Reduction Self-Attention
SSC	Soluble Solid Content
SSIM	Structural Similarity Index Measure
SVM	Support Vector Machines
SwinT	Swin Transformer

TPU Tensor Processing Unit

TSS Total Soluble Solid

TTA Titratable Acidity

ViT Vision Transformer

VOC Volatile Organic Compounds

WC Water Content

YOLO You Only Look Once



# Chapter 1

## Introduction

### 1.1 Research Motivation

Fruits are an important part of the human diet. Fruit ripeness determination is a critical task in modern agriculture, directly influencing harvest timing and overall fruit quality across both pre- and post-harvest stages. Accurate ripeness assessment not only informs precise fertilisation schedules, crop health monitoring and yield estimation in the pre-harvest phase, but also supports grading, storage, and transportation decisions to minimise spoilage in the post-harvest phase [132]. Traditional ripeness determination methods, such as manual inspection or biochemical analysis, suffer from several limitations, including subjectivity, labour intensity, and poor scalability for large-scale production systems due to the need for specialised and expensive equipment. These challenges are further complicated by the need for non-destructive techniques that preserve fruit integrity while delivering consistent and reliable assessments across different fruit varieties and environmental conditions.

Deep learning, a subset of artificial intelligence (AI) characterised by its ability to automatically learn hierarchical feature representations from raw data, holds great promise to overcome these constraints. By leveraging neural networks and other architectures, deep learning can extract complex patterns directly from images, eliminating the need for handcrafted features and reducing dependence on domain-specific expertise [118].

The motivation of this research is to explore deep learning as a practical, precise, and scalable solution for fruit ripeness determination. Specifically, the solution should enable instance-level fruit recognition under complex, real-world conditions, such as occlusions, lighting variation, and different viewing angles, while maintaining high accuracy and generalisability. Furthermore, it should be lightweight and efficient enough for deployment on in-field platforms, such as picking robots. Finally, the solution should also reduce reliance on manual annotations and provide end-users the flexibility to make decisions based on their own ripeness criteria.

## 1.2 Problem Statement

Although deep learning has significantly advanced computer vision and has been widely adopted across various agricultural applications, its application to fruit ripeness determination remains limited, with several critical challenges yet to be addressed.

First, most existing studies focus on fruit classification or object detection tasks, which can identify the presence and category of fruit but fail to provide precise instance-level localisation or shape information. Only a few studies have applied segmentation methods in fruit-related tasks, and even fewer have explored their use in ripeness estimation. This presents a significant gap, as instance segmentation can capture more detailed spatial information that is critical for assessing subtle ripeness differences.

Second, most datasets in prior studies are relatively small in scale, consisting of only a few hundred images, and are often made private. This restricts reproducibility, comparative evaluation, and broader academic progress. In the rare cases where instance segmentation masks are publicly available, ripeness labels are absent. Moreover, in the limited studies where fruit ripeness is considered, the task is commonly simplified to a binary classification (unripe vs. ripe), which fails to reflect the multi-stage fruit ripening process. This indicates the need for publicly available, high-quality datasets that combine instance-level annotations with multi-category ripeness labels.

Third, segmentation models typically demand more computational resources than classification or detection models due to the higher complexity of the task. However, the need for lightweight models that are suitable for deployment on resource-constrained platforms, such as picking robots or embedded agricultural devices, is often ignored. The development of efficient, lightweight instance segmentation models for fruit ripeness determination, therefore, remains an under-explored area.

Fourth, existing ripeness prediction methods largely depend on manually defined labels, which are based on fixed visual criteria determined by annotators. Although it is an effective and widely-used solution, these labels may not align with the preferences of different end-users across regions, markets, or supply chains. Consequently, models trained on such labels may lack adaptability in wide applications. There is a clear need to explore alternative approaches to reduce reliance on human annotations and allow for a more flexible, user-oriented ripeness assessment.

## 1.3 Research Contributions

This thesis makes several contributions to the development of deep learning methods for fruit ripeness determination. These contributions directly address the key challenges identified in the problem statement and correspond to the research objectives outlined earlier. They are summarised as follows:

### 1. Creation and enhancement of datasets for ripeness determination.

A high-quality peach dataset named NinePeach is created and made publicly available. It contains 4599 images of nine peach cultivars at various ripeness stages, captured under natural field conditions. To the best of the author's knowledge, it is the largest and most diverse peach dataset with instance-level ripeness labels. In addition, a public strawberry instance-level dataset (3100 images) is extended with different ripeness labels to facilitate its application in maturity assessment. These two datasets are further combined to support multi-class ripeness determination under real-world conditions. Together, they serve as valuable benchmarks for future research.

### 2. Proposal of a one-stage CNN instance segmentation model to predict peach ripeness.

A novel one-stage CNN model called PeachSOLO is proposed for peach instance segmentation. Unlike traditional two-stage approaches, PeachSOLO does not need bounding box proposals as prior knowledge. It directly identifies peach instances by their centre locations and sizes, and then predicts their categories at the same time. The model incorporates both channel and spatial attention to improve object detection capabilities in key channels and spatial locations. PeachSOLO outperforms the state-of-the-art Mask R-CNN with 2.21 higher average precision points.

### 3. Development of lightweight Transformer-based fruit segmentation models for embedded devices.

Two lightweight models are developed to address the computational demands of segmentation tasks. The first, named LightStraw, is an efficient CNN-based model for strawberry instance segmentation. It adopts efficient self-attention for a lightweight backbone to extract semantic features. The CoordConv and Instance Activation Maps are introduced to add position and instance-aware weighted maps to the model. LightStraw demonstrate efficiency by requiring much fewer parameters (17.42M) and floating-point operations (78.3G) compared to Mask R-CNN (35.08M / 877.4G), making them suitable for deployment on embedded devices. The second, named FruitQuery, is a query-based end-to-end segmentation framework that combines convolutional features with Transformer decoders. Trained on the combined peach-strawberry dataset, it supports multi-stage ripeness classification while maintaining a small model size. FruitQuery achieves the highest average precision of 67.02 with only 14.08M parameters, outperforming 13 state-of-the-art models with 33 variants. Both models achieve competitive accuracy with significantly reduced parameters and computational cost, making them suitable for use on resource-constrained or embedded devices in agricultural environments.

#### 4. Introduction of a self-supervised learning method to predict ripeness for occluded apples with few labels.

A novel self-supervised framework, named AppleSSL, is introduced to estimate apple ripeness in scenarios where fruits are partially hidden by leaves or branches. The method applies contrastive learning and image reconstruction tasks to learn from a large number of unlabelled data, requiring only a small fraction of labelled images. It generates a continuous ripeness score instead of discrete pre-defined categories, offering greater flexibility for end-users to make their decisions. AppleSSL achieves the best Structural Similarity Index Measure of 0.75 and the second-best Peak-Signal-to-Noise Ratio of 25.36 for reconstructing missing apple parts, whilst using the fewest 86.3M parameters. It outperforms 15 other self-supervised methods and a supervised method in the ripeness score prediction, with the smallest score 0.0127 for fully unripe and the highest score 0.8933 for fully ripe apples.

## 1.4 Publications

1. **Zhao, Z.**, Hicks, Y., Sun, X., & Luo, C. (2023). Peach Ripeness Classification based on a New One-stage Instance Segmentation Model. *Computers and Electronics in Agriculture*, 214:108369. [10.1016/j.compag.2023.108369](https://doi.org/10.1016/j.compag.2023.108369)
  - This work corresponds to Chapter 3.
  - Contributions: **Zhao, Z.**: Methodology, Software, Validation, Visualization, Writing, Data curation. **Hicks, Y.**: Methodology, Writing, Supervision. **Sun, X.**: Methodology, Writing, Supervision. **Luo, C.**: Resources, Writing.
2. **Zhao, Z.**, Hicks, Y., Sun, X. (2024). Faster Segmentation Models for Peach Ripeness Determination. In *Proceedings of the Cardiff University School of Engineering Research Conference 2024*, pages 33–37. [10.18573/conf3.i](https://doi.org/10.18573/conf3.i)
  - This work corresponds to Chapters 3 and 4.
  - Contributions: **Zhao, Z.**: Methodology, Software, Validation, Visualisation, Writing, Data curation. **Hicks, Y.**: Methodology, Writing, Supervision. **Sun, X.**: Methodology, Writing, Supervision.
3. **Zhao, Z.**, Hicks, Y., Sun, X., McGuinness, B. J., & Lim, H. S. (2024). Lightweight and Efficient Attention-based CNN Models for In-field Strawberry Instance Segmentation. *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*, pages 3294–3299, ISSN:2161-8089. [10.1109/CASE59546.2024.10711802](https://doi.org/10.1109/CASE59546.2024.10711802)
  - This work corresponds to Chapter 4.



- Contributions: **Zhao, Z.:** Methodology, Software, Validation, Visualisation, Writing, Data curation. **Hicks, Y.:** Methodology, Writing, Supervision. **Sun, X.:** Methodology, Writing, Supervision. **McGuinness, B. J.:** Methodology, Writing. **Lim, H. S.:** Methodology, Writing.
4. **Zhao, Z.,** Hicks, Y., Sun, X., & Luo, C. (2025). FruitQuery: Lightweight Query-based Instance Segmentation Models for In-field Fruit Ripeness Determination. *Smart Agricultural Technology*, 12:101068. [10.1016/j.atech.2025.101068](https://doi.org/10.1016/j.atech.2025.101068)
- This work corresponds to Chapter 5.
  - Contributions: **Zhao, Z.:** Methodology, Software, Validation, Visualisation, Writing, Data curation. **Hicks, Y.:** Methodology, Writing, Supervision. **Sun, X.:** Methodology, Writing, Supervision. **Luo, C.:** Resources, Writing.
5. **Zhao, Z.,** Hicks, Y., Sun, X., McGuinness, B. J., & Lim, H. S. (2025). A Novel Self-supervised Method for In-field Occluded Apple Ripeness Determination. *Computers and Electronics in Agriculture*, 234:110246. [10.1016/j.compag.2025.110246](https://doi.org/10.1016/j.compag.2025.110246)
- This work corresponds to Chapter 6.
  - Contributions: **Zhao, Z.:** Methodology, Software, Validation, Visualisation, Writing, Data curation. **Hicks, Y.:** Methodology, Writing, Supervision. **Sun, X.:** Methodology, Writing, Supervision. **McGuinness, B. J.:** Methodology, Resources, Writing. **Lim, H. S.:** Methodology, Resources, Writing.

## 1.5 Thesis Outline

This thesis is divided into several chapters corresponding to the above main research questions.

### Chapter 1 – Introduction

This chapter introduces the background and motivation of the research, followed by a problem statement. It outlines the research objectives, summarises the key research contributions, lists relevant publications, and concludes with the structure of the thesis.

### Chapter 2 – Literature Review

This chapter reviews existing methods for fruit ripeness assessment. It begins with the biological background of fruit ripening, followed by traditional approaches such as colour inspection, spectral methods, and aroma analysis. It then discusses the role of machine learning and deep learning, including self-supervised learning in fruit ripeness determination, and highlights the current research gaps.

**Chapter 3 – PeachSOLO**

The chapter introduces the NinePeach dataset, a large, high-quality collection capturing real-world conditions like varying light, fruit adhesion, and occlusion. This chapter also presents PeachSOLO, a one-stage instance segmentation model for peach ripeness classification. It integrates channel attention and spatial attention to enhance feature representation, enabling accurate detection under complex contexts.

**Chapter 4 - LightStraw**

This chapter proposes lightweight, efficient CNN models for strawberry instance segmentation. The models integrate efficient self-attention and novel components to enhance positional and instance-level understanding. Their accuracy and computational efficiency are validated through extensive experiments.

**Chapter 5 – FruitQuery**

This chapter combines peach and strawberry datasets and introduces FruitQuery, a lightweight, end-to-end query-based instance segmentation model for multi-stage ripeness determination. The model combines convolutional features with Transformer decoders. Experimental comparisons show that the model outperforms 13 other models with 33 variants, highlighting its competitive accuracy and low computational cost.

**Chapter 6 - AppleSSL**

This chapter presents a self-supervised framework, AppleSSL, with a reconstructor, feature extractor, and predictor for in-field occluded apple ripeness. Trained on a limited number of labelled data ( $<1\%$ ), it reconstructs missing parts and predicts continuous ripeness scores, outperforming 15 self-supervised and one supervised methods. It demonstrates good performance under occlusion and shows the ability to cover different ripeness criteria from different users.

**Chapter 7 - Conclusions and Future Work**

The final chapter summarises the main findings and contributions of the thesis. It also discusses the limitations of the current work and suggests potential directions for future research.

# Chapter 2

## Literature Review

### 2.1 Background of Fruit Ripeness

#### 2.1.1 Introduction

Fruits are an essential part of the human diet, providing vital nutrients such as vitamins, minerals, and antioxidants that contribute to overall health and well-being. They play a foundational role in supporting growth in children and sustaining physiological functions across all ages [249]. Regular fruit consumption has been shown to help prevent various forms of malnutrition, including under-nutrition and obesity, and reduce the risk of non-communicable diseases [67]. Given their nutritional significance, ensuring that fruits reach consumers in an optimal state is necessary for maximising health benefits and maintaining market value.

Food waste remains a significant global challenge, especially in the context of a growing global population. Food and Agriculture Organisation of the United Nations (FAO) estimated that one-third of all global food production is lost or wasted, with fruits and vegetables being among the most wasted food categories [31]. This not only causes severe environmental damage but also exacerbates food insecurity.

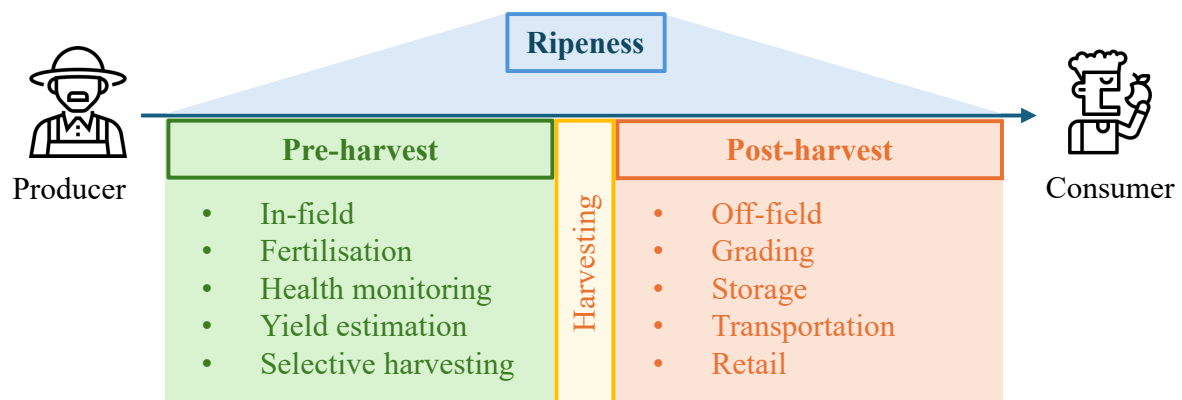


Figure 2.1: Ripeness is a key factor throughout the whole process of fruit production.

A major factor contributing to this wastage is improper harvesting and storage practices, which result in substantial losses along the supply chain. World Wide Fund for Nature UK (WWF-UK) reveals that up to 14% of total food production is lost post-harvest based on total harvest weight, including at retail stages, with 8.3% wasted at or around harvest and 7.0% during farm-stage post-harvest activities [290]. It is estimated that the food lost on farms alone is enough to feed the world's undernourished population almost four times over [289].

Building on the issue of fruit waste, one of the key factors influencing losses is ripeness. The ripeness of fruit plays a crucial role throughout its entire production lifecycle from pre-harvest to post-harvest, directly affecting harvest timing and consumer satisfaction, as shown in Fig. 2.1.

In the pre-harvest stage, fruit ripeness is considered by producers to schedule precise fertilisation, monitor crop health and estimate the yield. It also helps the development of selective harvesting, one practice of precision agriculture, ensuring that fruits are harvested only when grown with the desired flavour, texture, and nutritional profile [208].

In the post-harvest stage, ripeness continues to play a vital role before the fruits reach consumers. Activities such as grading, storage and transportation have to take ripeness into account, as overripe fruits are prone to spoilage during handling and moving, while under-ripe fruits may fail to meet consumer expectations in terms of taste and appearance. Effective ripeness management becomes critical to preserve product quality and maximise value [132].

### 2.1.2 Biology of Fruit Ripening

Fruit ripening is a highly coordinated, genetically programmed, and irreversible natural process involving a series of physiological, biochemical, and organoleptic changes. In most cases, a green, hard and immature fruit becomes more colourful, softer, sweeter, and aromatic [81]. For example, a green apple may turn red or yellow, while its flesh softens and gains sweetness, making it ready to eat. These changes matter in farming because they affect when fruits are picked and how good they taste for consumers.

The phenotypic changes during fruit ripening are complex and varied. Numerous physical and chemical attributes can be quantified during ripening. These include size, shape, texture, firmness, external colour, internal colour, the concentration of chlorophyll, starch, sugars, Soluble Solids Contents (SSC), Total Soluble Solids (TSS), oils, and internal ethylene concentration [241]. Ethylene, for instance, acts like a signal to start softening and colour shifts in fruits such as bananas or tomatoes. These traits help farmers know the best time to harvest and ensure fruits meet market needs. The main aspects of fruit ripening and relevant tools are summarised in Table 2.1.

Table 2.1: Key aspects and tools for assessing fruit ripening characteristics.

Aspect	Description	Tools
Colour [192]	Ripening fruits change colour due to the breakdown of chlorophyll (green) and the synthesis of carotenoids (yellow/orange) and anthocyanins (red/purple).	Imaging, Spectroscopy
Volatiles [295]	Ripening fruits produce aromas through the metabolic process. Compounds such as esters, alcohols, and aldehydes are closely linked to the fruit's flavour.	GC-MS <sup>1</sup> , E-Nose <sup>2</sup>
Texture (Firmness) [268]	Enzymatic, such as pectin methylesterase breakdown of cell wall components such as pectin, cellulose, and hemicellulose, softening of the fruit flesh.	Fluorescence
Sugar Content [254]	Hydrolysis of starch into simpler sugars like glucose, fructose, and sucrose. This process is driven by enzymes, which contribute to the increase in sweetness.	Spectroscopy
Acidity [175]	Organic acids such as malic acid and citric acid are metabolised through respiration or converted into sugars, reducing the fruit's sourness.	Titration, pH

<sup>1</sup> GC-MS: Gas Chromatography-Mass Spectrometry<sup>2</sup> E-Nose: Electronic Nose

## 2.2 Sensing Methods for Ripeness Determination

Based on the biology of fruit ripening, extensive research has been conducted on developing methods to determine fruit ripeness over recent decades. Most sensing methods rely on non-destructive techniques, such as colourimetry, spectral imaging and spectroscopy. Some of these methods are shown in Fig. 2.2, and a comprehensive summary is provided in Table 2.2.

### 2.2.1 Colour Inspection

Colourimeters are analytical instruments designed to quantify the colour of the fruit by measuring its absorbance or reflectance of specific wavelengths of light. They are more accurate than human visual assessment using CIELAB colour space. This method has been applied to various fruits, including apples [69], bananas [193], mangoes [125], tomatoes [64], and peaches [70].

Visible imaging, by contrast, refers to the use of digital cameras to capture a two-dimensional image in RGB colour space. It employs technologies such as charge-coupled devices (CCD) or complementary metal-oxide-semiconductor (CMOS) sensors to detect red, green, and blue wavelengths. For example, an approach was proposed to assess fruit ripeness by extracting mean RGB values from four directional images of a fruit [55].

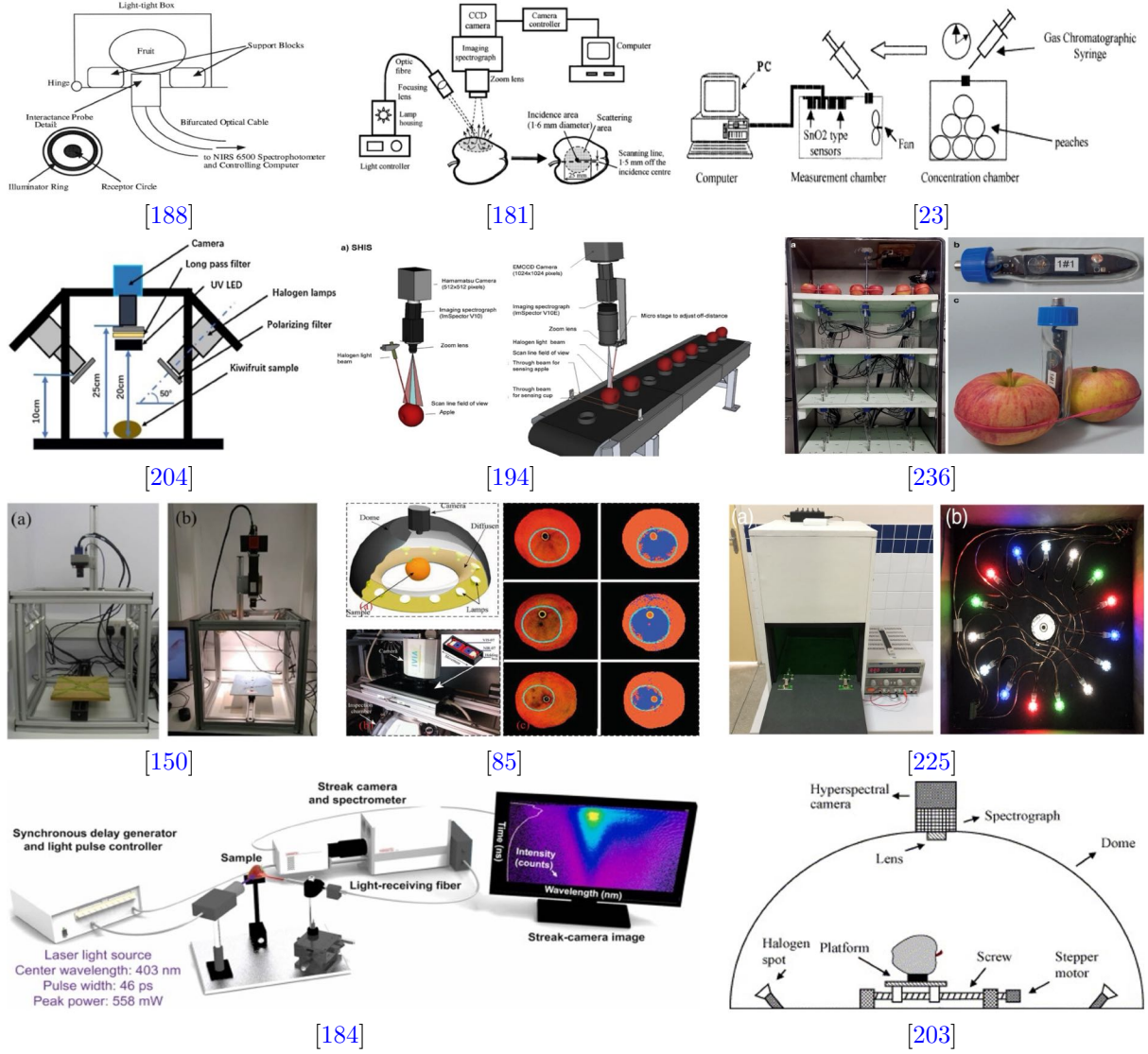


Figure 2.2: Some sensing methods used for determining fruit ripeness.

However, colourimeters require controlled lighting conditions and calibration, making them less feasible for rapid, in-field assessments. Similarly, visible imaging struggles with inconsistent results due to variations in illumination and fruit orientation, often demanding extensive manual processing.

## 2.2.2 Spectral Imaging

Hyperspectral imaging (HSI) and multispectral imaging (MSI) represent two widely adopted methodologies within spectral imaging.

HSI generates a three-dimensional imaging cube with images at a range of narrow, contiguous spectral bands. A single spectrum can be extracted from each pixel, representing the absorption properties and the textural information of fruit samples. For instance, [75] collected HSI data on strawberries at early ripe and ripe stages, covering a wavelength range of 370 to 1015 nm, with specific wavelengths of 530 nm and 604 nm

selected for field samples, and 528 nm and 715 nm for laboratory samples.

MSI is similar but different from HSI, as it acquires a limited number of predefined, discrete spectral bands, rather than capturing hundreds of spectral bands as in HSI. These bands are typically wider and non-contiguous, designed to target particular spectral features of interest, making MSI more computationally efficient and suitable for in-field applications. For instance, [181] employed MSI with five wavelengths to correlate scattering profiles with the firmness and SSC of apples, achieving a high correlation for both quality attributes.

However, HSI requires significant computational resources and sophisticated equipment, often limiting its use to laboratory settings and reducing real-time applicability. MSI sacrifices spectral resolution for efficiency, potentially missing subtle ripeness indicators, and requires careful band selection, which may not generalise across fruit types.

### 2.2.3 Spectroscopy

Spectroscopy is a fundamental analytical technique that examines the interaction of electromagnetic radiation with matter, widely employed in fruit quality assessment to evaluate physicochemical properties. The sensing principle involves illuminating a sample and capturing its spectral response, which reflects its chemical composition and structural characteristics. Unlike HSI, spectroscopy typically acquires spectral data from a single point or a small area, without spatial information. Near-infrared spectroscopy stands out as the most prevalent method in this domain, offering rapid and non-destructive analysis of fruit samples. For example, [32] highlighted its extensive application in determining fruit quality attributes, such as ripeness and composition. In a specific study, [109] developed an automated classification procedure for peaches, utilising multispectral imaging to compute ratio images (red at 680 nm divided by infrared at 800 nm), effectively distinguishing ripeness stages. Similarly, [201] investigated visible-near-infrared (450–1040 nm) hyperspectral reflectance imaging to assess the internal properties and sensory attributes of two varieties of nectarines, demonstrating its efficacy in quality monitoring.

However, spectroscopy-based methods often require expensive equipment and careful calibration, which can restrict their use to controlled laboratory environments. Furthermore, their accuracy may vary across different fruit species.

### 2.2.4 Fluorescence

Fluorescence is an optical technique that leverages the emission of light from a sample following excitation by specific wavelengths, offering a non-invasive approach to assess fruit ripeness and quality parameters. This method relies on the detection of fluorescence signals, often linked to compounds such as chlorophyll, which diminish as fruits ripen. For example, [21] developed a fluorescence imaging system that measured emissions at



Table 2.2: Fruit ripeness analysis based on sensing methods.

Fruit	Colourimetry	Spectroscopy	Fluorescence	Visible Imaging	Hyperspectral Imaging	Multispectral Imaging	GC-MS/ E-Nose
Apple	Colour [69]	Chl [325], SSC [213], Firmness [326], Flavonols [250], Anths+Cars [196]	Chl [327], SSC [194], Anths [19], Flavonols [19], Firmness [215]	Colour [55]	SSC [194], Firmness [215],	SSC [214], Firmness [298]	VOC [117]
		SSC+Firmness +TTA [25]					VOC [86]
Avocado		MC [207], DM [207]			DM [82], Firmness [270]		
Banana	Colour [193]	Chl [1], TSS [121]		Colour [193]	Firmness+ TSS [224]		VOC [22]
Cherry	Colour [53]	SSC [27], Firmness [180]					
Grape		Anths [123], SSC+TTA [284]	Chl+Anths +TSS [2], Flavonols [149]		SSC+TTA [7]		VOC [68]
Kiwifruit		DM [188], Firmness [147], TSS+SSC+ Starch [248]	Colour [204]		SSC+Firmness [18]		VOC [76]
		Colour [126], Starch [234], Firmness [238], DM+SSC [255]	Chl [145]		Firmness+ SSC+WC [252], DM [99]		
Mango	Colour [125]						

Abbreviations: Chl (Chlorophyll), Anths (Anthocyanins), Cars (Carotenoids), SSC (Soluble solid content), DM (Dry matter), MC (Moisture content), TTA (Titratable acidity), TSS (Total soluble solid), VOC (Volatile organic compounds), WC (Water content)



Table 2.2 (Continued): Fruit ripeness analysis based on sensing methods.

Fruit	Colorimetry	Spectroscopy	Fluorescence	Visible Imaging	Hyperspectral Imaging	Multispectral Imaging	GC-MS/E-Nose
Mandarin		TTA [189], Firmness [83], DM [98], SSC [88]	Chl [200]		SSC [183]		VOC [84]
Melon		SSC [176]					VOC [220]
Nectarine	Colour [182]	SSC [218], Firmness [218]	Firmness [21]		Firmness+SSC+ TA+Colour [201]		VOC [279]
Peach	Colour [70]	Chl [324], Colour [70], Firmness [21]	Firmness [181]		Firmness [174]		VOC [63]
Pear		SSC [243], Firmness [30]	Chl [187]		SSC [112]		
Persimmon		SSC [124]			Firmness [284]		
Plum		SSC [211], Firmness+Colour [119]	Chl [244]		SSC+Colour [150]		
Pineapple	Colour [8]	DM [97], SSC [43]					VOC [283]
Strawberry	TSS+TTA [3], Colour+ Firmness [259]		Firmness [166], TSS+TTA [65]		SSC [168], Firmness [261]		VOC [223]
Tomato	Colour [64], Firmness [16], TSS [232]	SSC [47], Lycopene [48]	Chl [111]	Colour [77], Firmness [239]		Phenolic+ Lycopene [165]	

690–740 nm, excited by UV-blue and red light, achieving strong correlations between fluorescence signals and quality attributes such as firmness and SSC in fresh apples, despite minimal changes in skin hue. However, the same study noted weaker correlations for peaches and nectarines, suggesting variability across fruit types. Notably, their system did not require dark adaptation of samples, though it was designed exclusively for laboratory use, highlighting limitations in its practical application. These findings underscore fluorescence as a promising tool for ripeness determination, with its efficacy dependent on fruit species and experimental conditions.

However, fluorescence relies on specialised equipment that is often confined to laboratory settings, making it less suitable for field use. The technique’s sensitivity to fruit-specific responses and the need for controlled conditions further limit its scalability and real-time applicability.

### 2.2.5 Aroma

Aroma analysis is a key method for evaluating fruit ripeness and quality by detecting Volatile Organic Compounds (VOC) using techniques such as Gas Chromatography-Mass Spectrometry (GC-MS) and Electronic Nose (E-Nose) systems. These methods provide valuable insights into the aromatic profiles that define fruit ripeness. For instance, the E-Nose has been applied to monitor mandarin ripeness across different harvest dates, linking VOC to quality parameters such as firmness, SSC, and acidity [84]. Similarly, GC-MS has been utilised to characterise the aroma of bananas, identifying and observing the evolution of critical VOCs during ripening and drying processes, where some compounds diminish while others remain stable under varying temperature conditions [22]. Additionally, in kiwifruit, GC-MS analysis of bound volatiles revealed fluctuations in VOC across under-ripe, ripe, and over-ripe stages, although these compounds showed limited influence on the characteristic aroma of ripe fruit [76].

However, aroma-based methods often require costly, sophisticated equipment and controlled laboratory environments, restricting their use outside specialised settings. Moreover, the complexity of interpreting VOC data and the variability in compound detection across fruit species can reduce their consistency and practicality for large-scale applications.

### 2.2.6 Summary

The sensing methods mentioned above provide detailed insights for fruit ripeness determination but come with notable limitations. Many of these techniques rely on expensive, specialised equipment that can be cost-prohibitive for widespread adoption. They often require controlled laboratory conditions, making them impractical for on-site or field use. Additionally, the complexity of data processing and the need for precise calibration can

slow down analysis, while variability across fruit types reduces their general applicability. These constraints, high costs, lack of portability, and challenges in efficient implementation limit their scalability and accessibility.

## 2.3 Machine Learning for Ripeness Determination

### 2.3.1 Introduction

To overcome the limitations of sensing ripeness determination methods, researchers have turned to machine learning (ML) techniques as an alternative. Machine learning enables automated, non-destructive ripeness prediction by learning patterns from data, often using visual features extracted from fruit images or simple sensor data.

This chapter introduces the foundations of machine learning in the context of fruit ripeness estimation, outlines common model structures and workflows, and reviews their typical applications. ML methods utilise knowledge-based insights and pattern recognition capabilities to enhance the accuracy of fruit ripeness determination. Unlike sensing methods that rely on biochemical measurements, these methods enable non-destructive identification and classification of fruit ripeness stages, often taking fruit images as inputs.

### 2.3.2 Concepts and Algorithms

Machine learning is a data-driven approach that builds models capable of learning from examples. These models use training data to identify patterns and relationships, which are then used to make predictions or classifications on new, unseen data [118].

A key feature of traditional ML methods is their reliance on feature engineering, which is the manual extraction of meaningful attributes from raw data. These features, such as colour histograms or texture descriptors, are often designed with domain expertise and significantly influence model performance.

Common machine learning algorithms can be broadly divided into two categories: supervised and unsupervised learning.

On the one hand, supervised learning algorithms are trained on labelled data, where each input is paired with a known output. These models learn to map inputs to outputs and are widely used for classifying object categories or predicting scores. Typical examples include Linear Regression [60], which models relationships between features, and Support Vector Machines (SVM, 51), which are effective for binary or multi-class classification problems. K-Nearest Neighbours (KNN, 52) predicts the output for a test sample by identifying the K closest samples in the training data based on a distance metric, while Artificial Neural Networks (ANN, 101) can capture more complex, non-linear relationships, though they require more data and tuning. Naïve Bayes (NB, 282) is based

on Bayes' theorem, which assumes independence between features and is commonly used for classification tasks.

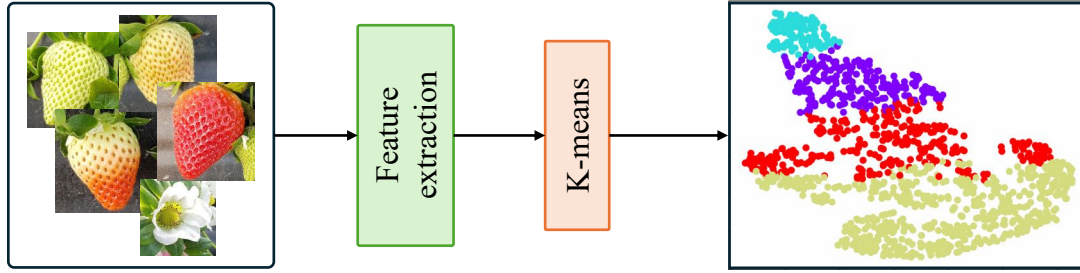


Figure 2.3: The example of using K-means to cluster fruits.

On the other hand, unsupervised learning methods work with unlabelled data, aiming to uncover hidden structures or groupings. These are especially useful when labelled training samples are unavailable. K-means clustering [186], for example, groups fruit samples based on similarities in features like colour or texture, as shown in Fig. 2.3. It is simple and widely used for preliminary data exploration. More advanced methods like Gustafson–Kessel clustering [96] allow clusters to take on elliptical shapes, offering more flexibility when object features are not evenly distributed.

Both types of algorithms provide different strengths depending on the availability of data. However, their performance often relies on manually designed features, such as colour histograms or texture descriptors, which may not fully capture the complex visual patterns.

### 2.3.3 Applications

Machine learning has a wide range of applications, from email spam filtering and fraud detection in finance to personalised recommendation systems, permeating many aspects of daily life. For example, Logistic Regression was used to predict football match results [222], and NB was applied to check whether social media contents express hatred for people [136].

ML methods have also been applied to determine fruit ripeness, as summarised in Table 2.3. Supervised learning algorithms such as SVM, KNN, NB, ANN, and various regression models are the most frequently used. These methods are well-suited for classification and regression tasks where labelled data, such as predefined ripeness stages, is available. They have been applied to fruits like avocado [44], blueberry [154], mango [287], papaya [17], and tomato [64], often using features such as colour, texture, or spectral signatures extracted from images or sensors.

Unsupervised techniques like K-means and Gustafson–Kessel clustering are commonly used when label information is unavailable or difficult to obtain. These methods group fruits based on similarities in visual or spectral features and have been employed for

Table 2.3: Machine learning for fruit ripeness determination.

Fruit	Machine Learning Methods	References
Apple	Random Frog	2020, [311]
Avocado	SVM, KNN, Ridge Regression, Lasso Regression	2020, [44]
Banana	K-means, Gustafson–Kessel	2012, [55]
Blueberries	KNN, NB, Supervised K-means Clustering	2014, [154]
Grape	KNN, SAE, Logistic Regression, Decision Tree	2018, [240]
Mango	K-means, NB, SVM	2022, [287]
Oil palm fruit	Fast Fuzzy C-means	2024, [228]
Papaya	KNN, SVM, NB	2021, [17]
Peach	ANN, LDA, Random Forest	2022, [173]
Pear	SAE	2018, [303]
Tomato	SVM, LDA	2015, [64]

Abbreviations: LDA (Linear Discriminant Analysis), KNN (K-Nearest Neighbours), NB (Naïve Bayesian), SVM (Support Vector Machines), SAE (Stacked Auto-Encoders), ANN (Artificial Neural Network)

bananas [55] and mangoes [287], among others. [311] also incorporates a hybrid approach that combines supervised classifiers with feature selection techniques, to reduce dimensionality and predict apple ripeness.

Across all applications, these models demonstrate flexibility in handling different data types and fruit species. However, they typically rely on manually designed features and domain-specific knowledge, which may limit their scalability and performance when applied to more complex or diverse datasets.

### 2.3.4 Summary

Machine learning offers a flexible and efficient alternative to sensing ripeness determination methods. Its ability to work with visual and sensor data makes it suitable for real-time, non-destructive ripeness assessment in both laboratory and field conditions.

Though ML models are generally lighter and more interpretable, they rely more heavily on manually engineered features and may struggle with complex, high-dimensional data. These challenges have led to the growing use of deep learning methods (discussed in the next section), which aim to automatically learn more robust features directly from raw data.

## 2.4 Deep Learning for Ripeness Determination

### 2.4.1 Introduction

Deep learning (DL) is a specialised subset of ML that focuses on building models using artificial neural networks with multiple layers. While both are subfields of artificial intelligence (AI), deep learning distinguishes itself by its ability to automatically extract features and learn hierarchical representations from raw data, often achieving superior performance in complex tasks. The relation between DL, ML and AI is shown in Fig. 2.4.

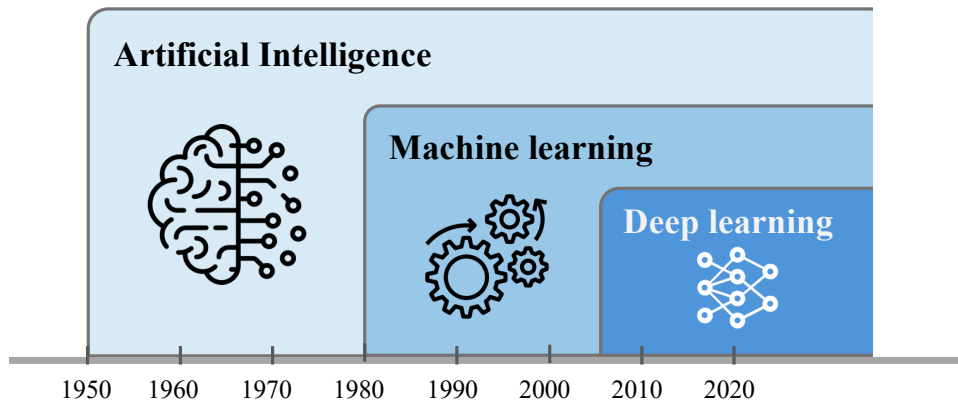


Figure 2.4: The relation between deep learning, machine learning and artificial intelligence.

Inspired by the structure and function of biological neural networks, deep learning models focus on developing systems capable of simulating humans’ cognitive functions like learning and decision-making. It is also called a deep neural network or deep neural learning [118]. A biological neural network is illustrated in Fig. 2.5. This inspiration has led to the development of DL, which is capable of solving problems in vision and language.

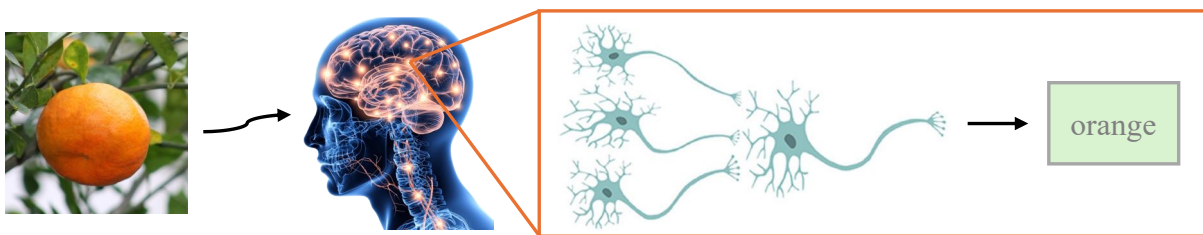


Figure 2.5: A biological neural network.

Deep learning distinguishes itself by automatically learning features from data through multi-layer neural networks, eliminating the need for extensive manual feature engineering. The term “deep” is because the structure of artificial neural networks consists of input, output, and multiple hidden layers [118].

Each layer contains units that transform the input data into features that the next layer can use for a specific predictive task. Features extracted by each layer progress from

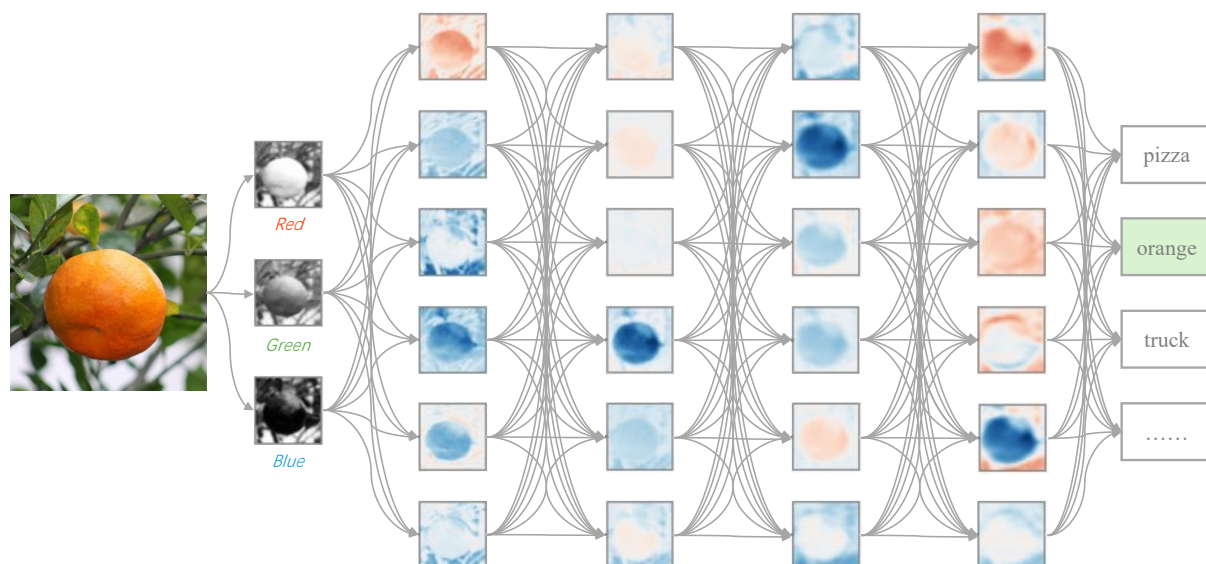


Figure 2.6: A deep learning network for image classification.

simple to complex. For instance, in image processing tasks, initial layers may capture edges and textures, intermediate layers learn structural patterns, and final layers grasp semantic information [20]. Thanks to this structure, deep learning can automatically extract features from raw data, learn hierarchical representations, and analyse complex patterns. A simple example of deep learning is shown in Fig. 2.6.

Deep learning does well with large datasets and powerful computational resources, using backpropagation algorithms to optimise large model parameters for high-dimensional, nonlinear data. It has been successful in various tasks, such as image recognition, natural language processing, and speech recognition [118].

### 2.4.2 Deep Learning Structure

Fig. 2.7 shows a typical structure of a deep learning model. A deep learning model usually consists of five essential components: a network (an input layer, hidden layers, and an output layer), a loss function, regularisation, a learning rate scheduler and an optimiser [118]. Each of them plays a distinct role in processing data and making predictions.

#### Network

A typical network comprises an input layer, hidden layers and an output layer. Layers are the fundamental building blocks of deep learning models, determining how data is processed and transformed as it moves through the network [118].

First, the input layer serves as the starting point of the model, which takes raw data like images and processes it into vectors or tensors. Its structure is determined by the number of features in the input data, such as pixel values. Second, the hidden layers form the core part of the network. They consist of multiple layers that progressively transform



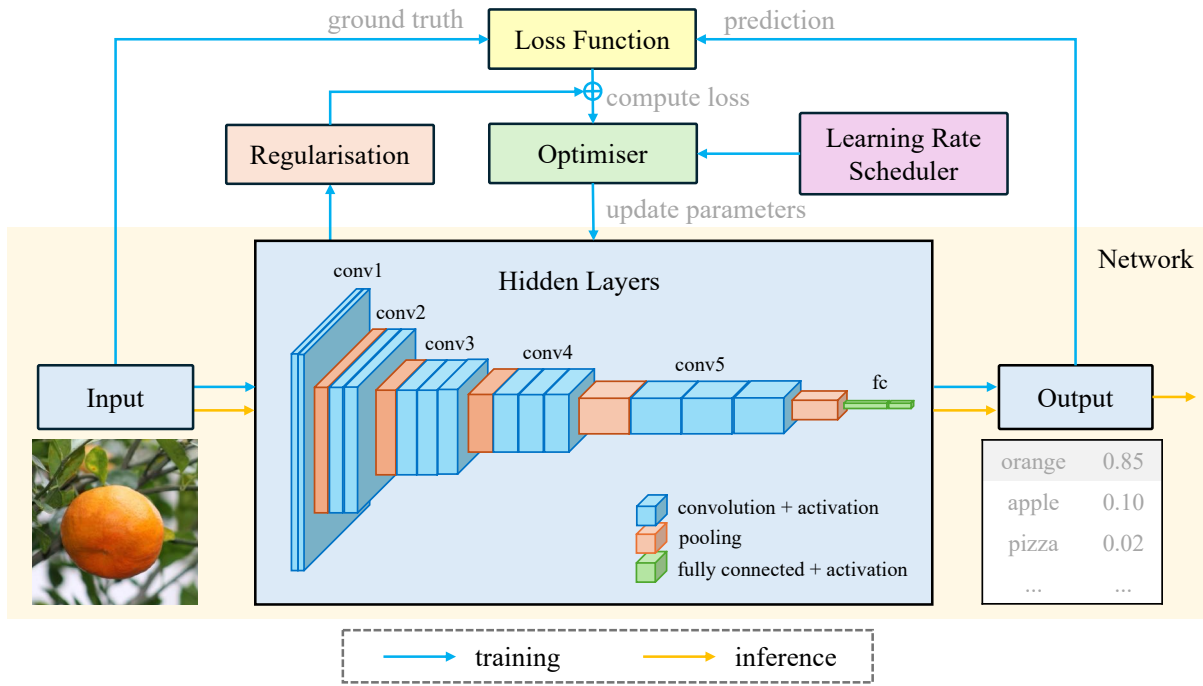


Figure 2.7: The typical structure of a deep learning model.

and extract features from the input. These layers learn hierarchical representations, with earlier layers capturing basic patterns (e.g., edges in images) and deeper layers identifying more complex structures or semantics [102]. Third, the output layer generates the final prediction based on the processed information. For example, in classification tasks, the output is the probabilities for each class, while in regression tasks, the output is continuous values.

Table 2.4 summarises some common deep learning networks for different tasks. This thesis mainly focuses on computer vision tasks.

### Activation Function

Activation functions are crucial components of deep learning models as they introduce nonlinearity into the network, enabling it to learn and represent complex patterns in data. Without activation functions, the entire model would behave like a linear function, regardless of the number of layers, limiting its ability to solve non-linear problems [118].

Some common activation functions are shown in Table 2.5. For example, Rectified Linear Unit (ReLU) is a popular activation function that helps address the vanishing gradient problem.

### Loss Function

The loss function measures the discrepancy between the predicted outputs and ground truths, providing feedback for optimisation. This measurement is used to compute gra-

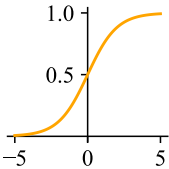
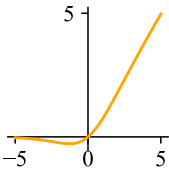
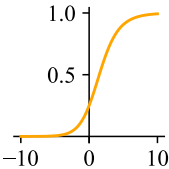
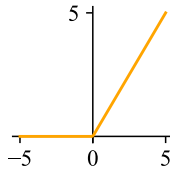
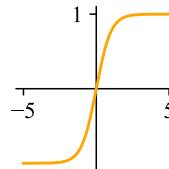


Table 2.4: Common deep learning networks.

Network	Description	Task
CNN [146]	Uses convolutional layers to capture spatial hierarchies in images.	Computer Vision
RNN [237]	Processes sequential data by maintaining a memory of previous time steps.	Time Series Forecasting
Transformer [271]	Uses attention mechanism to capture relationships between input elements.	Natural Language Processing
ViT [59]	Applies transformer architecture to the image by processing the image into patches and sequences.	Computer Vision
GAN [87]	Generates realistic data by training two networks in competition: a generator and a discriminator.	Generation
Autoencoder [9]	Learns to compress data into a lower-dimensional representation and then reconstruct it.	Compression

Abbreviations: CNN (Convolutional Neural Network), RNN (Recurrent Neural Network), ViT (Vision Transformer), GAN (Generative Adversarial Network)

Table 2.5: Common deep learning activation functions.

Sigmoid	Swish	Softmax	ReLU	Tanh
				
$\sigma(x) = \frac{1}{1 + e^{-x}}$	$f(x) = \frac{x}{1 + e^{-x}}$	$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$	$f(x) = \max(0, x)$	$f(x) = \frac{2}{1 + e^{-2x}}$

dients through backpropagation, which quantifies how each model parameter contributes to the error [118]. Effective loss functions not only reflect the problem’s requirements but also impact the model’s convergence rate and overall performance. A well-chosen loss function ensures that the optimisation process aligns with the intended goals, improving the model’s ability to generalise [20].

Some common loss functions are listed in Table 2.6. For example, cross-entropy loss is often used for classification, while mean squared error is suitable for regression tasks.

Optimiser

Optimisers are algorithms that are designed to adjust the network’s parameters to minimise the loss function. Optimisers calculate updates to model weights based on gradients

Table 2.6: Common deep learning loss functions.

Loss	Formula*	Task
Mean Squared Error (MSE) Loss [118]	$\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$	Regression
Cross-Entropy (CE) Loss [20]	$-\frac{1}{N} \sum_{i=1}^N \sum_{c=1}^C y_{i,c} \log(\hat{y}_{i,c})$	Classification
Dice Loss [256]	$1 - \frac{2 \sum y_i \hat{y}_i}{\sum y_i + \sum \hat{y}_i}$	Segmentation

\*  $N$  is the batch size,  $y$  is the label,  $\hat{y}$  is the prediction,  $C$  is the number of classes

computed via backpropagation, often incorporating techniques to accelerate convergence and avoid local minima [118]. The choice of optimiser is important as it directly influences model performance, convergence speed, and stability [230].

Some common optimisers are detailed in Table 2.7. For instance, the widely used optimiser Stochastic Gradient Descent (SGD) provides a simple and effective update rule, and the Adam optimiser can adjust learning rates based on momentum and variance of gradients.

Table 2.7: Common deep learning optimisers.

Optimiser	Description	Update Rule*
SGD [227]	Updates weights using a small batch of data.	$\theta_{t+1} = \theta_t - \eta \nabla_{\theta} J(\theta_t)$
Momentum [219]	Builds on SGD by adding a weighted average of past gradients.	$\theta_{t+1} = \theta_t - \eta \nabla_{\theta} J(\theta_t) + \beta v_t$ , where $v_t$ is the accumulated gradient and $\beta$ is the momentum term.
Adam [138]	Uses Momentum and variance of gradients to adjust learning rates for each parameter.	$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}}$ , where $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$ and $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$ are estimates of the gradient's first and second moments.
AdamW [178]	An improved version of Adam that decouples weight decay from the gradient-based update.	$\theta_{t+1} = \theta_t - \eta \left( \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} + \lambda \theta_t \right)$ , where $\lambda$ is the weight decay coefficient, $\hat{m}_t$ and $\hat{v}_t$ are defined as above.

\*  $\theta$  is the weight,  $t$  is the time step,  $\eta$  is the learning rate,  $J$  is the loss function,  $\nabla$  is the gradient

## Regularisation

Regularisation techniques are employed to prevent overfitting, ensuring that the model generalises well to unseen data. Overfitting occurs when a model becomes excessively complex and learns to memorise the training data rather than capture the underlying patterns. Regularisation introduces penalties or constraints during training, discouraging overly complex models that might overfit [118].

Some common regularisation methods are detailed in Table 2.7. For example, L1 and L2 regularisation add penalties to the loss function based on the magnitude of the model's weights, thereby encouraging simpler models with smaller weights.

Table 2.8: Common deep learning regularisation methods.

Regularisation	Description	Type
L1 [100] Regularisation	Adds a penalty of $\lambda \sum  \theta $ to the loss function, where $\lambda$ is a hyperparameter and $\theta$ is the weight.	Add penalty
L2 [100] Regularisation	Adds a penalty of $\lambda \sum \theta^2$ to the loss function, also known as weight decay.	Add penalty
Batch [120] Normalisation	Normalises the input to each layer to a mean of 0 and a standard deviation of 1.	Improves training stability
Dropout [251]	Randomly sets a fraction of input units to zero during training.	Deactivate part of network units
Early Stopping [20]	Stops training when the model's validation performance begins to degrade.	Training strategy

Table 2.9: Common deep learning rate schedulers.

Scheduler	Description	Expression
StepLR [118]	Decays the learning rate $\eta$ every fixed number of steps.	$\eta_t = \eta_0 \cdot \gamma^{\lfloor t/\text{step\_size} \rfloor}$ , where $\gamma$ is the decay factor.
ExponentialLR [118]	Decays the learning rate exponentially over time.	$\eta_t = \eta_0 \cdot e^{(-\lambda t)}$ , where $\lambda$ is the decay rate.
CosineAnnealingLR [177]	Decays the learning rate according to a cosine function.	$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_0 - \eta_{\min}) \left( 1 + \cos \left( \frac{t}{T_{\max}} \pi \right) \right)$

## Learning Rate Scheduler

The learning rate is one of the most important deep learning parameters, as it controls the step size of weight updates; too high a learning rate may cause the model to overshoot

optimal solutions, while too low a learning rate can lead to slow or stagnated convergence [118]. A learning rate scheduler modifies the learning rate dynamically based on the training progress, typically reducing it over time to fine-tune the model in later stages of training [230].

Some common learning rate schedulers are summarised in Table 2.9. For instance, StepLR reduces the learning rate by a fixed factor every few epochs, while ExponentialLR exponentially decreases it by a fixed rate at every epoch or step.

### 2.4.3 Computer Vision Tasks

#### Overview

Deep learning has emerged as a transformative tool across diverse domains, leveraging its ability to model complex patterns and representations from large-scale data. In natural language processing, it powers applications such as machine translation, chatbots, and sentiment analysis, enabling systems to understand and generate human-like text [118]. In audio processing, deep learning drives advancements in speech recognition, music generation, and sound classification, effectively capturing temporal and spectral features [110]. In computer vision, it supports tasks like image classification, object detection, and facial recognition, revolutionising visual data interpretation [142]. In recommendation systems, deep learning uses neural collaborative filtering to personalise suggestions in e-commerce and content platforms [107]. These successes stem from deep learning’s hierarchical feature learning and adaptability across multimodal data.

Rapid developments in deep learning have made it a powerful tool, helping achieve the goal of digital, precise, and smart farming. In particular, models proposed for computer vision tasks have been applied to solve a wide range of agricultural problems. Fig. 2.8 illustrates some deep learning applications in agriculture, including plant disease recognition, pest and weed recognition, tree branch recognition, and fruit recognition.

First, plant diseases can severely affect crop health, often beginning with subtle signs like spots or wilting on leaves and stems that can spread rapidly if missed. Deep learning supports early diagnosis by detecting these signs, helping farmers intervene before losses grow [116, 242, 151, 89, 264].

Second, fields often face problems from insects and weeds that disrupt crop growth. Insects like aphids damage crops directly and spread viruses, while weeds compete with crops for nutrients, water, and sunlight, reducing yields. Deep learning enables the effective differentiation of these threats [74, 73].

Third, seed quality is vital for successful planting, particularly for crops like sunflowers, where size, shape, and health influence germination and output. Poor seeds risk uneven growth or crop failure. Deep learning helps automate seed counting, and quality checks ensure only the best are chosen [94, 143].



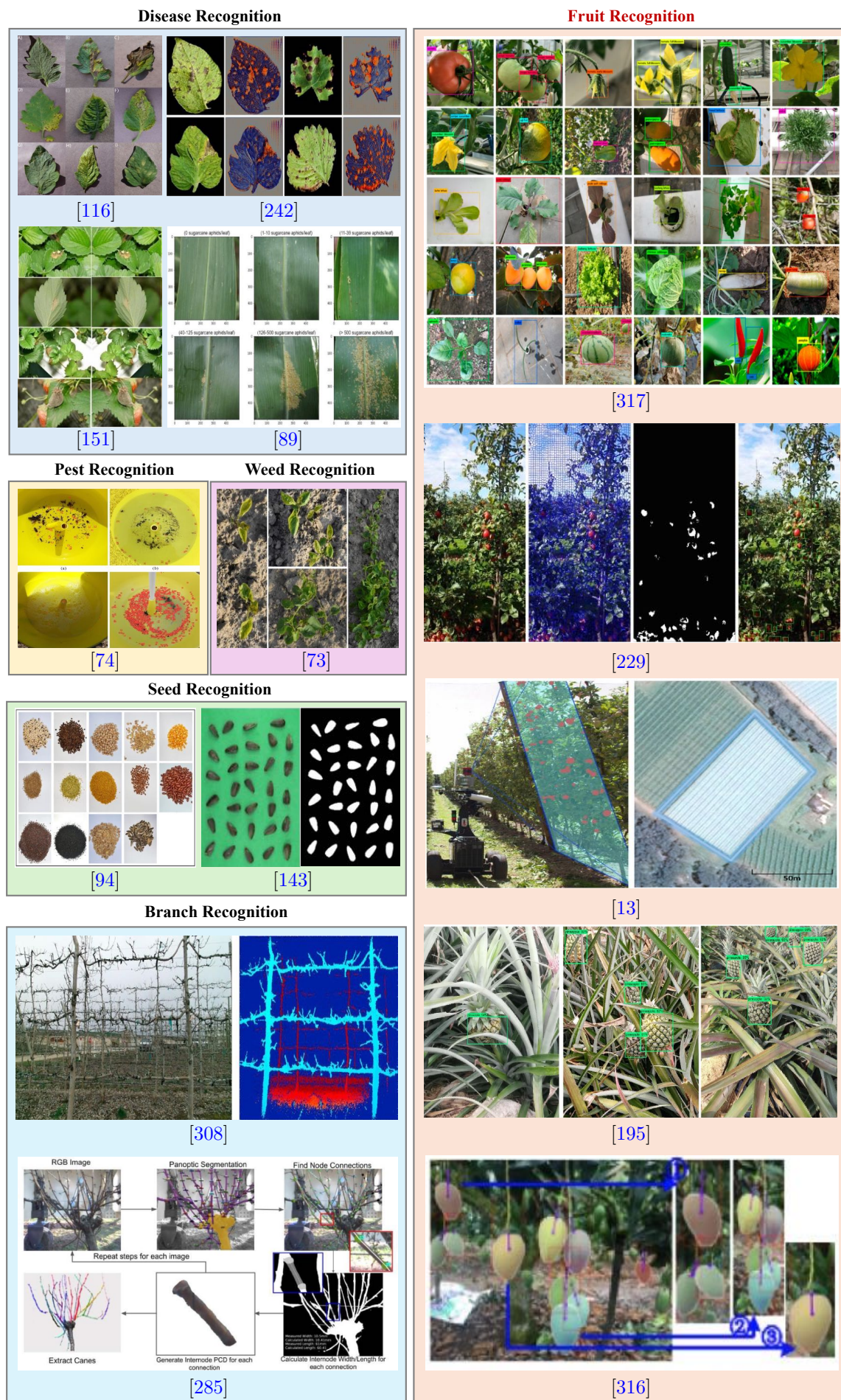


Figure 2.8: Some deep learning applications in agriculture.

Fourth, understanding tree structures is useful in orchard management, where branch patterns are handy for robotic pruning, health checks, or yield estimation. Manual inspection is slow and inconsistent. Deep learning insists on detecting tree branches with high efficiency [308, 285].

Finally, fruit recognition is essential for managing orchards efficiently. Missing ripe fruits or over-picking affects profits and planning. Deep learning enables accurate identification of fruits on trees, improving harvesting accuracy and field decisions [317, 229, 13, 265, 274, 275, 297]. These applications will be explored further in the next section.

In summary, deep learning is changing agriculture by making tasks easier and smarter across different areas. It supports better crop monitoring, resource use, and reduced waste, all at a lower cost. Its growing use is enhancing crop productivity and supporting sustainable farming practices for the future. These improvements help address global food security challenges by increasing food production to meet growing needs.

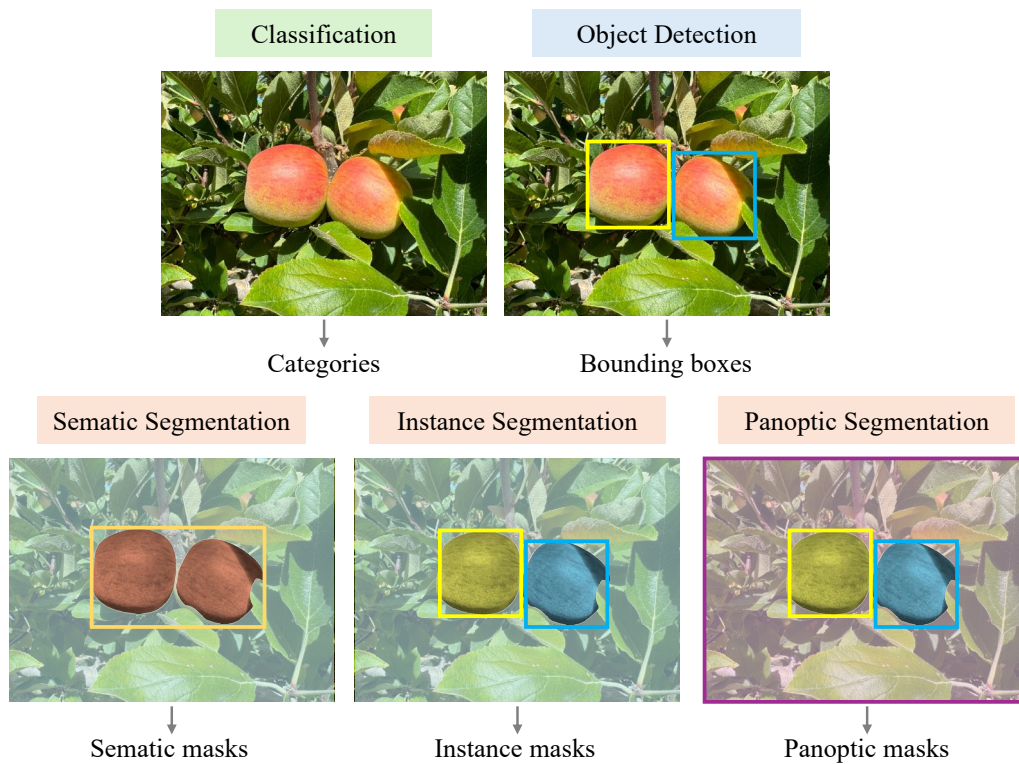


Figure 2.9: The vision tasks of deep learning.

## Fruit Recognition

This thesis focuses on its application within computer vision, a field where deep learning is used to extract and interpret patterns from images. Specifically, various vision tasks are explored, including image classification, object detection, and segmentation (both semantic and instance), that are important to the automated determination of fruit ripeness. The comparison of these vision tasks is illustrated in Fig. 2.9.



Computer vision has been widely adopted for fruit detection, enabling accurate fruit identification and localisation in images or videos. This capability is foundational in fruit production, as it supports tasks such as tracking crop development and guiding automated systems. Additionally, it has been extended to determining fruit ripeness, which is a critical factor in optimising harvest timing and ensuring quality. Recent studies on various fruit applications using deep learning are summarised in Table 2.10.

In terms of model structure, CNN and ViT are two main architectures for computer vision over the past few years. CNN is good at capturing local spatial patterns through convolutional layers, making it highly effective for tasks like image classification and object detection. Its hierarchical structure reduces computational complexity, though it may not capture global context as efficiently, and typically benefits from larger labelled datasets. Meanwhile, ViT leverages self-attention mechanisms to capture long-range dependencies across an entire image, offering excellent performance on tasks requiring global understanding, particularly with ample data. It can be more computationally demanding and often rely on pre-training, but its flexibility makes it highly adaptable. Both architectures have significantly advanced computer vision, each with unique strengths and manageable trade-offs.

**Image Classification** Image classification entails assigning a category to an entire image based on its content. In the context of fruit ripeness determination, this task involves training a model to distinguish between classes such as ripe or unripe fruit.

Several studies have developed CNN models to classify fruit ripeness, such as banana [235], mangosteen [198], mulberry [197], grape [225], papaya [17], tomato [310], and strawberry [75]. These models have proven effective in distinguishing ripe from unripe fruit, showcasing the ability to replace the traditional hand-crafted methods with deep learning models.

However, a key limitation of these studies is that most classification efforts have been conducted in controlled, off-field environments, where fruit images are captured indoors under consistent lighting and background conditions. This reliance on ideal settings makes it challenging to adapt these models for uncontrolled, real-world scenarios.

**Object Detection** Building upon classification, object detection not only identifies the presence of objects within an image but also localises them using bounding boxes. This capability enables the precise identification and isolation of individual fruits in complex scenes.

For example, these applications include the basic detection for fruit counting or yield estimation, such as apple [257], tree inflorescences [291], citrus fruit [179], kiwifruit [71], litchi [160, 130], mango [91, 281], passion fruit [152], pear [226], and strawberry [301, 61, 108]. These efforts demonstrate that deep learning is capable of handling different real

Table 2.10: Deep learning for various fruit recognition.

Task	Fruit	Input	Dataset Size	In or Off field	Method	Metrics	Comments	Ref.
C	Banana	RGB	273	Off	CNN	Acc	Classify bananas into four ripeness stages.	[235]
C	Mangosteen	RGB	1220	Off	CNN	P, R, F1	Classify mangosteens into 6 ripeness stages.	[198]
C	Mulberry	RGB	2000	In	CNN	Acc	Classify two types of mulberries into four ripeness stages.	[197]
C	Multiple	RGB, hyper-spectral	2670	Off	CNN	Entropy per band	Classify hyper-spectral fruit images with CNN pre-trained by RGB images.	[253]
C	Grape	RGB	1008	Off	CNN	Acc	Classify two types of grapes into six ripeness stages in the laboratory.	[225]
C	Tomato	RGB	200	Off	CNN	Acc	Classify tomatoes into five different ripeness categories.	[310]
C	Tomato	RGB	633	Off	CNN	Acc	Classify tomatoes into two ripeness stages.	[267]
C	Tomato	RGB	4419	Off	CNN	Acc	Classify tomatoes into eight ripeness stages based on the colour and shape.	[57]
C	Papaya	RGB	300	Off	CNNs	Acc	Classify papayas into three ripeness stages: immature, partially mature and mature.	[17]
C	Strawberry	hyper-spectral	240	Both	CNN	Acc	Classify the early ripe and ripe strawberries in the field and laboratory.	[75]
D	Apple	RGB	524	In	MTYOLOX	AP	A Transformer-based CNN model to detect tree inflorescences in orchards.	[291]

Abbreviations: C (Classification), D (Object Detection), S (Segmentation), Acc (Accuracy), P (Precision), R (Recall), AP (Average Precision), RMSE (Root Mean Squared Error)



Table 2.10 (Continued): Deep learning for various fruit recognition.

Task	Fruit	Input	Dataset Size	In or Off field	Method	Metrics	Comments	Ref.
D	Apple	RGB	1361	In	FBoT-Net	AP	A focal bottleneck Transformer-based model for small green apple detection.	[257]
D	Apple	RGB	480	In	YOLO	F1	Detection of apples in three different growth stages.	[265]
D	Apple	RGB	4000	Off	YOLO	AP	Classify apple ripeness into three stages using YOLOv8 models.	[292]
D	<i>Camellia oleifera</i> fruit	RGB	1148	In	YOLO	AP	Improved YOLOv5 for the detection of occluded <i>C. oleifera</i> fruit.	[35]
D	<i>Camellia oleifera</i> fruit	RGB	3228	In	YOLO	AP	Lightweight YOLO to detect <i>C. oleifera</i> fruit maturity in orchards.	[323]
D	Citrus fruit	RGB	557	In	YOLO	AP	Lightweight YOLO model to detect green citrus fruit.	[179]
D	Green tomatoes	RGB	854	In	TEAVit	MSE	A camouflage object detection model for identifying green tomatoes.	[307]
D	Kiwifruit	RGB	700	In	Faster R-CNN, ZFNet	AP, Acc	Integrate two CNN models to detect kiwifruit in the field.	[71]
D	Litchi	RGB	3979	In	YOLO	AP	Detect litchi fruit and stem in the nighttime natural environment.	[160]
D	Litchi	RGB	3500	In	YOLO	AP	An optimised YOLOv3 for litchi detection on edge devices.	[130]
D	Mango	RGB	2147	In	YOLO	AP	Detection of multi-species mango fruits and fruiting stems.	[91]

Table 2.10 (Continued): Deep learning for various fruit recognition.

Task	Fruit	Input	Dataset Size	In or Off field	Method	Metrics	Comments	Ref.
D	Mango	Video	110 frames	In	MangoYOLO	RMSE	Dual-view method for on-tree mango detection, tracking, and counting.	[281]
D	Multiple	RGB, NIR	122	In	Faster R-CNN	F1	Fruit detection using imagery obtained from two modalities of RGB and NIR.	[231]
D	Multiple	RGB	2760	In	CNN	AP	Detect common fruit blooms and classify into 12 categories.	[320]
D	Passion fruit	RGB	917	In	YOLO	AP	Improved YOLOv8n model for edge device-based passion fruit detection.	[152]
D	Pear	RGB	5213	In	YOLO	AP	Detect phenological period of ‘Yuluxiang’ pear in orchard environment.	[226]
D	Strawberry	RGB	4002	Off	YOLO	AP	Stolon-YOLO detection model for strawberry seedlings in a greenhouse.	[301]
D	Strawberry	RGB	1968	In	YOLO	AP	Classify strawberries into seven growing stages from near-ground imaging.	[321]
D	Strawberry	RGB	1538	In	DSW-YOLO	P, R, AP	Detection of ground-planted ripe strawberries and their occlusion levels.	[61]
D	Strawberry	RGB	1089	In	LS-YOLOv8s	AP	Add YOLOv8 with the LW-Swin Transformer to detect strawberry ripeness.	[299]
D	Strawberry	RGB	1540	In	YOLO	AP	Real-time strawberry detection based on YOLOv5s for robotic harvesting.	[108]
D	Strawberry	RGB	3262	In	YOLO	AP	STRAW-YOLO detection model for strawberry fruits targets and key points.	[185]

Table 2.10 (Continued): Deep learning for various fruit recognition.

Task	Fruit	Input	Dataset Size	In or Off field	Method	Metrics	Comments	Ref.
D	Tomato	RGB	500	Off	CNN	P, R, F1	Classify tomatoes into five ripeness stages in a laboratory.	[141]
D	Tomato	RGB	932	In	YOLO	P, R	Lightweight real-time tomato detection model for mobile deployment.	[306]
D	Tomato	RGB	917	In	RTDETR	AP	Detection method for tomatoes based on an improved RTDETR model.	[92]
S	Apple	RGB	211	In	U-net	AP	Ensemble U-Net for green apple semantic segmentation in orchard environment.	[158]
S	Apple	RGB	1100	In	Multi-scale MLP, CNN	F1	Semantic segmentation for apple fruit detection and yield prediction.	[13]
S	Apple	Video	30 frames	In	Gaussian Mixture Models	F1, Acc	Apple instance segmentation for detection and yield prediction.	[229]
S	Apple	RGB	1240	In	Mask R-CNN	P, R, F1	Improve Mask R-CNN for apple instance segmentation in orchards.	[274]
S	Apple	RGB	268	In	RS-Net	AP	Extend the vanilla Mask R-CNN for the instance segmentation of green apples.	[129]
S	Cucumber	RGB	522	In	Mask R-CNN	P, R, F1	Cucumber fruits instance segmentation in greenhouses.	[170]
S	Cucumber	RGB	223	In	YOLO, U-net	P, R, F1	YOLO for detection and U-net for semantic segmentation.	[6]
S	Grape	RGB	300	In	R-CNNs	AP, P, R, F1	Grape detection, segmentation and counting in orchards.	[233]
S	Grape	RGB	980	In	DualSeg	AP	Combine the CNN and Transformer blocks for grape semantic segmentation.	[275]

Table 2.10 (Continued): Deep learning for various fruit recognition.

Task	Fruit	Input	Dataset Size	In or Off field	Method	Metrics	Comments	Ref.
S	Grape	RGB	958	In	YOLO	AP	An improved lightweight YOLOv8 for the grape pedicel instance segmentation.	[245]
S	Mango	RGB	44	In	CNN	AP	A deep semantic segmentation CNN to detect and count mangoes.	[135]
S	Mango	RGB	409	In	Mask R-CNN	AP	Mango picking algorithm on instance segmentation and key point detection.	[316]
S	Multiple	RGB	2068	In	FoveaMask	AP	Segment apple and permission in complex environment.	[128]
S	Multiple	RGB	920	In	CNN	AP	Segment green apple and peach in orchard environment.	[246]
S	Strawberry	RGB	2000	In	Mask R-CNN	AP	Classify strawberries into two ripeness stages and estimate the picking points.	[304]
S	Strawberry	RGB	3100	In	Mask R-CNN	AP	Instance segmentation of strawberries in the field and a large annotated dataset.	[217]
S	Strawberry	RGB	700	In	Mask R-CNN	AP	A three-stage method to classify into six ripeness stages.	[262]
S	Strawberry	RGB	5000	In	YOLO	AP	Real-time strawberry segmentation in an unstructured environment.	[273]
S	Tomato	RGB	800	In	Faster R-CNN	RMSE	Semantic segmentation for accurate detection of ripe tomatoes on plants.	[114]
S	Tomato	RGBD	1078	In	Mask R-CNN, YOLO	Acc	Improved Mask R-CNN for visual recognition of cherry tomato in fields.	[297]
S	Tomato	RGB	/	In	YOLO	AP, F1	Fruit bunch detection, and the segmentation of pedicel and main-stem.	[159]

scenarios, whether the fruit is in-field with lots of occlusions or at nighttime with limited lighting.

The limitation of the above studies is that they do not consider fruit ripeness during the detection process. There are some exceptions, such as apple [265, 292], strawberry [321, 299], and tomato [141], which combine ripeness with locating fruits.

It is noted that most of the current fruit detection models are based on the You Only Look Once (YOLO) series. For instance, [292] applied YOLOv8 to classify apple ripeness into three stages. In addition, [108] proposed a real-time improved YOLOv5s for robotic strawberry harvesting. [301] introduced Stolon-YOLO to detect strawberry stolons in greenhouse environments. [35] combined fusion clustering with YOLOv5 to tackle *Camellia oleifera* fruit detection under multiple occlusions. [323] further extended this task by providing modified lightweight YOLO for *C. oleifera* fruit ripeness determination in orchards. [226] presented YOLO-RCS to detect the phenological period of Yuluxiang pears. [185] adopted STRAW-YOLO for the identification of strawberries and their key points.

However, detection models help find fruits but only give the boxes' coordinates around fruits, which does not provide more details about the fruit. This creates a gap in figuring out fruit ripeness accurately with detection models only.

**Segmentation** Segmentation takes the detection even further by providing detailed, pixel-level information about fruits, enabling a detailed spatial understanding of the image. It contains three categories: semantic, instance and panoptic segmentation. As panoptic segmentation also segments the background, which is usually unnecessary for fruit ripeness studies, it is not discussed further.

Semantic segmentation labels all pixels belonging to a specific class without distinguishing between separate instances. For example, in an image of multiple apples, all apple pixels would be labelled as "apple", irrespective of whether they belong to distinct objects. Some studies have been conducted on fruit semantic segmentation, which labels all pixels of a fruit type as one category. Examples include apple [13, 158], cucumber [6], grape [275], mango [135], and tomato [114].

Unlike semantic segmentation, instance segmentation differentiates between individual objects of the same class, making it ideal for analysing clusters of fruits. For instance, it can segment and identify each fruit in a bunch of apples, facilitating ripeness evaluation on a per-fruit basis. Some studies have explored fruit instance segmentation, which gives a unique label to each individual fruit. It provides more precise details for fruit ripeness determination, therefore, it is the key focus of this thesis.

Fruit instance segmentation models can be grouped into two types: CNN-based and Transformer-based.

On the one hand, CNN-based models have dominated vision architectures for a long

time, of which Mask R-CNN [105] is the pioneer of instance segmentation. Lots of fruit instance segmentation models are based on Mask R-CNN. For example, [233] showcased that Mask R-CNN was effective in detecting, segmenting, and tracking grape clusters, demonstrating its robust performance across significant variations in shape, colour, size, and compactness. [127] improved Mask R-CNN by adopting ResNet and DenseNet as the backbone to construct a picking robot vision detector, which improves the recognition accuracy of apples in the environment of overlaps and occlusions. [202] introduced Mask R-CNN to detect and segment individual blueberries of four cultivars and classify them into two maturity stages. Similarly, Mask R-CNN and its modified version are adopted to build strawberry fruit detectors with better universality and robustness than traditional machine vision algorithms [304, 217].

Besides, only few CNN-based segmentation models are not based on Mask R-CNN. For example, [128] constructed a fast segmentation model for green fruits FoveaMask, which integrates a position attention module into the embedding mask branch to aggregate valuable information. [133] proposed a one-stage detection model DaSNet-v2, which combines an instance and a semantic segmentation branch, performing apple instance segmentation and tree branches semantic segmentation. [246] designed an edge-guided fruit segmentation model, which included modules specially designed to locate potential target areas and sharpen the edges. [245] proposed a multi-scale adaptive YOLO for grape pedicel instance segmentation.

On the other hand, Transformer-based models have gained more attention recently. Based on the vanilla ViT, several Transformer-based models like DETR [26], MaskFormer [40] and SegFormer [293], etc, have arisen.

Some studies combined Transformer models with CNN models to perform segmentation. For example, [275] proposed a parallel network structure DualSeg, which leverages the advantages of CNN at local processing and Transformer at global interaction for grape peduncle segmentation. [291] introduced a Transformer-based CNN model MTY-OLOX, for the detection of full tree inflorescences in the uncontrolled orchard environment. [257] presented a focal bottleneck Transformer network FBoT-Net, to incorporate global semantic information with local feature information through the focal bottleneck Transformer module for small green apple detection.

It can be seen that Transformer-based models have not been widely applied in fruit detection. Here are some other applications in relevant agricultural sectors. [206] proposed a segmentation model HSI-TransUNet for crop mapping, which processes the spatial and spectral information of HSI and UAV images simultaneously. [151] presented an improved Transformer-based model to achieve fast recognition of multiple classes of strawberry diseases.

However, most current work still relies on Mask R-CNN and YOLO models. The potential of Transformer in fruit instance segmentation has not been well explored. In-

stance segmentation is more complex than classification and detection, one of the reasons is that pixel-level ripeness labels are tough to get. Therefore, there is a gap in applying fruit instance segmentation for ripeness determination.

## Integration with In-field Robots

In fruit production, robots bring deep learning models into the field for practical use. Locating fruits quickly and accurately is the foundational prerequisite for building a fruit-picking robot. These robots sense their surroundings and guide a mechanical arm to pick the fruit. They can alleviate labour shortages by taking over repetitive tasks, improving production efficiency.

Fig. 2.10 illustrates several typical in-field fruit-picking robots. Most agricultural robots share a similar setup: an autonomous mobile platform, a light mechanical arm with multiple joints, a force feedback system with a flexible end effector, a vision system with many sensors, a drive control system, a decision-making system, and auxiliary software and hardware [263]. Significant research efforts across the world have been devoted to the development of fruit-harvesting robots in recent years. For example, [122] built a multi-purpose gripper with vacuum suction and rotation, evaluated in both laboratory and commercial orchard settings, achieving a 66.1% success rate for apple thinning and showing promise for harvesting. Other examples include automatic picking systems for apple [133], litchi [160], and radiata pine [190].

Before deep learning became common, researchers used basic vision tools to locate fruits. For instance, [247] designed a cost-effective harvester that picked 84% of the apples in a commercial orchard using a simple machine vision algorithm. Similarly, [34] proposed vision algorithms for fruit-picking robots to move, locate themselves, and pick fruits efficiently. However, these methods are specifically designed for a particular fruit, limiting their generalisability to other fruit types.

The rise of deep learning has sped up the growth of in-field robot systems. It is now widely used to build the vision system or software parts for fruit-picking robots. For example, [24] evaluated an apple harvester with YOLOv4, finding that a “horizontal pull with bending” motion outperformed a human-like one in success rate and speed while avoiding stem-pulling and bruising. [205] used an improved YOLOv4 to search for sweet peppers in dense fields, using the centre of each detection box as the picking target point. [153] applied an improved YOLOv5s for a dragon fruit robot to work in day and night environments and deployed it on a mobile device. [134] built an apple system with a light detection network and PointNet, achieving an 80% success rate and a cycle time of 6.5 seconds in both laboratory and orchard tests. [195] introduced a spatio-temporal model to detect in-field pineapples accurately for a picking robot. [144] developed a dual-arm robotic apple-harvesting system with O2RNet detection model, achieving a 60% success





Figure 2.10: Some fruit harvesting robots.



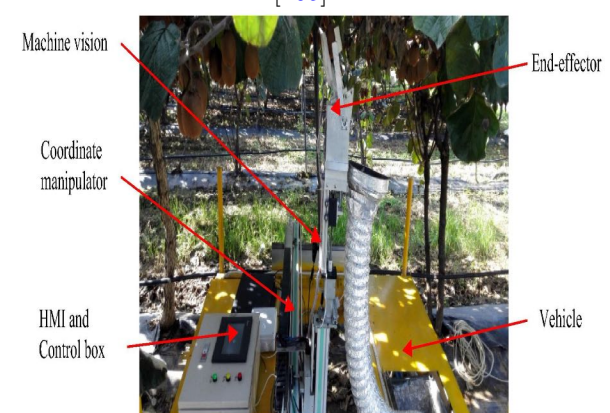
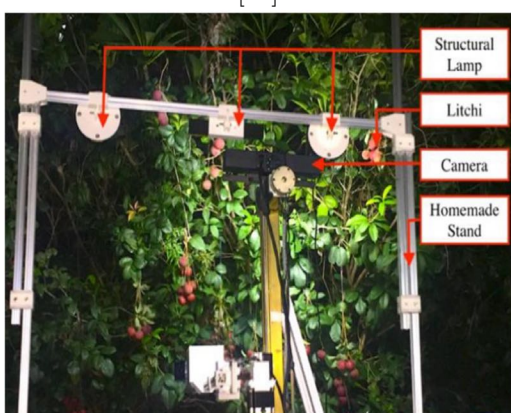
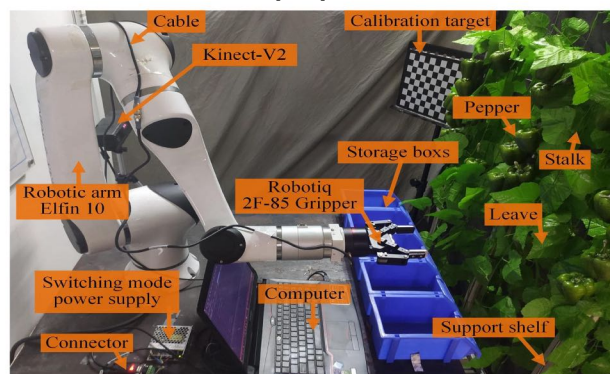
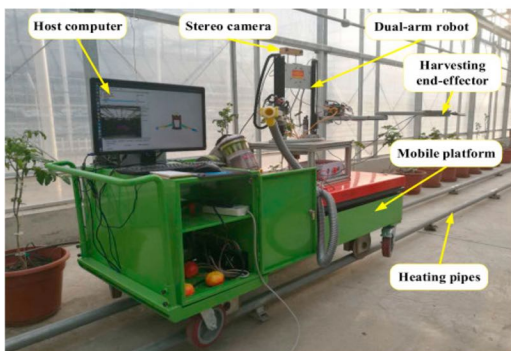
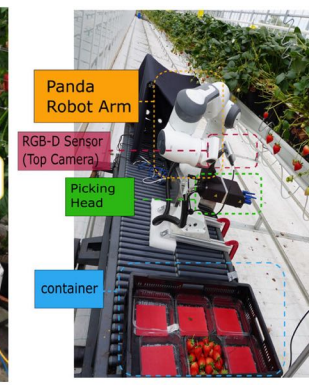
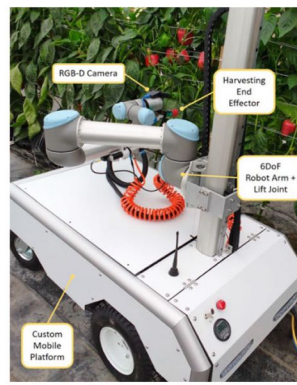
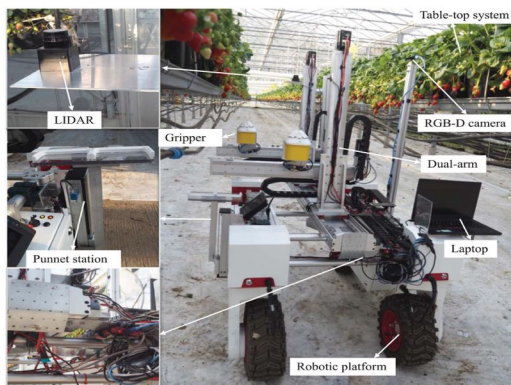


Figure 2.10 (Continued): Some fruit harvesting robots.

rate in field trials and demonstrating significant harvest time improvement. [212] used Faster R-CNN for an asparagus-harvesting machine. For kiwifruit, [199] applied Faster R-CNN to locate fruits in RGB images and extract their coordinates. [309] proposed an apple harvester prototype that integrates a Mask R-CNN for segmentation, which contains a three-joint arm and a vacuum gripper, showing good performance in field tests.

It can be seen that detection models are the most common choice for robot vision systems to find fruit in the field, which only gives the coordinates of the fruits' bounding boxes. Segmentation models, especially instance segmentation models, give detailed pixel-level information about fruits, which helps robots not just locate fruits but also check fruit ripeness in the field. Then robots can decide if the fruits are ready to be picked, thus harvesting them selectively. This shows there is a gap between the segmentation model with in-field robots. Developing fast and efficient segmentation models for robots is necessary to improve fruit production efficiency.

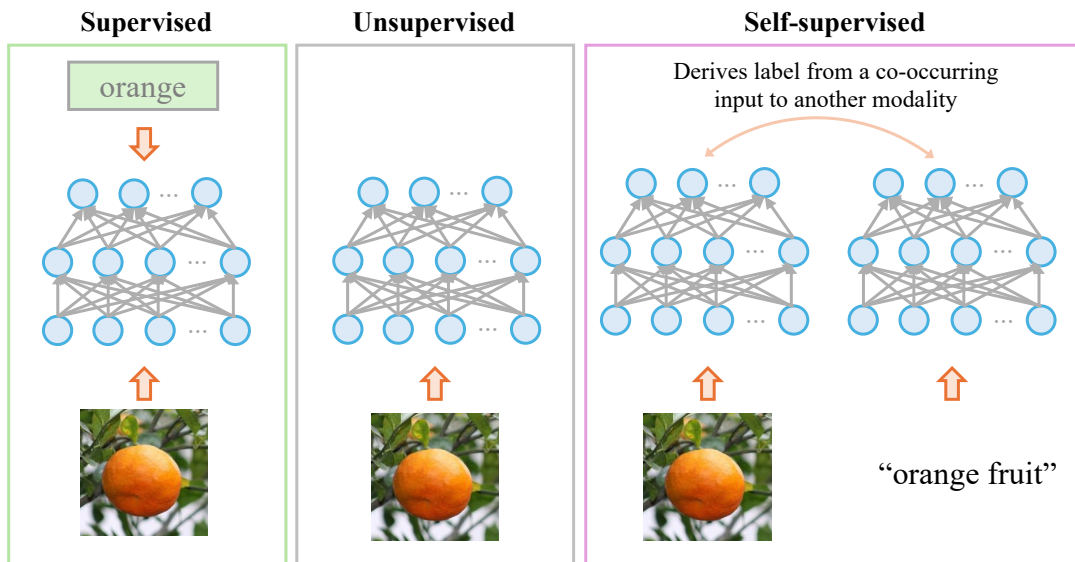


Figure 2.11: The comparison of supervised, unsupervised, and self-supervised learning.

#### 2.4.4 Self-Supervised Learning

The deep learning models discussed earlier, such as those for detection and segmentation, are supervised models. These models usually need a lot of labelled data to achieve satisfactory performance. However, collecting and labelling such data is expensive and time-consuming. It often requires skilled people with expert knowledge to mark the data correctly, which can slow down research and real-world applications.

Self-supervised learning (SSL) is one of the solutions to tackle these problems [93]. It helps models learn useful features from large amounts of unlabelled data, eliminating the need for human labels. This makes it faster and cheaper to build models, especially

when labelled data is hard to get. Fig. 2.11 shows the difference between supervised, unsupervised and self-supervised learning, based on [58]. Unsupervised learning looks for patterns in the input data without any guidance. SSL is a subset of unsupervised learning, which aims to learn discriminative features from unlabelled data on its own.

A key concept in SSL is the pretext task, which is a carefully designed objective that encourages the model to learn meaningful representations without requiring explicit human annotations [93]. Specifically, “pretext” denotes that the task being solved is not the primary objective but serves as a means to generate a robust pre-trained model. Then, the acquired models can be fine-tuned with a small amount of labelled data for downstream tasks such as detection and segmentation, saving time and effort.

There are some common pretext tasks, including context-based methods, contrastive learning, generative algorithms and contrastive generative methods [93]. This thesis focuses on contrastive learning and generative algorithms, which are represented by masked image modelling. Fig 2.12 illustrates the basic structures of contrastive generative and masked image modelling, of which the actual details can be different depending on method designs.

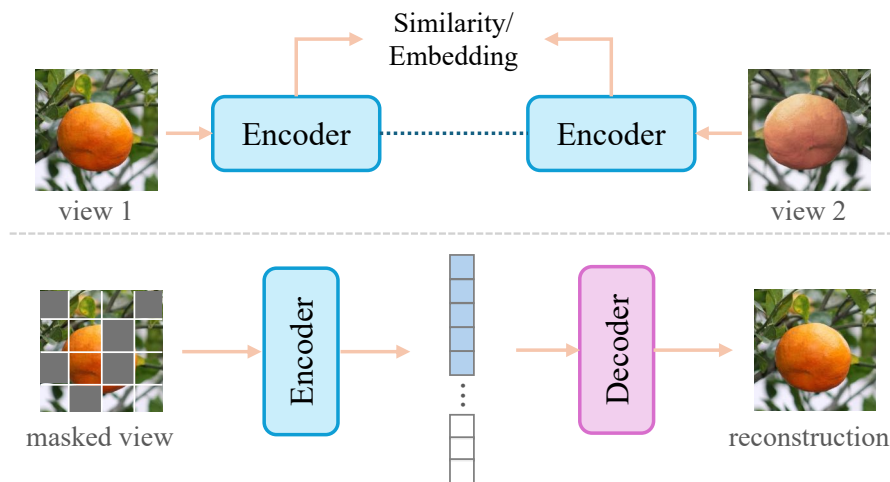


Figure 2.12: Contrastive learning (top) and masked image modelling (bottom).

## Contrastive Learning

Contrastive learning (CL) is built upon the foundation of simple instance discrimination tasks. It trains models to distinguish between similar and dissimilar data points by maximising the similarity between positive pairs (similar instances) and minimising the similarity between negative pairs (dissimilar instances). CL usually has an encoder-only structure.

Early CL methods were based on negative examples. For instance, MoCo [104] keeps a memory of negative examples and updates it to compare with positive pairs. SimCLR [36] takes a simple design by using lots of negative examples from the same batch, boosting



feature quality with strong image augmentations like flipping or cropping. These methods are built upon the foundation of simple instance discrimination tasks, for which negative examples can be used as supervision.

Later, self-distillation-based CL came up to avoid the need for negative examples. These models focus on matching two changed versions of the same image. BYOL [90] uses one network to predict another's output, learning features without negatives. SwAV [28] adds clustering to group similar images, which aims to make two codes derived from different views consistent. SimSiam [38] keeps it simple by matching two views of an image directly, while FastSiam [221] speeds up the matching for quicker training. These methods stop the model from giving the same answer for everything, a problem called collapsing.

Another type, feature decorrelation-based CL, aims to make learned features stand apart from each other. Barlow Twins [305] pushes features to be different while keeping them tied to the image, using a trick to balance them. VICReg [11] adds rules to keep features varied and stop them from shrinking or growing too much. These methods help the model pick up clear, unique details from data, which can improve tasks like spotting fruits in messy field images.

## Masked Image Modelling

Masked image modelling (MIM) is another SSL method that trains models by covering up parts of an image and asking them to predict what is hidden. By filling in the blanks, the models learn how different parts of an image fit together, helping to figure out the shape, texture, and details of objects. Different from CL methods, MIM leverages co-occurrence relationships among image patches as supervision signals. MIM usually has an encoder-decoder structure.

MIM has gained significant popularity and demonstrated good performance. MAE [103] covers big chunks of an image, up to 75%, and trains the model to reconstruct them using a simple encoder-decoder design. It is fast and works well as it forces the model to focus on the big picture, not just small details. SimMIM [294] takes a lighter method, using smaller masks and a direct prediction approach. It is simpler and quicker, making it handy for real-time tasks. BEiT [10] adds a twist by turning image patches into tokens, like words in a sentence, and predicts them using a transformer model. This teaches the model deeper connections. Context Autoencoder [37] focuses on nearby patches to guess the missing ones, helping the model learn local details, like object edges or colour changes.

MIM offers a different way to learn from images compared to CL. It digs into the details within a single image, using patch links to build strong features without labels. MIM emphasises the importance of a robust representation that remains resilient to

input noise. This fits well with tasks like segmentation for fruit ripeness, where every pixel counts.

## Applications

SSL has been explored in various agricultural studies to address different challenges. The examples below demonstrate how SSL can perform well with little or no labelled data, offering a practical tool for farming tasks.

For classification tasks, SSL helps identify plant conditions and features. [314] built a CL method to find leaf diseases, adapting to new settings with few labels to quickly spot health problems. Similarly, [167] developed a transformer-based approach that pre-trains on unlabelled data with a masking step and then classifies pests and diseases, proving SSL can tackle tough farm issues. [319] took SSL further by using drone data to estimate soybean yield and check lodging, blending different data types with CL to follow growth from flowering to maturity. Along the same lines, [72] created SSMDA, a cherry maturity classification model that uses multi-feature CL to assess ripeness with minimal labelled data, focusing on details like colour and shape.

For detection tasks, SSL helps find unusual patterns. [45] applied it to spot crop anomalies by mixing up image channels, training the model to catch oddities like damage or disease in crops. In a related effort, [171] paired an autoencoder with an SSL classifier to detect strawberry anomalies using hyperspectral images, finding odd fruits without needing many labels.

For segmentation tasks, SSL helps outline plant parts with precision. [80] used a semi-self-supervised method for wheat head segmentation, adjusting deep learning to new fields with just a few labelled images to mark wheat heads. Likewise, [157] designed CoRE-Net, a two-step SSL approach for leaf vein segmentation, starting with a bit of supervised training before switching to self-supervised learning to trace vein patterns like branching with little labelled data. Another study, [46], introduced SSL-NBV for 3D plant reconstruction by robots, predicting the best scanning angles from unlabelled data to help robots map plants efficiently.

These studies highlight SSL's promising ability in agriculture, covering tasks from spotting diseases to mapping plants. Still, most efforts target disease detection, anomaly finding, or yield tracking, with little work on fruit detection and ripeness determination. This gap shows a clear need for SSL in fruit ripeness research, making it a key area for this thesis to investigate, especially for improving in-field robot systems.

## 2.5 Summary

This chapter reviewed the development of fruit ripeness determination methods, from sensing techniques to machine learning and deep learning approaches.

Sensing methods, such as spectroscopy and fluorescence analysis, can provide detailed internal insights but have major limitations. These include high equipment costs, the need for controlled environments, complex calibration, and poor scalability, making them unsuitable for practical in-field use.

Machine learning methods offer more flexible and cost-effective alternatives. They can process visual and sensor data for non-destructive ripeness estimation. However, many ML models depend on hand-crafted features and struggle with complex or high-dimensional data, which limits their general use in varied field conditions.

Deep learning has emerged as a more powerful approach, capable of automatically learning features from raw data. It has been widely applied in agriculture for tasks such as fruit detection, classification, and segmentation, offering better accuracy and adaptability than earlier methods. Among these tasks, instance segmentation is especially relevant to ripeness evaluation, as it enables per-fruit analysis. CNN-based models like YOLO and Mask R-CNN have shown good results, while Transformer-based models are being explored for their ability to capture global features. However, challenges remain in dealing with occlusions, variable lighting, and the need for large labelled datasets.

Integrating deep learning models with in-field robots enables selective harvesting, which can help reduce labour demands. However, most current systems use object detection rather than segmentation. To help develop in-field robots, lightweight instance segmentation models are needed.

One of the main limitations of existing deep learning methods is their reliance on large amounts of labelled data. To address this, self-supervised learning is a promising solution. It allows models to learn from unlabelled data using methods like contrastive learning and masked image modelling. Although self-supervised learning has been used in other agricultural tasks, its use in ripeness estimation is still limited.

In conclusion, deep learning is a promising method for fruit ripeness determination. However, its effectiveness is still limited by some challenges mentioned above. This thesis explores instance segmentation, self-supervised learning models tailored for practical use in real agricultural environments.

## Chapter 3

# PeachSOLO: A One-stage Instance Segmentation Model for Peach Ripeness Classification

### 3.1 Introduction

Despite the progress made in fruit instance segmentation and ripeness classification using deep learning, the literature reveals several practical limitations that hinder real-world deployment. Most existing datasets are limited in scale and diversity, especially in terms of representing the full spectrum of ripeness stages under complex natural conditions such as variable lighting, occlusion by leaves or stems, and multi-fruit overlap. In addition, while two-stage models like Mask R-CNN offer strong performance, they are often computationally intensive and less suitable for resource-constrained applications in agricultural settings.

This chapter addresses these challenges by introducing a new large-scale and high-quality annotated dataset called NinePeach, which captures nine peach cultivars at various ripeness stages in real orchard environments. To effectively leverage this dataset, this chapter proposes PeachSOLO, a novel one-stage, anchor-free instance segmentation model that integrates both channel and spatial attention mechanisms to enhance feature representation. Without relying on a region proposal network, PeachSOLO directly detects peaches and classifies their ripeness with improved precision and efficiency. Empirical evaluations demonstrate that PeachSOLO surpasses state-of-the-art models in both accuracy and segmentation quality, while maintaining faster inference speeds, making it well-suited for practical use.

The remainder of this chapter presents the dataset construction process, the architecture of PeachSOLO, and detailed experimental analyses that validate its effectiveness in challenging in-field scenarios.

## 3.2 Dataset

### 3.2.1 Image Collection

Peach images were collected from an experimental orchard in Huazhong Agricultural University, Wuhan, China. The collection was conducted between May and June 2022 and included nine cultivars of peaches: Dahongpao, Qingfeng, Chunmei, Chunmi, Chunxue, Songsen, Maotao, Youpantao, and XiahuiNo5. The image capture device used was a smartphone whose specifications are detailed in Table 3.1.

Table 3.1: Specifications of the mobile device used.

Aspect	Details
System OS	Android 11
CPU	Octa-core (1×3.2 GHz Kryo 585 3×2.42 GHz Kryo 585 4×1.80 GHz Kryo 585)
Main Camera	Sony IMX598(1/2")
Focal Length	4.7 mm

The camera was positioned at a distance of 30-50 cm from the peaches and captured images from various angles. It is noted that all peach images were acquired under natural lighting conditions and in real-world production settings, where the peaches exhibited diverse physical configurations. These configurations include but are not limited to isolated peaches, peaches that are in close proximity to one another, peaches that are partially obscured by leaves, stalks or other peaches, and peaches that are illuminated from the opposite side. The images were originally captured and stored in JPEG format at a resolution of 4000×3000 pixels. Samples of images from each kind of peach are presented in Fig. 3.1.

### 3.2.2 Image Preprocessing

A total of 3849 images were selected to form the dataset, representing nine cultivars of peaches that have been classified into three distinct stages of ripeness: unripe, semi-ripe, and ripe. Two annotators were independently in charge of carrying out the image labelling process, and one reviewer would make decisions in case of disagreement. All cultivars of peach were annotated individually.

Each image was manually labelled using Label Studio [266], generating ground-truth labelled images containing individual segmentation masks of all peaches depicted in the image. The labelling process adhered to a rigorous standard, which involved generating a precise mask for each peach captured in the image, even in challenging scenarios where





Figure 3.1: The 9 cultivars of peach in NinePeach.

the peaches may have appeared nearly imperceptible due to distance, occlusions, or their proximity to the image boundaries. The overview of Label Studio is shown in Fig. 3.2.

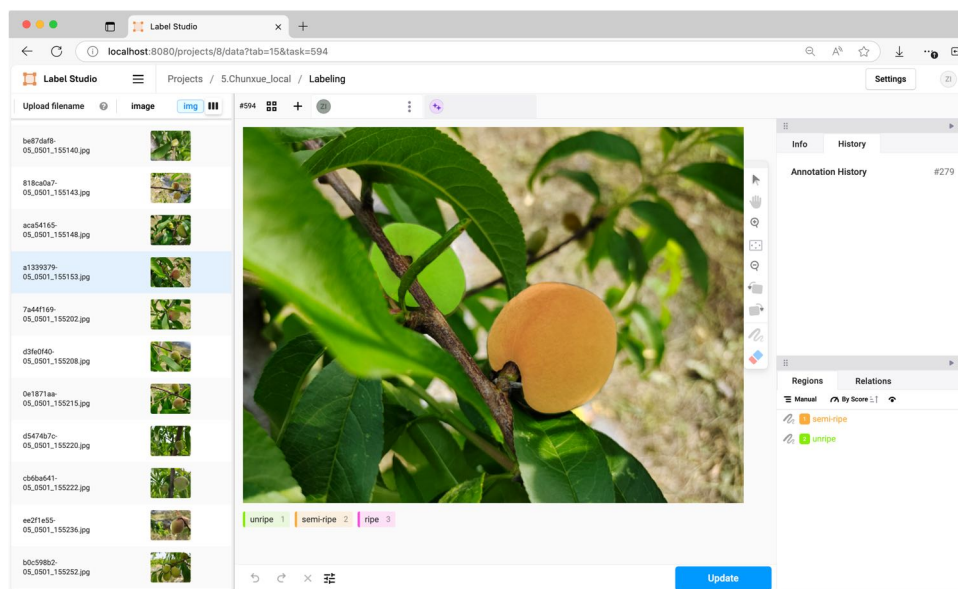


Figure 3.2: The overview of Label Studio.

The instance category distribution of 3849 peach images is presented in Fig. 3.3. Some cultivars lack images of the ripe stage due to objective conditions. For example, there are relatively fewer Chunmi and Songsen trees, and their ripe fruits are dropped by weather or picked by animals, whilst Maotao takes a longer time to become ripe than other cultivars, which exceeded the collection schedule. Therefore, the “long-tail” phenomenon exists in

the dataset, which is discussed in Section 3.5.1.

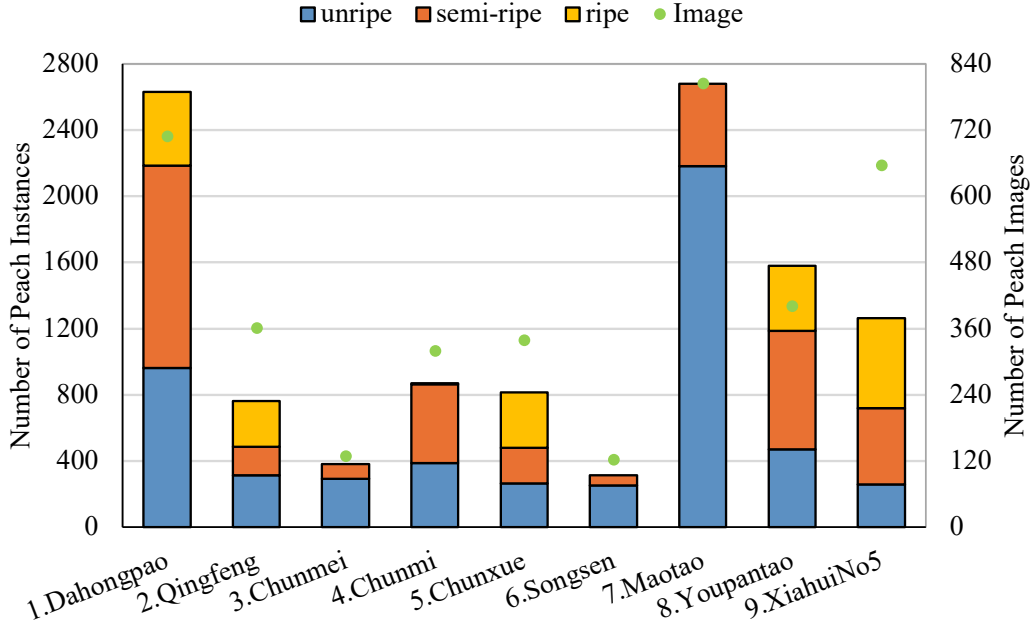


Figure 3.3: Instance category distribution of NinePeach.

To reduce the computational requirements of the models, the images were resized to  $1024 \times 768$  pixels using the ‘mogrify’ library. For every cultivar of peach, the images were randomly split with a ratio of 7:3 for training and validation sets, respectively. Then, the individual training sets and validation sets were combined to form a total training set of 2690 images and a total validation set of 1159 images. To alleviate the “long-tail” problem, the number of semi-ripe and ripe instances was increased by oversampling 750 randomly selected images that did not contain unripe instances to make the category distribution more balanced. The data augmentation methods used included random angle rotation, random jitter, and random flipping. Thus, the balanced dataset called NinePeach was created to contain 3240 images for training and 1359 images for validation. The instance category distribution of the dataset is detailed in Table 3.2. A sample of annotation is shown in Fig. 3.4.

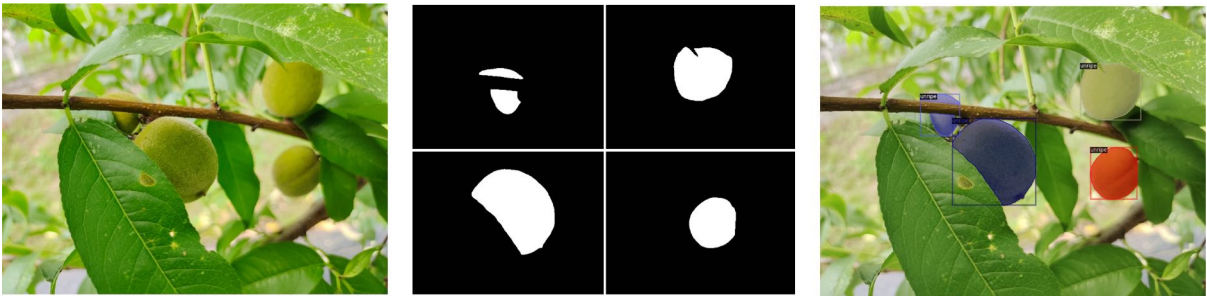


Figure 3.4: Original image (left), individual masks (middle), and annotated image (right).

### 3.2.3 Dataset Summary

The NinePeach dataset<sup>1</sup> comprises 4599 images ( $1024 \times 768$ ) of nine peach cultivars, which were taken under natural illumination and in real-world production settings, including peaches with factors like different intensities of natural light, multi-fruit adhesion, and occlusion caused by stems and leaves. This dataset is divided into training (3240 images) and validation (1359 images) subsets, and each peach is categorised into three ripeness stages: unripe, semiripe, and ripe.

Table 3.2: The instance category distribution of NinePeach dataset.

Category	Original		Balanced	
	Training	Validation	Training	Validation
unripe	3669	1717	3669	1717
semiripe	2768	1140	3312	1307
ripe	1403	589	1689	737
Total	7840	3446	8679	3761

A statistical overview of NinePeach is presented in Table 3.3. The distribution of peach instances demonstrates a relatively balanced representation across different sizes, with a higher proportion of large-sized instances in the training set. The number of instances per image and the pixel ratio remain stable across all sets, indicating a consistent frequency of occurrence and an overall average peach shape.

Table 3.3: Statistics of the NinePeach dataset.

Aspect	Category	Training	Validation
Ratio of size	Small ( $\text{area} \leq 32^2$ )	0.01	0.01
	Medium ( $32^2 < \text{area} \leq 96^2$ )	0.18	0.17
	Large ( $\text{area} > 96^2$ )	0.81	0.82
Mean/standard deviation	Number of peach instances	2.68/2.06	2.77/1.98
	Ratio of peach pixels per image (%)	5.03/5.11	4.92/4.82

## 3.3 Method

### 3.3.1 Model Structure

The proposed model PeachSOLO is designed to simultaneously segment instance masks and predict their categories using full instance mask annotations as supervision. Unlike

<sup>1</sup>The NinePeach dataset is made publicly available at [ninepeach.link](#).

anchor-based segmentation models, which rely on predefined anchor boxes to propose object regions, PeachSOLO is anchor-free. Anchor boxes are randomly generated rectangles used to hypothesise object sizes and locations. Instead of using them, PeachSOLO directly predicts object locations and boundaries.

The architecture of PeachSOLO is presented in Fig. 3.5. The model consists of three parts: a backbone, a feature pyramid network, and a shared detection head following the pipeline from SOLOv2 [277]. The details of these three parts are explained in the following section.

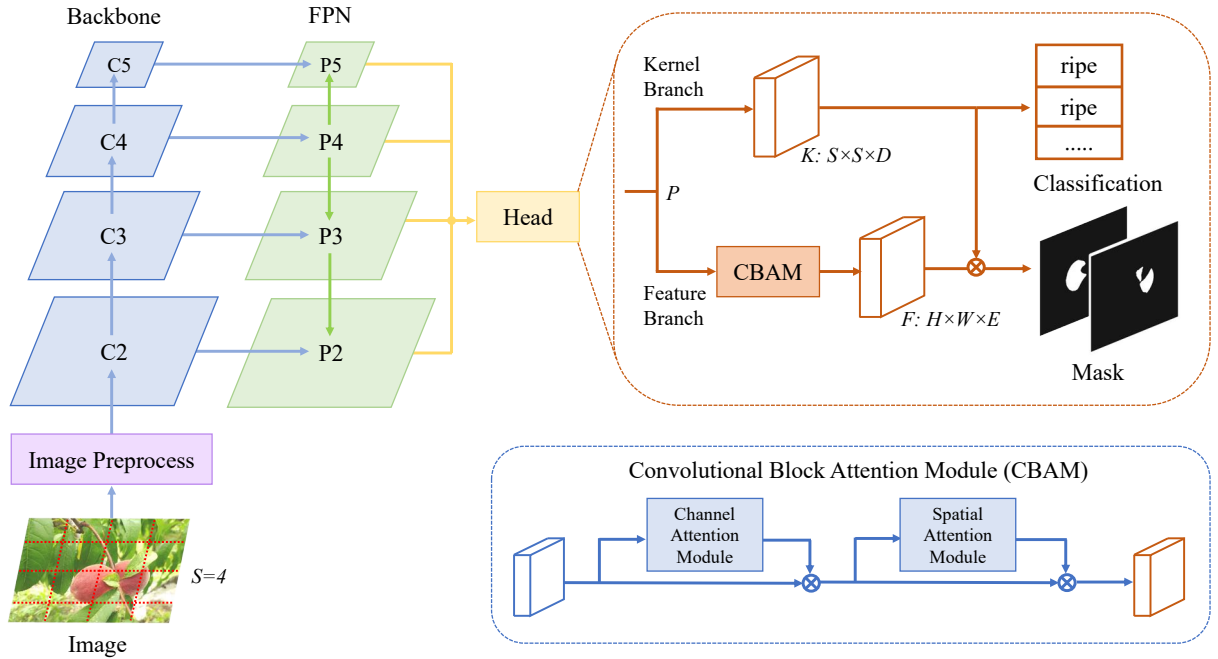


Figure 3.5: The architecture of the proposed PeachSOLO.

In contrast to Mask R-CNN, PeachSOLO does not rely on an RPN to generate proposals, which would reduce much calculation and resource consumption. Specifically, it directly identifies object instances by their centre locations and sizes. To determine object locations, the input image would be divided into a uniform grid of size  $S \times S$ , resulting in  $S^2$  possible centre location classes. If the centre of an object falls within a grid cell, that cell is responsible for predicting the object's semantic category and segmenting its instance. Later in the chapter, it is also demonstrated that PeachSOLO outperforms the original baseline due to the embedding of the convolutional block attention, which enables the model to focus on objects in key channels and spatial locations.

### Feature Extractor

Image feature extraction is the process of identifying and extracting relevant information or features from an image. PeachSOLO employs a feature extractor, which is made up of two parts: a backbone and a Feature Pyramid Network (FPN, 162).

The backbone extracts the low-level features, such as edges and angles, while the high-level features are fed into a classifier to determine the object category. ResNet [106] and Swin Transformer (SwinT, 172) are used as the backbone.

ResNet was proposed to solve the image classification task. It overcame the vanishing gradient problem that can occur in very deep neural networks by introducing residual blocks. The residual blocks allow information to flow directly from earlier layers to later layers without being affected by intermediate layers.

SwinT is a recently proposed neural network for visual recognition tasks that has shown strong performance. It uses a hierarchical architecture where image patches are progressively downsampled to multiple scales. This enables it to capture both local and global features in an image. Additionally, SwinT incorporates a shifted window mechanism that improves the processing of spatially adjacent patches, further enhancing its ability to capture fine-grained details. The output of the backbone is made of a set of feature maps at four different resolutions ( $C2$  to  $C5$ ).

The FPN is introduced to extend the backbone network, which is especially effective for the detection of targets at different scales. FPN works by taking the feature maps produced by the backbone at different levels of the network, and building a feature pyramid that includes high-level features with strong semantics, as well as low-level features with strong spatial information. The final output of the FPN consists of a set of feature maps at four resolutions.

Overall, the ResNet/SwinT with FPN are powerful architectures for image feature extraction, as they leverage the strengths of both ResNet/SwinT and FPN to extract high-level and low-level features from the input image and combine them to accurately detect objects at different scales.

## Detection Head

Given the output of the pyramid network, the detection head consists of two branches: a kernel branch and a feature branch, accepting each pyramid feature as input. The output of the feature pyramid is denoted as  $P$ .

In the kernel branch,  $P$  is resized into a shape of  $S \times S \times C$ , and then a series of 4 convolution layers and a final  $3 \times 3 \times D$  convolution layer are used to produce the kernel  $K \in R^{(S \times S \times D)}$ . It should be noted that in the first of the four convolution layers, two additional input channels are concatenated, which contain pixel coordinates normalised to the range of  $[-1, 1]$  following CoordConv [169]. In each grid, the kernel branch predicts  $D$ -dimensional outputs, which indicate the predicted convolution kernel weights. The final stage of the kernel branch involves the use of two convolution layers to generate predictions for the kernel and category; the last convolution layer used to predict category is a Deformable Convolution Network (DCN, 56). The weights of the



detection head are shared among different levels.

In the mask feature branch,  $P$  is first passed through Convolutional Block Attention Module (CBAM, 286). CBAM introduced attention to design network architecture, which consists of Channel Attention Module and Spatial Attention Module. These two modules use max pooling and average pooling to extract feature information from channels and spatial locations. By connecting Channel Attention Module and Spatial Attention Module, CBAM enables the model to increase its expressive ability, focus on important features and suppress unimportant ones. CBAM does not change the shape of input features, therefore the shape of the output of CBAM remains the same as  $P$ . Then, feature pyramid fusion is applied to learn a unified and high-resolution mask feature representation. This is achieved through multiple stages of convolution layers, group normalisation, ReLU activation, and  $2\times$  bilinear upsampling, and the FPN features ( $P2$  to  $P5$ ) are scaled into  $1/4$  of the original image size. Similar to the use of CoordConv in the kernel branch, normalised coordinates are also concatenated with the FPN feature  $P5$ , enabling the model's position sensitivity. A final  $1\times 1$  convolution layer is applied on scaled features ( $P2$  to  $P5$ ) to generate mask feature  $F \in R^{(H\times W\times E)}$ .

Here, the  $D$  from the kernel branch is set equal to  $E$ , implying that the predicted kernel is for a  $1\times 1$  convolution. After the mask kernel  $K_{i,j}$  from the kernel branch and mask feature  $F$  from the mask branch are obtained, a dynamic convolutional operation is employed to generate the instance mask of  $S^2$  channels corresponding to  $S\times S$  grids. The operation can be written as:

$$M_{i,j} = K_{i,j} * F \quad (3.1)$$

where  $K_{i,j} \in R^{1\times 1\times E}$  is the convolution layer kernel predicted by the kernel branch, and  $M_{i,j} \in R^{1\times H\times W}$  is the mask prediction containing only one instance whose centre is at grid cell  $(i, j)$ . For example, if  $D$  and  $E$  are set equal to 4, the mask branch would generate an output with a shape of  $H\times W\times 4$ . The kernel branch would generate an output with a shape of  $S\times S\times 4$ , which can be viewed as  $S^2$   $1\times 1$  convolution kernels whose depths are 4. The dynamic convolutional operation would use the two outputs above to get the predicted mask. At last, the predicted mask would be post-processed to get the peach instance segmentation results.

### 3.3.2 Model Training

#### Loss Function

In this chapter, the proposed model only generates the predictions of peach categories and peach masks. To simultaneously consider the performance of both predictions, the loss function is designed to consist of two major components: the classification loss  $L_{class}$

and the mask loss  $L_{mask}$ , and  $\lambda$  is the weight factor of mask loss.

$$L = L_{class} + \lambda L_{mask} \quad (3.2)$$

where  $L_{class}$  is the focal loss [163] for semantic category classification and  $L_{mask}$  is the dice loss [256] for mask prediction. The focal loss  $L_{class}$  is calculated as follows:

$$L_{class} = -\alpha(1-p)^\gamma \log(p) \quad (3.3)$$

where  $\alpha$  is set to 0.25 and  $\gamma$  is set to 2.0 in this study.  $p$  is the probability of the predicted instance. A sigmoid operation is used to calculate  $p$ .

The dice loss  $L_{mask}$  is calculated as follows:

$$L_{mask} = 1 - \frac{2 \left| \sum_{x,y} (p_{x,y} \cdot q_{x,y}) \right|}{\sum_{x,y} p_{x,y}^2 + \sum_{x,y} q_{x,y}^2} \quad (3.4)$$

where  $p_{x,y}$  and  $q_{x,y}$  refer to the value of pixel located at  $(x, y)$  in predicted mask  $p$  and ground truth mask  $q$ .

## Training Details

Res50, Res101 and SwinT are used as backbones. For Res50 and Res101 backbone, the batch size is set to 16 with 27K iterations in all, and the initial learning rate is set to 0.005 and divided by 10 at iterations 18K and 24K. For SwinT backbone, the batch size is set to 4 with 54K iterations in all.

The initial learning rate is set to 0.005 and divided by 10 at iterations 36K and 48K. Additionally, PeachSOLO is also trained on nine individual peach datasets separately to validate the generalisation ability of PeachSOLO. Res50 is used as the backbone network, the batch size is set to 16 with 10K iterations, and the initial learning rate is set to 0.005 and divided by 10 at iterations 6K and 8K. An SGD optimiser is used with a weight decay of 0.0001 and a momentum of 0.9. The learning rate is warmed up for the first 1000 iterations, then updated according to the StepLR method.

The backbone is initialised with pre-trained weights on ImageNet [142], and all convolution layers in the detection head are initialised with a normal distribution. The data augmentation strategies used in training contain random horizontal flipping, resizing the input images such that the shortest side is one of 640, 672, 704, 736, 768 or 800 pixels while the longest is at most 1333. The number of grids for four feature map levels is (40, 36, 24, 16).

The loss weights for  $L_{class}$  are set as {unripe:1.0, semiripe:1.5, ripe:2.0} to pay more attention to categories with fewer instances. The  $\lambda$  of the loss function  $L$  is set to 3 during training.

### 3.3.3 Model Inference

#### Evaluation Metrics

- **Average Precision (AP)**

The average precision and average recall are frequently used to measure the performance of segmentation models. The definitions of precision and recall are:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (3.5)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (3.6)$$

where TP is the number of true positive cases that the target is a fruit and is correctly detected, FP is the number of false positive cases that the target is not a fruit, but it is wrongly detected, and FN is the number of false negative cases that the target is a fruit, but it is not detected.

AP is a standard measure for measuring the sensitivity of the network to a target object and is an indicator that reflects the global performance of the network. The higher the AP value, the better the detection accuracy of the proposed model. Following the criterion of MS COCO [164], mean AP is used as the primary metric to evaluate the model performance, which is calculated by averaging 10 Intersection over Union (IoU) thresholds ranging from 0.50 to 0.95 across all categories. Three different sizes of objects are defined based on the instance area: small area ( $\leq 32^2$ ), medium area ( $32^2 \sim 96^2$ ) and large area ( $> 96^2$ ). AP values for IoU thresholds of 0.50, 0.75, and three different object sizes, referred to as  $\text{AP}_{50}$ ,  $\text{AP}_{75}$ ,  $\text{AP}_{\text{small}}$ ,  $\text{AP}_{\text{medium}}$  and  $\text{AP}_{\text{large}}$  are reported,.

- **Learnable Parameters (Params)**

The learnable parameters refer to the weights and biases within the model's layers, which are adjusted during the training process to optimise performance and make accurate predictions. The total number of learnable parameters in a model is often considered an indicator of its capacity and complexity.

- **Floating-point Operations (FLOPs)**

Floating-point operations is a measure of the computational complexity of a deep learning model. It represents the number of arithmetic operations performed by the model during the process of forward propagation, where input data passes through the layers of the model to produce output predictions. FLOPs is typically quantified in terms of the number of multiplications and additions performed by the model.



- **Frames per second (FPS)**

Frames per second is a measure of the model inference speed, which indicates how many input images the model can process per second. A higher FPS reflects faster model execution, which is critical for real-time applications such as autonomous driving or in-field analysis. It is influenced by factors such as model size, hardware performance, and optimisation techniques.

## Inference Details

The data augmentation strategy used in inference is only resizing the input images such that the shortest side is 800 pixels while the longest is at most 1333 pixels.

During the inference, the preprocessed input image would be passed through the backbone network, the feature pyramid network, and the detection head to generate two predictions. The first prediction from the kernel branch includes the predicted category scores and predicted mask kernels, while the second prediction from the feature branch includes predicted mask features.

Then, the predicted mask kernels are utilised to perform a convolution operation on the predicted mask feature to generate predicted soft masks, followed by a sigmoid operation, with the value range being  $[0,1]$ . A threshold of 0.5 is used to convert predicted soft masks to binary masks. It is noted that the final category scores are calculated by pixel-wise multiplication of the predicted category scores with binary masks, followed by division by the count of binary masks. Then, the top 500 predictions are kept, and the redundant predicted masks are removed via Non-Maximum Suppression (NMS).

Finally, the predicted masks are reshaped and interpolated to the original image size.

## 3.4 Experiments and Results

### 3.4.1 Experiments

In this chapter, the experiments are based on Detectron2 [288] and have been carried out using Python 3.9.13 and PyTorch 1.13 on a computer with the specifications shown in Table 3.4.

It is demonstrated that the proposed PeachSOLO achieves competitive results compared to Mask R-CNN on the NinePeach dataset. A detailed ablation study of the detection head and class loss weights is also provided. To check how well the model adapts, the model is trained separately on individual peach datasets. Lastly, the segmentation results are visually shown, and the computational details are calculated.

Table 3.4: The computer specifications.

Aspect	Details
System OS	Cent OS 7
CPU	Inter Xeon Gold 6152 @2.1 GHz
GPU	2 × Nvidia Tesla V100
Memory	32.0 GB

### 3.4.2 Main Results

PeachSOLO and state-of-the-art Mask R-CNN are trained using the NinePeach dataset, and their instance segmentation performance is then compared. Fig. 3.6 illustrates the training loss and periodic evaluation (14 checkpoints for Res50/101 and 6 checkpoints for SwinT) results of the proposed model with different backbones, with the losses converged and evaluation results stabilised at the end of the training. The results are presented in Table 3.5.

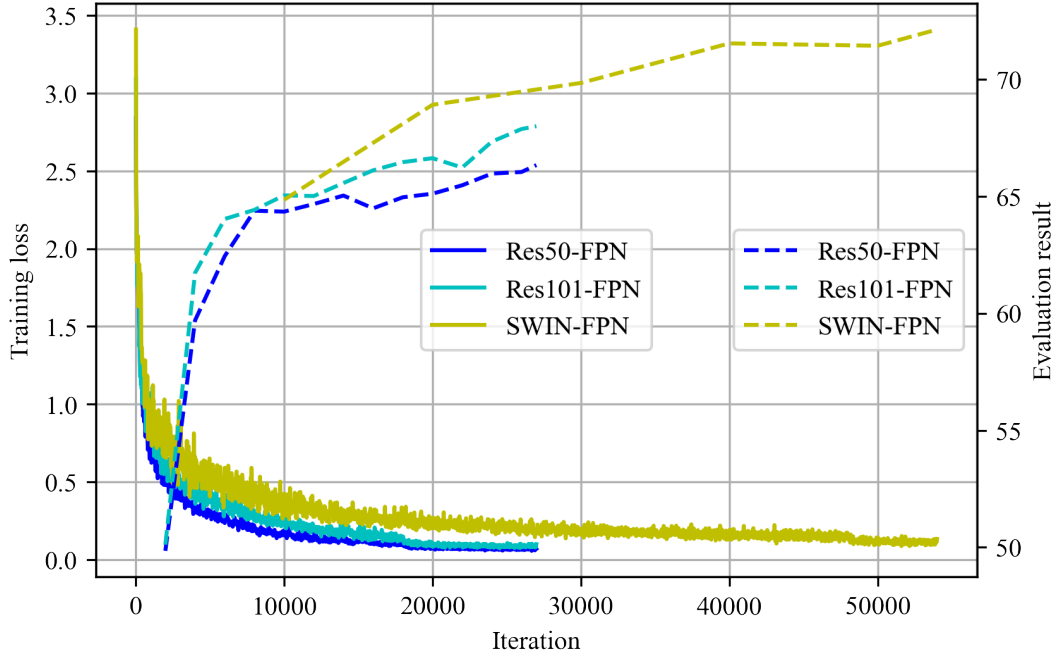


Figure 3.6: The training loss (solid line, left ordinate) and evaluation AP (dash line, right ordinate) of the proposed PeachSOLO.

PeachSOLO with a SwinT backbone achieves the highest AP of 72.12 in all experiments. Besides, PeachSOLO outperforms Mask R-CNN on overall AP when using the same backbone.

First, with increasing backbone complexity and capacity, performance gains are progressively enhanced. For example, PeachSOLO increases about 1.66 and 5.79 AP when changing Res50 to Res101 and SwinT. This observation means the FPN and the detection

head need more representative features generated by a stronger backbone as a condition for segmentation.

Second, PeachSOLO has a lower  $AP_{75}$  and a higher  $AP_{50}$  than Mask R-CNN, which indicates that PeachSOLO is stricter when outputting predictions. This suggests a slight confidence reduction of PeachSOLO, which is caused by the pixel-wise calculation on predicted category scores with binary masks in the inference phase. Mask R-CNN first proposes a bounding box and then generates a high-quality mask within that box, might allow for finer mask refinement and thus better performance at stricter IoU thresholds.

Table 3.5: Instance segmentation results on the NinePeach validation set.

Model	Backbone	AP	$AP_{50}$	$AP_{75}$	$AP_{\text{small}}$	$AP_{\text{medium}}$	$AP_{\text{large}}$	$AP_{\text{unripe}}$	$AP_{\text{semiripe}}$	$AP_{\text{ripe}}$
Mask R-CNN	Res50-FPN	65.02	75.53	70.93	17.47	36.10	77.31	63.98	59.65	71.43
	Res101-FPN	66.02	75.91	71.76	13.01	33.93	78.29	64.92	60.25	72.99
	Swin-FPN	69.91	83.11	<b>76.26</b>	<b>24.26</b>	<b>45.81</b>	76.47	64.57	64.09	76.08
PeachSOLO	Res50-FPN	66.33	78.59	68.95	13.92	32.03	76.57	65.21	61.93	71.86
	Res101-FPN	67.99	77.73	70.84	11.21	32.41	78.75	65.76	62.39	75.84
	SwinT-FPN	<b>72.12</b>	<b>83.76</b>	75.49	11.52	40.25	<b>82.19</b>	<b>68.24</b>	<b>69.26</b>	<b>78.87</b>

Third, compared to Mask R-CNN which has relatively higher  $AP_{\text{small}}$  and  $AP_{\text{medium}}$ , PeachSOLO tends to have better performance in predicting large peach instances, which is similar to related work [304]. Large object detection of PeachSOLO benefits from the mask feature fusion in the mask feature branch, which fuses features of different scales to get a unified and high-resolution feature representation. Mask R-CNN employs a RPN that generates numerous anchor boxes at different scales and aspect ratios. This systematic generation and refinement of proposals can be effective at localising smaller and medium objects.

Finally, PeachSOLO outperforms Mask R-CNN on every ripeness AP, which demonstrates that it has better segmentation performance. Notably, both models are relatively good at predicting ripe peach instances. It is suggested that the complexity of segmenting ripe instances is reduced because the ripe peach not only has a conspicuous colour to distinguish, but also appears alone usually as a result of naturally falling fruit and artificial fruit thinning.

### 3.4.3 Ablation Results

A series of ablations is conducted to investigate the impact of different components in the detection head and different loss weights in the loss function on segmentation performance.

### Detection Head

The detection head plays a critical role in the proposed model. Table 3.6 shows the ablation results of the components of the detection head. Compared to the vanilla baseline, adding coordinates and replacing the last convolution with a deformable convolution attains 2.11 and 2.47 AP gains. Besides, adding the CBAM for stronger spatial perception ability gives a significant 4.55 AP improvement. The proposed model leverages the above three components and improves the baseline by 5.66 AP.

Table 3.6: Ablation on different components of the detection head.

Model	AP	AP <sub>75</sub>	AP <sub>50</sub>	AP <sub>unripe</sub>	AP <sub>semiripe</sub>	AP <sub>ripe</sub>
Vanilla	59.81	77.00	62.48	53.52	58.03	67.88
+Coord	61.92	76.22	64.54	57.63	58.57	69.57
+DCN	62.28	74.56	63.83	60.39	57.42	69.04
+CBAM	64.36	76.57	66.60	61.91	61.32	69.84
=PeachSOLO	<b>65.47</b>	<b>77.29</b>	<b>68.13</b>	<b>63.42</b>	<b>62.31</b>	<b>70.67</b>

### Class Loss Weights

As the category distribution of the dataset is imbalanced, different weights for different categories are needed to reduce the imbalance. Table 3.7 shows some ablation results on different loss weights set for three categories. The loss weight settings {unripe:1.00, semiripe:1.50, ripe:2.00} demonstrated the best performance, emphasising the importance of specific losses and thereby enhancing model performance. However, overly imbalanced weight setting {1.00 : 2.00 : 3.00}, which pays much more attention to semiripe and ripe instances, deteriorates model performance.

Table 3.7: Ablation on different weights for class loss.

Weights	AP	AP <sub>75</sub>	AP <sub>50</sub>	AP <sub>unripe</sub>	AP <sub>semiripe</sub>	AP <sub>ripe</sub>
1.00 : 1.00 : 1.00	65.02	77.86	67.53	63.94	59.97	71.18
1.00 : 1.25 : 1.75	63.59	75.79	65.70	61.64	59.47	69.66
1.00 : 1.50 : 2.00	<b>66.33</b>	<b>78.59</b>	<b>68.95</b>	<b>65.21</b>	<b>61.93</b>	<b>71.85</b>
1.00 : 1.75 : 2.25	65.33	77.64	67.71	63.89	60.49	71.62
1.00 : 2.00 : 3.00	63.06	74.14	65.60	60.75	58.10	70.34

### 3.4.4 Combined and Separate Training

The segmentation performance of training PeachSOLO with Res50-FPN on separate (t/o separate) and combined (t/o combined) NinePeach dataset are compared. The results are shown in Table 3.8. The evaluation results of the model trained using NinePeach perform better compared to those of models trained using individual peach datasets. The average AP improvement stands at 21.05, with Songsen showing the most significant enhancement at 36.61. It is suggested that after merging the datasets, not only does the distribution of peach categories become more balanced, but the model also has more data samples for learning the characteristics and patterns of peaches in different ripeness stages, thus improving the generalisation ability.

Table 3.8: The comparison of training on separate and combined NinePeach dataset.

Peach	t/o separate			t/o combined		
	AP	AP <sub>75</sub>	AP <sub>50</sub>	AP	AP <sub>75</sub>	AP <sub>50</sub>
1.Dahongpao	39.52	57.74	38.44	57.48	69.92	59.69
2.Qingfeng	48.96	66.73	48.66	<b>76.77</b>	84.30	<b>79.84</b>
3.Chunmei	37.92	56.41	38.19	72.78	82.70	77.12
4.Chunmi	46.55	59.38	45.74	49.38	56.64	51.07
5.Chunxue	50.81	<b>68.38</b>	52.68	73.87	<b>82.98</b>	79.30
6.Songsen	32.60	59.77	31.54	69.21	76.17	74.75
7.Maotao	46.06	62.08	46.65	53.90	61.13	55.69
8.Youpantao	38.34	57.47	37.13	61.71	73.62	64.84
9.XiahuiNo5	<b>54.59</b>	70.33	<b>55.32</b>	69.76	78.52	72.02
Average	43.93	62.03	43.82	64.98	74.00	68.26

### 3.4.5 Visualisation

The peach segmentation performance of PeachSOLO is visualised in Fig.3.7. As shown in Fig. 3.7 (left), besides the easy cases when the peaches are fully visible and can be segmented accurately, PeachSOLO is capable of detecting peaches in more complex cases. Specifically, when the peaches overlap with each other or are partially obscured by tree branches or leaves, PeachSOLO still performs well in identifying them accurately. The good segmentation performance shows the feasibility of the dynamic convolution operation in the detection head, of which two operators are mask features and mask kernels that are both learned from the output of the feature pyramid network.

It is worth noting that PeachSOLO not only detects multiple peaches of varying sizes

within a single image accurately, but also generates almost as smooth boundaries as the ground truths, benefitting from fused and high-resolution mask feature representation after CBAM operation. Fusing features of different scales that merge the information of peaches of varied sizes to a unified feature enables the model to make predictions of varying sizes at the same time. The high-resolution mask feature brings larger predicted masks, which means negligible loss when reshaping them back to their original sizes.



Figure 3.7: Segmentation visualisations of PeachSOLO, with accurate (left) and non-accurate (right) examples.

However, there are a few cases where called wrong prediction and missing prediction output by PeachSOLO as shown in Fig. 3.7 (right). Wrong prediction means that two or more objects are wrongly predicted to be one object, or a part of the background is wrongly predicted as a peach. It is assumed that a wrong prediction occurs when some parts in the image have similar features to each other or with known category features, which makes the model regard them as the same object or target objects. On the other hand, if peaches are too obscured to be discovered or look like the background because of misleading light conditions, the model tends to ignore them or treat them as the background, resulting in the problem of missing prediction in these scenarios.

Furthermore, the peach segmentation performance between PeachSOLO and Mask R-CNN is compared in Fig. 3.8. The red and blue boxes are used to emphasise the difference. In the case Fig. 3.8 (left), Mask R-CNN ambiguously predicts the leaf as a part of the peach, while PeachSOLO can segment the peach without the leaf clearly. It can be observed that PeachSOLO produces more precise and smoother boundary predictions than Mask R-CNN. Fig. 3.8 (middle) shows a challenging case where a peach is occluded by leaves and stalks at the same time. PeachSOLO segments the peach almost perfectly; it accurately detects the peach in most of the regions, especially those along the tricky boundaries, while Mask R-CNN cannot clearly segment the boundaries between the peach and leaves and stalks, producing much more inaccurate and incomplete predictions. In



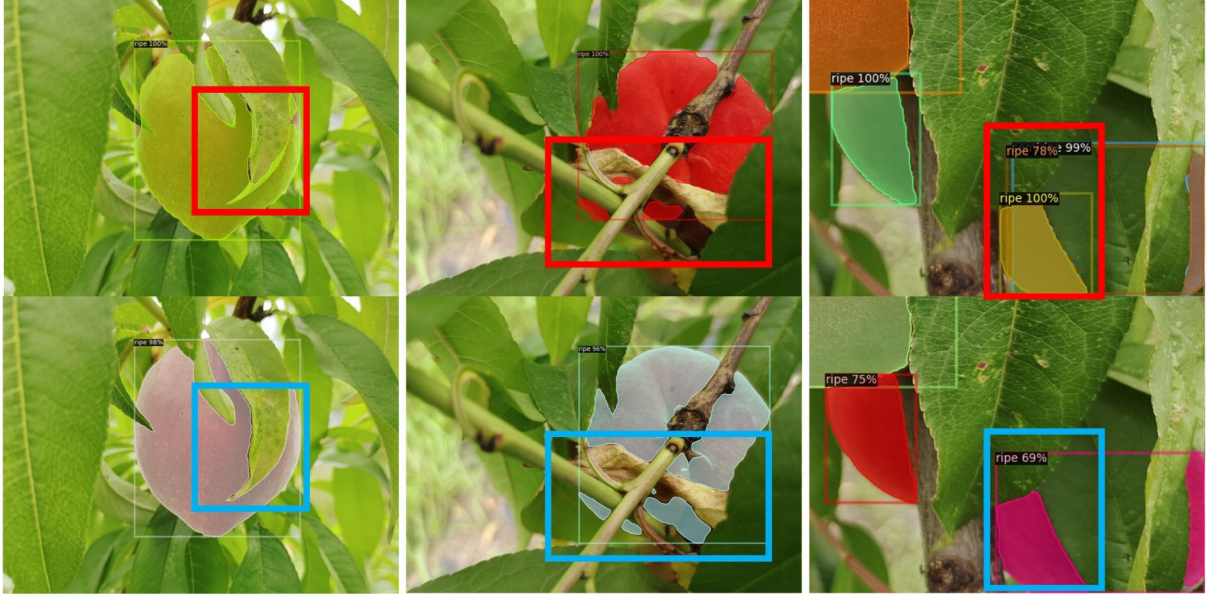


Figure 3.8: The segmentation comparison of Mask R-CNN (top) and the proposed PeachSOLO (below).

Fig. 3.8 (right), Mask R-CNN predicts one peach separated by a leaf as two individual peaches, whilst PeachSOLO predicts the separated parts as one object.

### 3.4.6 Model Complexity

The number of learnable parameters, FLOPs, FPS and maximum GPU memory usage during training between PeachSOLO and Mask R-CNN using the same backbone Res50-FPN are compared. The results are shown in Table 3.9.

Table 3.9: The complexity comparison of PeachSOLO and Mask R-CNN.

Model	Params (M)	FLOPs (G)	FPS	Max Mem. (M)
Mask R-CNN	<b>43.93</b>	<b>174.9±1.0</b>	8.33	11135
PeachSOLO	46.17	213.4±0.2	<b>11.11</b>	<b>8542</b>

PeachSOLO has 2.24M more parameters than Mask R-CNN, most of which are introduced by convolution layers. This means that PeachSOLO is more complex and requires more data for training. PeachSOLO has more 38.5G FLOPs than Mask R-CNN, showing that PeachSOLO has higher computational complexity. PeachSOLO runs 25% faster than Mask R-CNN during inference, which indicates that PeachSOLO is relatively faster to execute. PeachSOLO saves 2593M GPU memory than Mask R-CNN, which makes it hardware-friendly to be trained on different devices.

Although PeachSOLO has more parameters and FLOPs than Mask R-CNN, it benefits from a one-stage design, which avoids the region proposal step and reduces overall



processing time. Moreover, PeachSOLO shares its detection head across feature levels, enabling more efficient multi-scale processing. Operations such as DCN are also well supported by the underlying framework and can be accelerated through GPU parallel computing. These design and implementation choices contribute to the faster inference speed compared to Mask R-CNN.

In summary, Mask R-CNN has fewer parameters and FLOPs but longer inference time and more GPU memory usage as a result of the abundant anchors generated during training and inference. Despite having more Params and FLOPs, PeachSOLO manages to keep inference time and GPU usage relatively low. It maintains better accuracy and precision than Mask R-CNN while delivering results faster. PeachSOLO is able to perform a larger number of FLOPs quickly, striking a fine trade-off between performance and complexity. This efficiency can be attributed to the detection head that is anchor-free and shared between different feature map levels, which allows PeachSOLO to maximise computational power while minimising memory requirements and enables it to be potentially deployed on GPUs with limited memory capacities.

## 3.5 Discussion

### 3.5.1 The Details of NinePeach

According to the current state of the literature, there is no official standard for classifying the ripeness of peaches on trees. With the cooperation of a botanist specialising in peaches, the peach ripeness is determined into three stages subjectively. The only criterion is that annotators must choose their first judgment when meeting ambiguous cases. Similar to other large datasets, the NinePeach dataset also has a long-tail phenomenon, which refers to a situation where few categories have a high frequency of occurrence, while most categories have relatively few instances, forming a “long tail” in the distribution curve. The images were additionally oversampled to increase the number of instances of fewer categories and set different weights for different categories to alleviate this problem. The improved dataset has a balanced category distribution, facilitating the training of a large and well-performing peach instance segmentation model.

### 3.5.2 Limitations

PeachSOLO demonstrates accurate peach detection capabilities, even when peaches are obstructed by tree branches or leaves. However, in a few cases where certain regions within the image exhibit similar features to each other or with known category features, it may generate false predictions, or missing predictions occurring when peaches are too obscured or look like the background due to lighting conditions. These unreliable predictions were

attributed to the larger receptive field of PeachSOLO and the misleading illumination conditions of the image.

The incorporation of CBAM has led to a noteworthy 4.55 point increase in AP, but it has also augmented the complexity of the model, with the extensive use of convolution operations resulting in an elevation of both learnable parameters and floating-point operations. The potential improvement direction of PeachSOLO is to reduce unreliable predictions and to reduce computational complexity.

### 3.6 Summary

Precise identification of the peach ripeness stage plays a crucial role in developing automated harvesting systems for large peach orchards, as it enhances picking efficiency and reduces production costs. Motivated by this, a high-quality peach dataset called NinePeach and a one-stage peach instance segmentation model were constructed in this chapter.

The NinePeach dataset comprises a total of 4599 peach images, categorised into three distinct stages of ripeness: unripe, semiripe, and ripe. This dataset aims to reproduce the natural field conditions, including images with factors like different intensities of natural light, multi-fruit adhesion, and occlusion caused by stems and leaves.

The proposed one-stage peach instance segmentation model does not require an RPN to generate bounding box proposals. The prediction of masks is obtained through dynamic convolution operations on the mask feature and kernel feature output from two branches. Channel attention and spatial attention are considered to enhance the ability to detect objects in key channels and spatial locations, which brings a significant positive impact on model performance. Benefitting from the anchor-free and memory-friendly design, the proposed model achieves a delicate balance between model performance and complexity, manifested by the fact that it utilises fewer GPU resources while delivering faster and better predictions compared to Mask R-CNN.

At present, the released large peach dataset provides a foundation for further peach-related studies and reduces their workload. The proposed model can accurately detect peaches and generate their smooth boundaries, even in some cases where peaches are occluded, which establishes a robust basis for further work, like peach picking point estimation and peach disease monitoring. These advances create opportunities for offering practical solutions for farmers, applying this technology to other fruits or crops and considering the ever-evolving nature of agriculture.

# Chapter 4

## LightStraw: Lightweight CNN-based Strawberry Instance Segmentation Models

### 4.1 Introduction

Building upon the instance segmentation framework introduced in the previous chapter, which focused on accurate peach ripeness identification, this chapter explores the development of lightweight CNN models for strawberry instance segmentation. Unlike the previous chapter, the emphasis here is not on fruit maturity classification but on designing segmentation models that are both accurate and computationally efficient, aiming to support real-time deployment in resource-constrained agricultural environments.

Strawberries are a high-value fruit crop that often requires labour-intensive cultivation and harvesting. Automating the segmentation of individual strawberry instances can significantly enhance harvesting efficiency and enable precise yield estimation. However, many state-of-the-art instance segmentation models, such as Mask R-CNN, require extensive computational resources, limiting their practicality in real-world field conditions.

To address this challenge, this chapter introduces a series of CNN-based instance segmentation models, collectively referred to as LightStraw. These models integrate a self-attention-based backbone for enhanced semantic feature extraction, a feature pyramid network for multi-scale representation, and a decoder that incorporates both coordinate-aware features and instance activation maps. A bipartite matching algorithm is used during candidate selection, allowing for efficient instance assignment without the need for sorting or Non-Maximum Suppression.

Experimental results show that the proposed models achieve substantial improvements over original and simplified Mask R-CNN baselines, both in terms of accuracy and efficiency. Specifically, they reduce the number of parameters and floating-point operations

while maintaining high segmentation performance, making them suitable for deployment on edge devices in practical agricultural scenarios.

## 4.2 Dataset

### 4.2.1 StrawDI\_Db1 Overview

The StrawDI\_Db1 dataset [217] comprises 3100 images captured in strawberry plantations at various times throughout a complete picking campaign. These images were taken using a mobile phone, featuring a resolution of  $4032 \times 3024$  pixels, 8 bits per colour channel, and stored in JPEG format. Then, the images have been rescaled to  $1008 \times 756$  pixels in PNG format. These images are organised into training (2800 images), validation (100 images), and testing (200 images) subsets, respectively.

A statistical overview of the dataset is presented in Table 4.1. The distribution of strawberry instances demonstrates a relatively balanced representation across different sizes, with a slightly higher proportion of medium-sized instances in the training set. The number of instances per image and the pixel ratio remain stable across all sets, indicating a consistent frequency of occurrence and an overall average fruit shape. The StrawDI\_Db1 dataset offers instance-level annotations for all sets. Example annotations are illustrated in Fig. 4.1.

Table 4.1: Statistics of the StrawDI\_Db1 dataset.

Aspect	Category	Train	Val	Test
Ratio of size	Small ( $\text{area} \leq 32^2$ )	0.21	0.22	0.22
	Medium ( $32^2 < \text{area} \leq 96^2$ )	0.48	0.44	0.48
	Large ( $\text{area} > 96^2$ )	0.31	0.34	0.30
Mean/standard deviation	Number of strawberry instances	5.8/2.9	5.7/2.7	5.7/2.8
	Ratio of strawberry pixels per image (%)	5.6/2.7	5.7/2.4	5.4/2.5

## 4.3 Method

### 4.3.1 Model Structure

The proposed model is designed to be lightweight and efficient in performing strawberry instance segmentation, of which the architecture is shown in Fig. 4.2. It is constructed by two main parts: an encoder and a decoder. The encoder consists of a backbone and an FPN, which extracts contextual information from images and builds multi-scale features

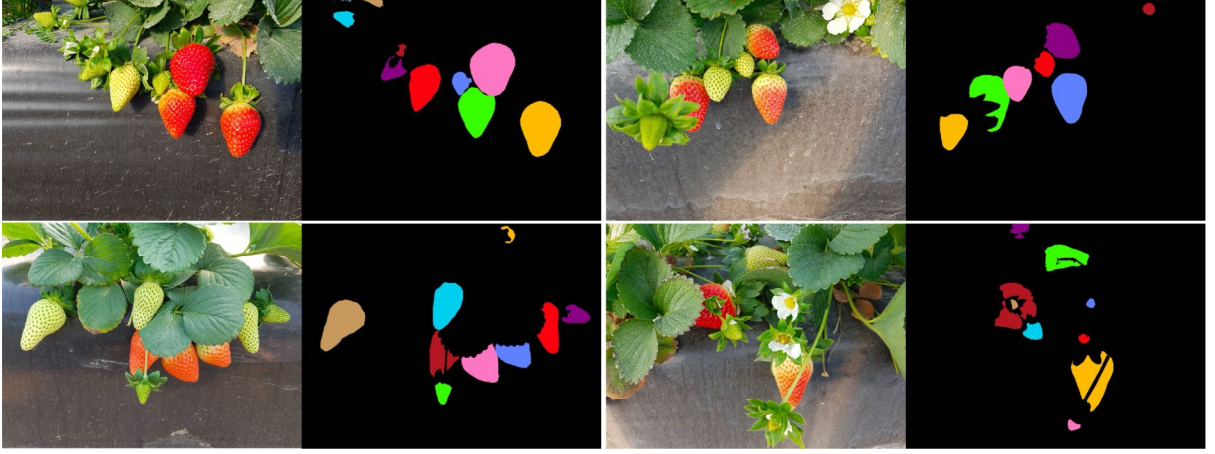


Figure 4.1: Sample images and annotations of StrawDI\_Db1.

for later prediction. The decoder is anchor-free and does not require an RPN to generate anchors; it mainly contains two branches and predicts class and masks directly based on features extracted by the encoder.

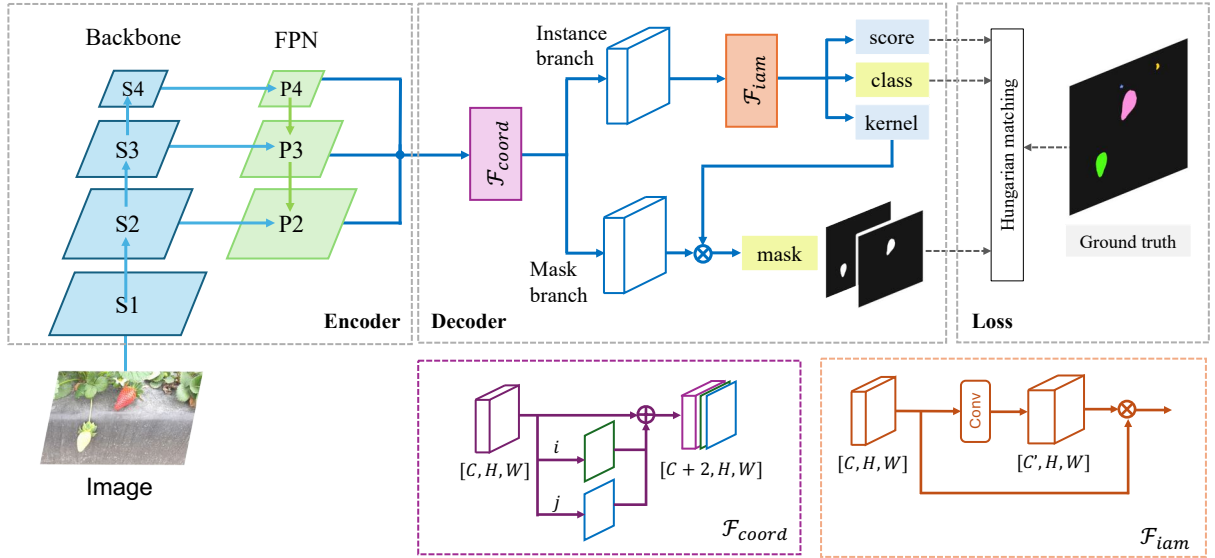


Figure 4.2: The architecture of the proposed LightStraw.

### 4.3.2 Encoder

#### Backbone

Modelling in computer vision has been dominated by CNNs for a long time. On the other hand, the tremendous success of Transformer in the language domain inspired the emergence of Vision Transformer (ViT). Compared with CNNs, ViT offers a powerful approach to capturing global dependencies and contextual understanding in images. The attention mechanism plays a crucial role in capturing relationships between different parts of an image in ViT, enabling ViT to attend to and aggregate information from all image

patches simultaneously. Therefore, instead of directly using CNNs like ResNet as the backbone in the previous chapter, an efficient attention-based backbone is proposed to extract features from input images.

The vanilla self-attention is calculated by Eq. (4.1). Firstly, the input embedding, including positional encoding, is linearly transformed into three sets of vectors: query  $Q$ , key  $K$  and value  $V$ . Then, the attention scores are computed using the scaled dot-product attention mechanism. For each token, its attention to other tokens is determined by the dot product of its query vector with the key vectors of other tokens. Next, the result is scaled by the square root of the dimension of the key  $\sqrt{d_k}$ . The attention scores are normalised using the Softmax function to obtain attention weights. Finally, the value  $V$  are multiplied by the attention weights, and the resulting weighted vectors are summed to produce the attention output.

$$\text{Attention} = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (4.1)$$

The vanilla self-attention in ViT undergoes a sequence of steps that contribute to its progressive and effective modelling of relationships between different parts of an input image. However, the vanilla self-attention has a quadratic time and space complexity with respect to the sequence length, which makes it computationally expensive.

- **Efficient Multi-head Self-Attention (EMSA)** To alleviate the problem, the Efficient Multi-head Self-Attention [312] is adopted as the basic block of the backbone. The detail is shown in Fig. 4.3. Firstly, a set of linear layers is adopted on 2D input token  $[n, d_m]$  to obtain query  $Q$ . Then, the input token is reshaped to 3D one  $[n, h, w]$  and performs a depth-wise convolution operation to reduce its dimensions to  $[n, h/s, w/s]$  by a factor  $s$ . Next, similar to the vanilla self-attention, the attention scores are computed using the scaled dot-product attention mechanism. Before the softmax operation,  $PWConv$  is used to model the interactions among different heads, which is a  $1 \times 1$  pixel-wise convolutional layer, as shown in Eq. (4.2). In the end, the resultant values from each head are concatenated and subjected to a linear projection to create the final output.

$$\text{EMSA} = \text{Softmax}\left(PWConv\left(\frac{QK^T}{\sqrt{d_k}}\right)\right)V \quad (4.2)$$

Self-attention does not inherently understand the order or position of tokens in a sequence. Positional encoding is used to provide this important information. Here, a simple module is used to encode positions in Eq. (4.3). Specifically, a  $3 \times 3$  depth-wise convolution layer is applied to generate pixel-wise weight and then scaled by a sigmoid function  $\sigma(\cdot)$ .

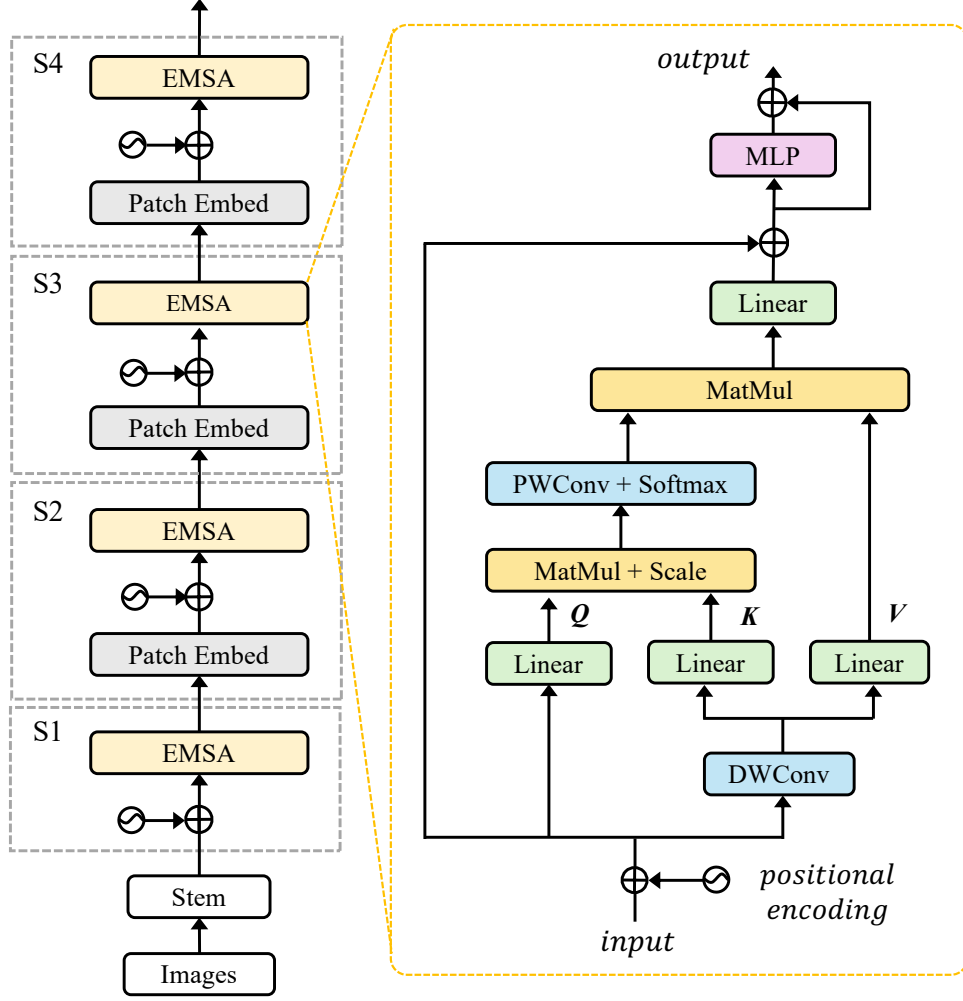


Figure 4.3: The backbone of the proposed LightStraw.

$$PE(x) = x \times \sigma(DWConv(x)) \quad (4.3)$$

- **Patch Embedding** Attention was originally designed for processing sequences of data, to apply it to images, it is necessary to convert the spatial information of the 3D image into a 2D sequence. Here, a stack of three  $3 \times 3$  convolutional layers is used, which is with stride=3/1/2, padding=1/1/1, as shown in Eq. (4.3). Batch Normalisation and ReLU activation are applied sequentially for the first two layers. The first two convolutional layers downsample and adjust channel dimensions, while the third further reduces spatial dimensions and increases output channels. Positional encoding is applied after the third convolutional layer, making it suitable for integration into the attention-based backbone.

To facilitate different scenarios, three different backbone variants (Tiny, Small and Base) are designed. The pipeline of the backbone is shown in Fig. 4.3, and the specification of backbone variants is shown in Table 4.2, of which  $N$  is the number of blocks,  $C$



is the number of embedded dimensions and  $H$  is the number of self-attention heads.

Table 4.2: LightStraw backbone architecture variants.

Stage	Stride	Backbone-T	Backbone-S	Backbone-B
S0	4	$C_0=64$	$C_0=64$	$C_0=96$
S1	4	$N_1/C_1/H_1=2/64/1$	$N_1/C_1/H_1=2/64/1$	$N_1/C_1/H_1=2/96/1$
S2	8	$N_2/C_2/H_2=2/128/2$	$N_2/C_2/H_2=2/128/2$	$N_2/C_2/H_2=2/192/2$
S3	16	$N_3/C_3/H_3=2/256/4$	$N_3/C_3/H_3=6/256/4$	$N_3/C_3/H_3=4/384/4$
S4	32	$N_4/C_4/H_4=2/512/8$	$N_4/C_4/H_4=2/512/8$	$N_4/C_4/H_4=2/768/8$

### Feature Pyramid Network (FPN)

The FPN introduces a top-down architecture where higher-resolution feature maps from earlier stages of the backbone are combined with lower-resolution feature maps from later stages. This is achieved through lateral connections, which involve upsampling the higher-level features and element-wise addition with the lower-level features. The pyramid typically consists of feature maps at different resolutions, different levels represent features at different scales. These scales correspond to different receptive fields and are crucial for handling objects of various sizes. Here, a convolutional layer is applied to aggregate the features of three levels into one at last.

### 4.3.3 Decoder

The simple decoder of SparseInst [42] is adopted to decode the features to predictions, which mainly consists of two branches. Before entering any branch, the feature generated by the encoder passes the CoordConv Module  $F_{coord}$ .

#### The CoordConv Module

The  $F_{coord}$  is implemented as a simple extension of standard convolution [169]. The details of  $F_{coord}$  are shown in Fig. 4.2. Given an input feature  $[C, H, W]$ , two coordinate channels  $i$  and  $j$  with the size of  $[1, H, W]$  are created. Specifically, within  $i$ , the first row is filled with 0, the second row is filled with 1, the third row is filled with 2, etc. The  $j$  channel is similar but with columns filled in with constant values instead of rows. Then, both  $i$  and  $j$  coordinate values are linearly scaled to fall in the range  $[-1, 1]$ . Finally, channels  $i$  and  $j$  are concatenated with the input feature, resulting in an output  $[C + 2, H, W]$ . For convolution over two dimensions, two coordinates  $(i, j)$  are sufficient to specify an input pixel.

This module can be regarded as a type of position embedding, which provides additional location information for later decoding operations. It is noted that two channels are generated by coordinates, thus no extra parameters are introduced, which is friendly to building a lightweight model.

### Instance Branch

This branch consists of an Instance Activation Map (IAM) and three prediction heads. The Instance Activation Map is inspired by Class Activation Map (CAM, 318), which suggests that objects can probably be found in informative regions. The features extracted from the highlighted areas are rich in semantic information and exhibit instance awareness, aiding in the recognition and differentiation of strawberries.

The details of  $F_{iam}$  are shown in Fig. 4.2. The  $F_{iam}$  is a  $3 \times 3$  convolutional layer with 4 groups to aggregate instance features by concatenating features from a group. Given an input feature  $[C, H, W]$ , the output computed by the  $F_{iam}$  is with the shape  $[C', H, W]$ , in which  $C'$  is the pre-set number of instance activation maps. Then, the sparse instance features can be calculated by multiplying the output (normalised to 1) and the transposed input. Finally, the sparse instance-aware features are forwarded to three prediction heads to predict score, class and kernel.

### Mask Branch

As validated by similar work SOLOv2 [277], it is feasible to use trained parameters as the kernel to perform mask prediction. Given the feature generated by FPN and the instance-aware mask kernels generated by the instance branch, the segmentation mask for each instance can be produced by  $m_i = w_i \cdot M$ , where  $m_i$  is the  $i$ -th predicted mask and corresponding kernel  $w_i$ , and  $M$  is the features. The final segmentation masks adopt bilinear interpolation to upsample to the original resolution.

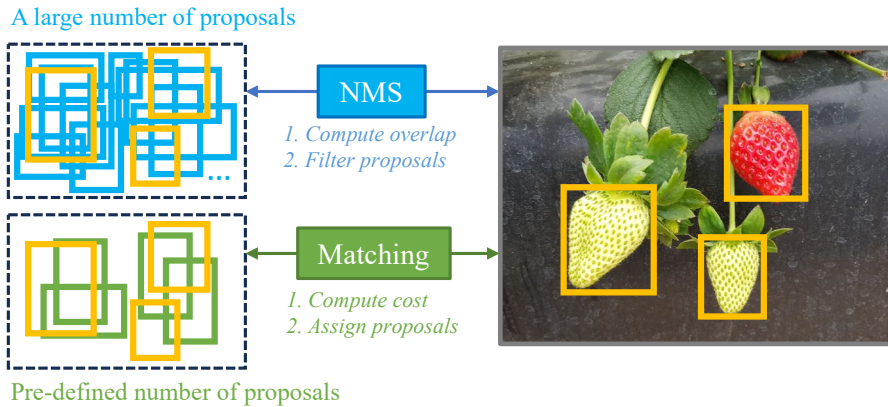


Figure 4.4: Non-maximum suppression and bipartite matching.

#### 4.3.4 Loss Function

Different from anchor-based segmentation models that generate a large number of anchors, the proposed model employs the Transformer decoder to treat fruit detection as an end-to-end dictionary lookup task. Specifically, the decoder generates a fixed number of  $N$  predictions by decoding the  $N$  learnable query embeddings layer by layer. Therefore, the necessity for manual processes like NMS is eliminated. Instead, the Hungarian matching is adopted, which is a kind of bipartite matching method, to find the best matching between predictions and ground truths for loss computation.

The difference between NMS and bipartite matching is illustrated in Fig. 4.4. NMS generates a large number of proposals and applies heuristic filtering based on overlap scores to remove redundant detections. This introduces a non-differentiable post-processing step. In contrast, bipartite matching employs a pre-defined fixed number of proposals and assigns each proposal to a specific ground truth or a "no object" class based on a cost matrix. By integrating this matching process directly into the optimisation framework, the proposed model enables a fully end-to-end differentiable pipeline where predictions and assignments are jointly optimised.

This model follows DETR [26], which treats the label assignment problem as a bipartite matching problem. Firstly, a pairwise dice-based matching score  $C(i, k)$  for the  $i$ -th prediction and the  $k$ -th ground-truth object is introduced in Eq. (4.4), which is determined by classification scores and dice coefficients of segmentation masks.

$$C(i, k) = p_{i, c_k}^{1-\alpha} \cdot \text{Dice}(m_i, t_k)^\alpha \quad (4.4)$$

where  $\alpha$  is a weight for two predictions {segmentation=0.8, classification=0.2},  $c_k$  is the category label for the  $k$ -th ground-truth target and  $p_{i, c_k}$  is the probability for the category  $c_k$  of  $i$ -th prediction. The Dice loss is defined in Eq. (3.4).

First, all of the predictions, including class predictions, mask predictions and class targets, mask targets, are used to calculate a cost matrix for prediction selection, where  $X$  indicates the number of instances in a batch. The class cost and mask cost are calculated by cross-entropy loss and Dice loss, respectively, as shown in Eq. (4.6) and Eq. (3.4).

Second, the Hungarian algorithm is used to search for the best bipartite matching by solving the cost matrix, resulting in a matching score  $C(i, k)$  for the  $i$ -th prediction and the  $k$ -th ground truth object. Therefore, the number of predictions is decreased from  $N$  to match that of the targets.

The training loss is defined in Eq. (4.5):

$$\mathcal{L} = \lambda_s \cdot \mathcal{L}_s + \lambda_c \cdot \mathcal{L}_{cls} + \mathcal{L}_{mask} \quad (4.5)$$

$\lambda$  indicates the different loss weights.  $\lambda_c$  and  $\lambda_s$  are loss weights for classification and

score.

$$\mathcal{L}_s = -\sum_i y_i \cdot \log(p_i) \quad (4.6)$$

$\mathcal{L}_s$  is the binary cross-entropy loss for score, as defined in Eq. (4.6), where  $y_i$  represents the ground truth probability and  $p_i$  represents the predicted probability.

$\mathcal{L}_{cls}$  is focal loss for classification, as defined in Eq. (3.3).

$\mathcal{L}_{mask}$  is the dice loss for mask, as defined in Eq. (3.4).

The evaluation metrics include AP, Params, FLOPs and FPS, as defined in Section 3.3.3.

## 4.4 Experiments and Results

### 4.4.1 Experiments

#### Implementation Details

The experiments are conducted on Detectron2 using Python 3.9.13 and PyTorch 1.13 on a computer with the specifications shown in Table 3.4.

During training, the batch size is set to 16 with 27K iterations in all, an AdamW optimiser is used, and the initial learning rate is set to 0.005 and divided by 10 at iterations 18K and 24K. No pre-trained weights are used, and the parameters of the backbone are initialised by a normal distribution. The training data augmentation strategy contains random horizontal flips, resizing the input images such that the short edge is one of 416, 448, 480, 512, 544, 576, 608 or 640 pixels while the longest is at most 853.

During inference, batch normalisation uses the running averages of mean and standard deviation computed during training, and dropout layers are deactivated during evaluation, so all layers in the model are used. The data augmentation strategy is only the resizing of input images such that the shortest edge is 640 pixels while the longest is at most 853.

### 4.4.2 Results

#### Main Results

Mask R-CNN is a state-of-the-art instance segmentation model that has been applied to strawberry images. An original [304] and a simplified version [217] of Mask R-CNN have been used to perform strawberry instance segmentation. Additionally, a fully convolutional neural network (FCN, 216) has been proposed to solve the same task more efficiently.

Table 4.3: Instance segmentation results on the StrawDI\_Db1 testing set.

Model	Backbone	AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>s</sub>	AP <sub>m</sub>	AP <sub>l</sub>
Mask R-CNN [304]	Res50	45.36	76.57	47.09	07.35	50.03	78.03
Mask R-CNN' [217]	Res50	43.85	74.24	45.13	07.54	51.77	75.90
FCNN [216]	Res50	52.61	69.24	57.84	16.96	65.26	53.31
LightStraw	Tiny	66.82	85.99	71.78	28.53	70.25	87.67
	Small	69.39	87.32	73.96	30.04	71.85	92.15
	Base	<b>70.22</b>	<b>87.70</b>	<b>76.05</b>	<b>31.44</b>	<b>73.63</b>	<b>90.29</b>

s: small ( $\text{area} \leq 32^2$ ); m: medium ( $32^2 < \text{area} \leq 96^2$ ); l: large ( $\text{area} > 96^2$ ).

Table 4.3 summarises the results of other models and LightStraw on the StrawDI\_Db1 testing set. As shown, LightStraw with a Base backbone achieves the highest AP of 70.22.

First, all of LightStraw demonstrates significant improvement over previous work; even the lowest performer with a Tiny backbone has an AP 21.46, 22.97, and 14.21 higher than the original, simplified Mask R-CNN and FCNN, respectively.

Second, model performance gains are progressively enhanced as backbone complexity and capacity increase. For example, LightStraw with Small and Base backbones deliver 2.57 and 3.4 points higher AP than the Tiny backbone. It is assumed that more features can be provided by more layers and bigger embedded dimensions, which helps locate the targets.

Third, the AP<sub>50</sub> of LightStraw is larger than AP<sub>75</sub>, and of which gaps between AP<sub>50</sub> and AP<sub>75</sub> are narrower than the original and simplified Mask R-CNNs, indicating that LightStraw usually output high-accurate results regardless of different IoU criteria. Finally, LightStraw demonstrates better performance when dealing with medium and large strawberries than small strawberries. It is suggested that the reasons for this could be that small strawberries have a similar colour to leaves and are normally covered, and they can be lost when resizing the input images to smaller ones.

Table 4.4: Params and FLOPs of the models.

Model	Backbone	Params (M)	FLOPs (G)
Mask R-CNN [288]	Res50	35.08	877.4
LightStraw	Tiny	<b>17.42</b>	<b>78.3</b>
	Small	20.58	86.9
	Base	33.54	111.7

### Model Complexity

To measure the model size and complexity, the number of learnable parameters and the number of floating-point operations during training are computed. Previous work [304, 217, 216] did not provide information about their model size and complexity, therefore, the complexity between Mask R-CNN with Res50 from Detectron2 and the proposed model is compared. The results are shown in the Table 4.4.

All of LightStraw have fewer parameters compared to Mask R-CNN, among which the Tiny backbone has less than half the number of parameters compared to Mask R-CNN. This indicates that the design of LightStraw is lightweight and efficient, making the models suitable for resource-constrained environments.

LightStraw generally requires significantly fewer FLOPs for each image during inference compared to Mask R-CNN. This suggests that the proposed models are computationally efficient.

In summary, LightStraw not only has fewer parameters and requires fewer floating-point operations during inference compared to the Mask R-CNN but also demonstrates a trade-off between model complexity and computational efficiency, which offers options for scenarios with strict resource constraints. The models with Tiny and Small backbones offer lightweight options for scenarios with strict resource constraints, while the Base backbone provides a higher-capacity variant for tasks that demand more accurate strawberry segmentation.



Figure 4.5: The segmentation visualisation of the proposed LightStraw.



### 4.4.3 Visualisation

The performance of LightStraw is visualised in Fig. 4.5. The model can segment strawberries under various conditions. There are some difficult cases in which strawberries are located at the side of the image or are partially covered, however, the proposed LightStraw can segment them accurately.

Table 4.5: The model FPS across different devices.

Device	Format	LightStraw (Tiny)	Mask R-CNN
NVIDIA Tesla V100	.pth	12.29	8.00
NVIDIA Jetson Orin Nano	.pth	<b>3.26</b>	<b>2.92</b>
	.onnx	0.06	/*
	.trt	1.40	/*
Apple M1	.onnx	1.21	/*

\* Currently Mask R-CNN from Detectron2 is not supported in ONNX.

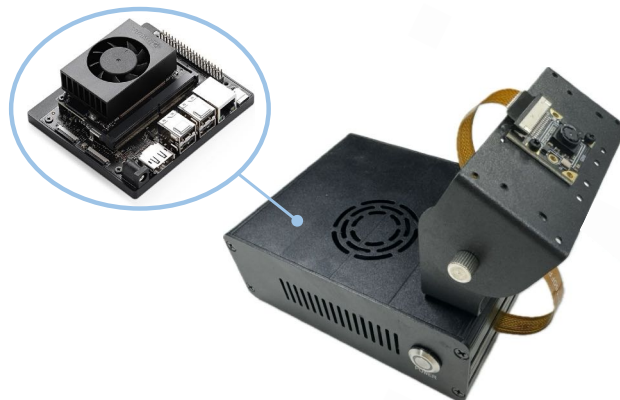


Figure 4.6: NVIDIA Jetson Orin Nano.

### 4.4.4 Deployment

The inference performance of the proposed Tiny LightStraw was compared with Mask R-CNN on heterogeneous hardware platforms, including a high-performance GPU (NVIDIA Tesla V100), an edge computing device (NVIDIA Jetson Orin Nano, as shown in Fig. 4.6), and a general-purpose CPU (Apple M1). Table 4.5 summarises the inference time per image using three model formats: PyTorch checkpoint (.pth), Open Neural Network Exchange (.onnx), and TensorRT engine (.trt).

Each model format serves distinct deployment needs. The .pth format is native to the PyTorch framework and commonly used during model development and training. The ONNX format enables interoperability across platforms and inference engines, allowing



deployment beyond the original training framework [49]. TensorRT is a platform-specific optimisation for NVIDIA hardware, offering accelerated inference through kernel fusion, precision calibration, and runtime tuning [50].

Due to current limitations, Mask R-CNN implemented with Detectron2 cannot be exported to ONNX, and hence its inference is reported only for the .pth format on supported devices.

Experimental results show that Tiny LightStraw consistently achieves lower inference latency compared to Mask R-CNN under the same conditions. On the Tesla V100 GPU, LightStraw reaches a 12.29 FPS, substantially faster than Mask R-CNN’s 8 FPS. On the Jetson Orin Nano, LightStraw demonstrates flexible deployment performance: 1.40 FPS using TensorRT, 0.06 FPS with ONNX, and 3.26 FPS with the PyTorch model. These results confirm its suitability for low-power and edge scenarios.

On the Apple M1, LightStraw (onnx format) achieves a 1.21 FPS. Although slower than optimised deployment on the Jetson platform, this result highlights the model’s functional portability across non-specialised hardware.

In summary, the proposed model exhibits strong deployment versatility and computational efficiency. It supports multiple formats and achieves real-time or near real-time inference across a range of devices, from data centre GPUs to edge and consumer-grade processors, making it well-suited for practical applications.

## 4.5 Discussion

### 4.5.1 Limitations

While LightStraw shows good efficiency and accuracy in strawberry instance segmentation, its main drawbacks come from its specific focus on segmenting one type of fruit without judging its ripeness. Ripeness information should be considered for selective harvesting. The models, being mainly CNN-based, may not fully leverage the global contextual understanding that Transformers might offer for more complex multi-fruit scenarios. Also, its performance is shown using a specific strawberry dataset, and this chapter does not cover how well it can be directly used for other fruits with different features or in different orchard conditions.

### 4.5.2 Future Work

The limitations found in the LightStraw suggest directions for future research, mainly focused on making the models capable of more complete fruit analysis. An important next step is to include multi-stage ripeness classification and to make the model able to handle multiple fruit types. This would make it more useful in practical farming situations.

At the same time, exploring Transformer-based or combined CNN-Transformer model designs is necessary to potentially improve performance on these more challenging tasks by better understanding the entire image. Continuing to improve lightweight designs will also remain important to make sure these models can be widely used on edge devices with limited processing power.

## 4.6 Summary

Accurately detecting and segmenting each strawberry within real-world production environments is pivotal for the development of automatic strawberry-harvesting robots. This precision enables precise calculation of the number and size of strawberries, providing accurate yield information crucial for agricultural planning and resource optimisation.

In this chapter, lightweight attention-based CNN models for strawberry instance segmentation are presented, named LightStraw. The simple models consist of an encoder (a backbone and an FPN) and a decoder. The proposed backbone is based on efficient self-attention, which introduces several depth-wise and pixel-wise convolutional layers to reduce the computation of vanilla self-attention. The last three of the four feature levels extracted by the backbone are used in the FPN to save memory usage and model size. The decoder mainly contains two branches: an instance branch and a mask branch. A  $F_{coord}$  module is applied before any branch, which provides coordinate information to the features. A  $F_{iam}$  module is added to the instance branch to produce instance activation maps, which aim to highlight the informative regions for each strawberry. The bipartite matching is used in LightStraw to avoid NMS in post-processing.

LightStraw outperforms the original and simplified Mask R-CNN with significant 21.46 and 22.97 AP improvements, respectively, among which the one with Base achieves the highest AP of 70.22. Besides, LightStraw requires many fewer parameters and FLOPs compared to Mask R-CNN. In summary, this study introduces lightweight, efficient, and effective models for strawberry instance segmentation. These models hold promise for deployment on embedded devices with limited computational resources in the future.

## Chapter 5

# FruitQuery: Lightweight Query-based Segmentation Models for In-field Fruit Ripeness Determination

### 5.1 Introduction

Based on the previous chapter focused on improving segmentation efficiency through lightweight convolutional networks for strawberries, this chapter further advances instance segmentation by introducing a Transformer-based architecture capable of handling both fruit type recognition and multi-stage ripeness determination. The objective is to achieve fine-grained segmentation of fruits at different ripeness stages under complex in-field conditions, while maintaining a compact and efficient model design suitable for edge deployment.

Most existing methods in agricultural vision rely on convolutional structures, and although Transformer-based models have shown promise in general computer vision, their application to fruit instance segmentation, particularly for ripeness determination, remains largely unexplored. In addition, datasets that jointly provide instance-level segmentation masks and ripeness labels for multiple fruit types are extremely limited, hindering progress in multi-fruit and multi-stage learning.

To address these limitations, this chapter introduces FruitQuery, a lightweight, query-based instance segmentation model that combines convolutional and Transformer components within an end-to-end framework. FruitQuery leverages a unified fruit dataset combining peaches and strawberries with detailed ripeness annotations, and applies efficient self-attention modules and multi-scale feature fusion to improve segmentation accuracy and generalisation. Unlike many traditional models, it avoids post-processing steps such

as non-maximum suppression by directly decoding instance queries into segmentation masks.

Quantitative evaluations show that FruitQuery surpasses a wide range of state-of-the-art models in both accuracy and model compactness, including several advanced YOLO and Transformer variants. These results demonstrate that query-based instance segmentation provides a promising direction for in-field fruit ripeness assessment, especially when lightweight deployment is required.

## 5.2 Dataset

### 5.2.1 Overview

In this chapter, two public fruit datasets, the NinePeach dataset [315] and the StrawDI\_Db1 dataset [217], are combined to form a unified benchmark for fruit instance segmentation. The sample images are shown in Fig.3.1 and Fig. 4.1. Both datasets provide pixel-wise individual annotation masks for every single fruit shown in the image.

**NinePeach dataset.** This dataset is divided into training (3240 images) and validation (1359 images) subsets, and each peach is categorised into three ripeness stages: unripe, semiripe, and ripe. More details are in Section 3.2.3.

**StrawDI\_Db1 dataset.** This dataset is divided into training (2800 images), validation (100 images) and testing (200 images) subsets. The training and testing sets are used in this chapter. More details are in Section 4.2.1. Unfortunately, the StrawDI\_Db1 dataset only offers class-agnostic annotations for strawberries, with no information provided on ripeness. Therefore, a solution to this problem is presented in the following section.

By merging a tree-fruit (peach) and a berry-fruit (strawberry), the dataset spans diverse canopy structures, occlusion patterns, and background textures. This variety offers a more challenging and comprehensive setting for segmentation models, as they must adapt to different orchard conditions and fruit morphologies.

### 5.2.2 StrawDI\_Db1 Ripeness Annotation

Based on the previous work [15, 260], four ripeness stages are selected to distinguish the strawberries from the StrawDI\_Db1 dataset, with the criterion described in Table 5.1.

To achieve this classification, a simple but effective method is adopted for dividing strawberries into four stages, as illustrated in Fig. 5.1.

First, the strawberry instances are cropped from the original images, and background pixels are filtered, as the contextual information from the background was assumed to introduce noise rather than contribute to the classification accuracy. All strawberry

Table 5.1: Four ripeness stages of strawberry.

Category	Description
rs1 (Green)	Dark green, with relatively small sizes.
rs2 (White)	Expanding, with white colour.
rs3 (Turning)	Below 90% red and not ready to be harvested.
rs4 (Red)	Over 90% red, edible and ready to be harvested.

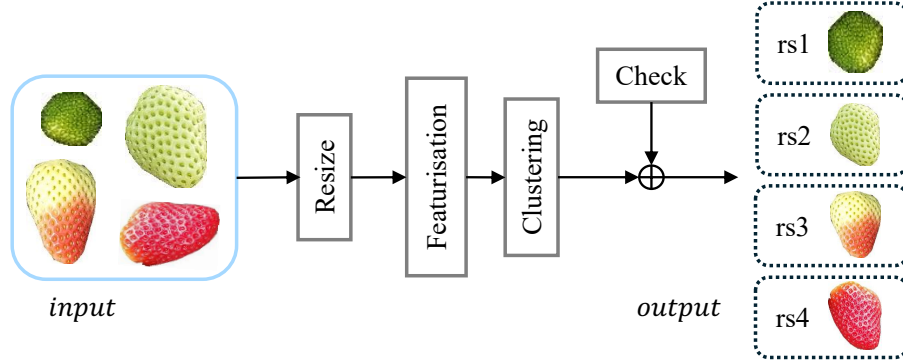


Figure 5.1: The process of strawberry mask classification.

instances are resized to  $280 \times 280$  pixels.

Second, some machine learning methods, like Histogram of Oriented Gradients, and deep learning methods like pre-trained CNN models are employed to extract features of the resized strawberry instances. Then, the cosine similarity is adopted to calculate the distance between features, resulting in similarity matrices.

Third, K-means clustering is applied to solve the similarity matrices, partitioning them into four clusters. The clustering method with the best performance was chosen to give the predictions.

Lastly, the clustering results were manually reviewed and corrected to ensure alignment with the predefined ripeness criteria. This refinement ensured that the final clustering outcomes adhered to the anticipated standards.

### 5.2.3 Dataset Summary

In summary, this study leverages two large fruit datasets, and both of them have individual mask annotations and ripeness stage labels. By integrating these two datasets, it can comprehensively cover scenarios involving both tree fruit (peaches) and berries (strawberries).

The combined dataset contains 7 different classes, with 3 classes corresponding to peaches and 4 classes to strawberries. This detailed dataset structure ensures a comprehensive representation of fruit development stages, facilitating more accurate and generalizable insights in subsequent analyses. Examples of images and their associated annota-



Figure 5.2: Examples of fruit instance annotation for the StrawDI\_Db1.

tions are presented in Fig. 5.2, and the distribution of instance categories is summarised in Table 5.2.

It is noted that the quantity of fruit instances decreases progressively over time as ripeness advances, revealing a real pattern that aligns with the natural growth and ripening process of fruit. By training on a combined dataset, the model is expected to handle these complexities across different object types, which enhances its robustness. Additionally, the inclusion of varied fruit types in a unified dataset can improve the model’s ability to distinguish between different objects, making it more adaptable to real-world applications where multiple fruit categories are often present simultaneously.

## 5.3 Method

### 5.3.1 Model Structure

For fruit ripeness determination, an instance segmentation model called FruitQuery is proposed, following the design of Mask2Former [39], which consists of a backbone, a pixel decoder and Transformer decoders. The architecture is illustrated in Fig. 5.4.

Table 5.2: The category distribution of the combined dataset.

NinePeach			StrawDI_Db1		
Category	Training	Validation	Category	Training	Validation
unripe	3669	1717	rs1	6693	453
semiripe	3312	1307	rs2	4014	319
ripe	1698	737	rs3	3010	212
/	/	/	rs4	2517	148
Instance	8679	3761	Instance	16234	1132
Image	3240	1359	Image	2800	100

## Backbone

It is well-known that the convolutional layer has inductive biases of locality and spatial invariance, which are capable of extracting low-level, small local features. The self-attention layer has a global receptive field and allows capturing global context information within an image. Therefore, these two types of layers are considered to build the backbone for multi-level feature extraction. The proposed backbone is illustrated in Fig. 5.3.

By combining convolutional layers with stronger generalisation performance and self-attention layers with higher model capacity and stronger learning ability, it is assumed that the backbone can achieve better generalisation performance and learning ability. Given an input image, the backbone can generate 4 levels of features, which provide high-resolution coarse features and low-resolution fine-grained features that usually boost the performance of fruit segmentation. It is noted that ConvBlock is removed in the last block in order to reduce the model parameters.

**Patch Embedding.** The input image is divided into a grid of non-overlapping patches, and each patch normally covers a square region of the image and is transformed into a fixed-dimensional embedding vector. According to different patch sizes and embedding dimensions, 4 different patch embedding blocks are attached in front of each block. As patch embedding does not inherently preserve positional information within each patch, it is required to add positional encoding to the subsequent two blocks.

**ConvBlock.** The ConvBlock is made of several convolutional layers with two residual connections. In the first residual connection, two  $1 \times 1$  point-wise convolutional layers (PWConv) are respectively placed before and after a  $5 \times 5$  convolutional layer. The  $5 \times 5$  convolutional layer has a larger receptive field to consider larger local regions and is expected to capture large-scale features like fruit edges and textures in images. In the second residual connection, two  $1 \times 1$  point-wise convolutional layers are used to perform MLP-like behaviour: increase the dimension to 4 times and then decrease it to the desired output dimension. This operation is designed to increase nonlinear



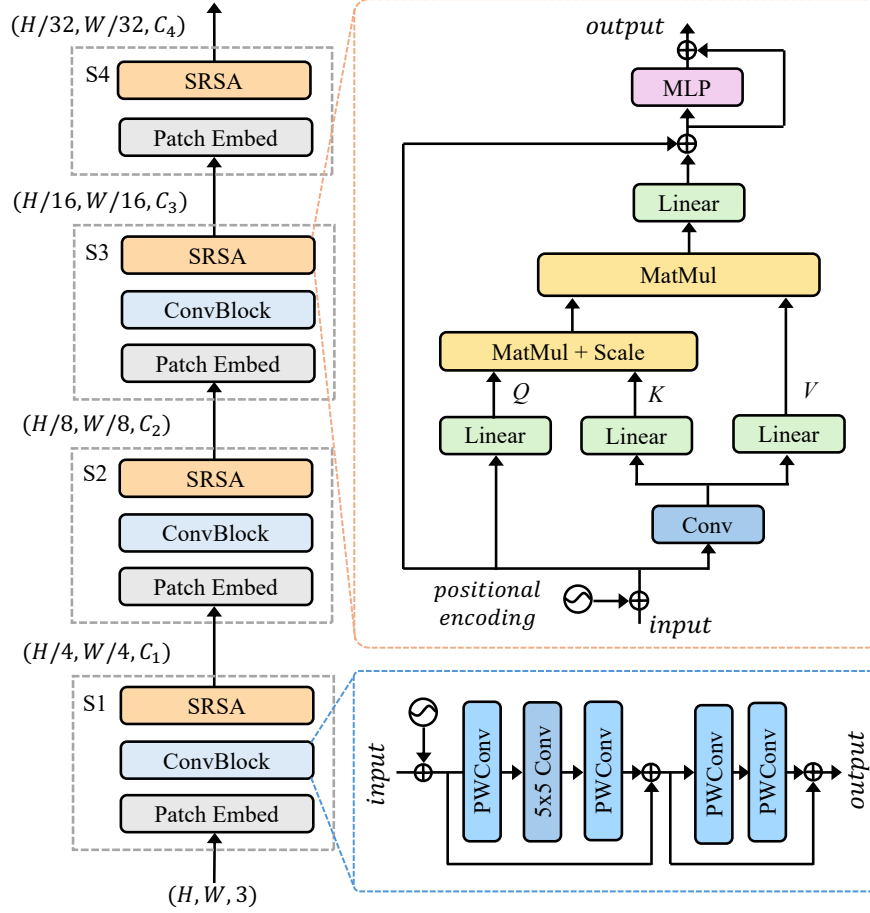


Figure 5.3: The proposed backbone of FruitQuery.

representation capacity and learn richer feature representations, thereby enhancing the model’s performance and generalisation ability. The  $1 \times 1$  point-wise convolutional layers only involve a single pixel and have fewer parameters to learn, therefore it is suitable for dimension expansion and compression. It is noted that ConvBlock is removed in the last stage to reduce the overall number of learnable parameters, which contributes to a more compact model size.

**Spatial Reduction Self-Attention (SRSA).** For each head of the multi-head self-attention, the query  $Q$ , key  $K$  and value  $V$  are obtained by applying three linear projections to the input embedding, including positional encoding.  $Q$ ,  $K$  and  $V$  have the same dimensions  $N \times C$ , where  $N = H \times W$ . Then, attention scores are calculated by the scaled dot-product attention. The scores are normalised using the Softmax function to obtain attention weights, which are used to compute a weighted sum of the  $V$  vectors of all tokens, as defined in Eq. (4.1), where  $d_k$  refers to the dimensionality of the key. Tokens with higher scores contribute more to the output of the self-attention mechanism.

The main bottleneck of the self-attention layer lies in its computation cost of  $O(N^2)$ , which scales quadratically with spatial dimension based on the input embedding. To alleviate this problem, the Spatial Reduction Self-Attention is introduced, which is based

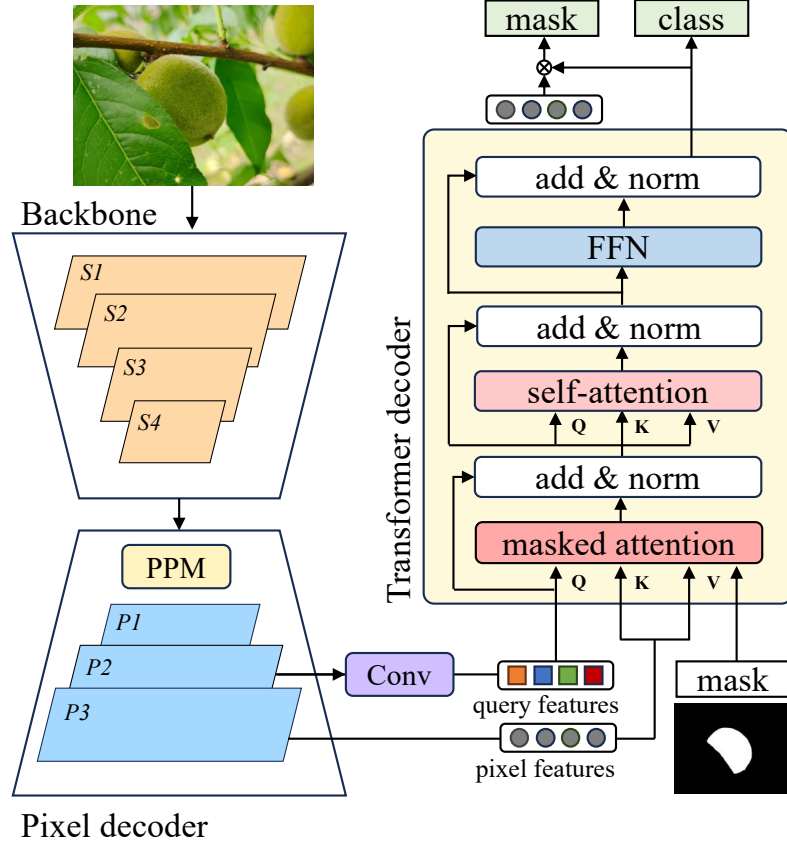


Figure 5.4: The architecture of FruitQuery.

on the spatial reduction method proposed in PVT [276]. The main idea of it is to reduce the length of the sequence with a reduction ratio  $R$ . For reducing computations, an input sequence with shape  $(C, H \cdot W)$  is reshaped to the  $\hat{K}$  with shape  $(C \cdot R^2, H \cdot W / R^2)$  based on Eq. (5.1). Here, a convolutional layer with  $kernel\_size = R$  and  $stride = R$  is used to perform the reshape operation. Eq. (5.2) refers to a linear layer taking  $\hat{K}$  as input and generating a new  $K'$  with shape  $(C, H \cdot W / R^2)$  as output.

$$\hat{K} = \text{Reshape}(K, R) \quad (5.1)$$

$$K' = \text{Linear}(C \cdot R^2, C)(\hat{K}) \quad (5.2)$$

As a result, the complexity of the efficient self-attention mechanism is reduced from  $O(N^2)$  to  $O(N^2/R^2)$ . It is noted that a residual MLP layer is appended at the end of SRSA to increase the model capacity and avoid overfitting.

To cater to diverse scenarios, two different settings for the backbone (s and xs) are proposed. The specifications are presented in Table 5.3, where  $C$  represents the number of embedded dimensions, and  $B$  denotes the number of blocks.

Table 5.3: The specification of the proposed backbones.

Stage	Size	Layer	xs	s
S1	$\frac{H}{4} \times \frac{W}{4}$	Patch Embed	Patch Size = 4, $C = C_1$	
		SRSA	$C_1 = 36$	$C_1 = 48$
			$B_1 = 1$	$B_1 = 1$
			$R_1 = 4$	$R_1 = 4$
S2	$\frac{H}{8} \times \frac{W}{8}$	Patch Embed	Patch Size = 2, $C = C_2$	
		SRSA	$C_2 = 72$	$C_2 = 96$
			$B_2 = 1$	$B_2 = 1$
			$R_2 = 2$	$R_2 = 2$
S3	$\frac{H}{16} \times \frac{W}{16}$	Patch Embed	Patch Size = 2, $C = C_3$	
		SRSA	$C_3 = 144$	$C_3 = 240$
			$B_3 = 3$	$B_3 = 3$
			$R_3 = 2$	$R_3 = 2$
S4	$\frac{H}{32} \times \frac{W}{32}$	Patch Embed	Patch Size = 2, $C = C_4$	
		SRSA	$C_4 = 288$	$C_4 = 384$
			$B_4 = 1$	$B_4 = 2$
			$R_4 = 1$	$R_4 = 1$

### Pixel Decoder

Multi-level contextual features play a crucial role in image segmentation, but employing a complex multi-scale feature pyramid network escalates the computational workload. For instance, multi-scale deformable attention used in Mask2Former demonstrates good performance, but it also brings a large number of parameters. To build a lightweight but effective model, the FPN is selected as the pixel decoder, which occupies less than half the size of the multi-scale deformable attention. FPN works by taking the features produced by the backbone at different levels (S1, S2, S3 and S4), and building a feature pyramid from top to down (P1, P2 and P3) through lateral connections (S4-P1, S3-P2, S2-P3).

A Pyramid Pooling Module (PPM, 313) is added to the top layer P1 to enlarge the receptive field and fuses the multi-scale features, of which the detail is shown in Fig 5.5. The input feature is divided into multiple regions of different sizes, using four different adaptive average pooling methods to capture information at different receptive field sizes. Then the pooled features are resized to the same size as the input, and concatenated with the input feature, resulting in a feature of shape  $(C + 4N, H, W)$ .

Finally, a simple convolutional layer is used to transform the shape of  $(C + 4N, H, W)$  back to  $(C, H, W)$  and fuse all information. Since the pooling operation does not introduce any new parameters, the introduction of PPM enhances the model’s performance without significantly increasing its computational complexity.

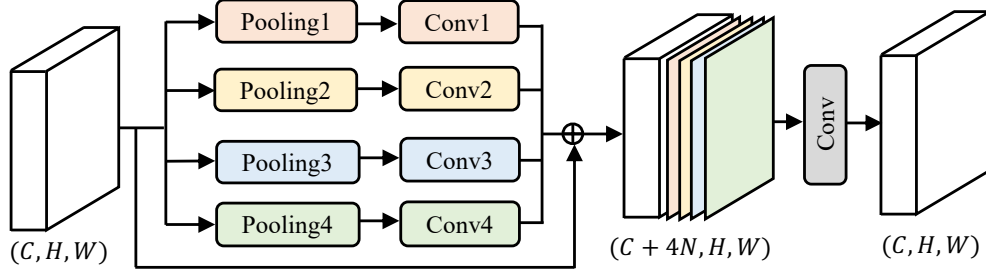


Figure 5.5: The illustration of the Pyramid Pooling Module.

The final output of the pixel decoder comprises features at three resolutions, incorporating both high-level features rich in semantics and low-level features rich in spatial information.

### Transformer Decoder

The Transformer decoder plays a crucial role in the model, which takes the learned features from the pixel decoder and processes them to produce the final output predictions. As shown in Fig 5.4, the decoder follows the paradigm of the standard architecture of the original Transformer, transforming  $N$  embeddings of objects into output embeddings. It is a stack of decoder layers, each of which consists of a masked attention layer, a self-attention layer and a Feed-Forward Network (FFN). Each Transformer decoder layer generates predictions for mask and class, but only the prediction of the last layer is used as the final prediction. The prior layer predictions can be used for auxiliary predictions optionally. The number of the Transformer decoder layers is set to 3 to achieve a better trade-off between accuracy and model size, and the feature P3 from the pixel decoder is used as pixel features.

**Query Features Initialisation.** The query features are important in the Transformer model, as they guide the decoder to attend to the most significant parts of the input embedding. Previous research indicates that query features can be initialised from zero [26], or can be updated by local features [39]. Although these two strategies are effective in generating query features, they require more decoders and longer training iterations to refine. Inspired by Deformable DETR, which selects a set of query bounding boxes from pyramidal features to perform object detection, and SparseInst [42], which introduces a simple convolutional module  $F_{iam}$  to highlight informative regions for each foreground object.

Therefore, these two advantages are combined in FruitQuery. A  $F_{iam}$ -like convolu-

tional module is added to efficiently initialise the query features in FruitQuery, which directly picks the queries with high semantics from underlying multi-scale feature maps. The simple module only consists of two convolutional layers. The first convolutional layer is a typical  $3 \times 3$  convolution layer with the same input and output dimensions. The second convolutional layer is a  $1 \times 1$  convolution layer to reduce the number of dimensions to the number of classes +1, where the extra one means “no object  $\phi$ ”. Specifically, feature  $P2$  from the pixel decoder is selected to generate  $N$  pixel embeddings with the highest foreground probabilities as the query features.

**Masked Attention.** The cross-attention in the original Transformer decoder is replaced with masked attention. The standard cross-attention is computed by Eq. (5.3).  $l$  is the layer index,  $X_l$  indicates the query features with the shape  $N \times C$  at the  $l$ -th layer.  $Q_l = f_q(X_{l-1})$  is calculated by applying a linear transformation  $f_q$  on the query features of previous layer.  $K_l$  and  $V_l$  are the pixel features from pixel decoder after linear transformations  $f_k$  and  $f_v$ .

$$X_l = \text{Softmax}(Q_l K_l^t) V_l + X_{l-1} \quad (5.3)$$

Based on cross-attention, masked attention adds an attention mask  $\mathcal{M}_{l-1}$ , as calculated in Eq. (5.4).

$$X_l = \text{Softmax}(\mathcal{M}_{l-1} + Q_l K_l^t) V_l + X_{l-1} \quad (5.4)$$

The attention mask  $\mathcal{M}_{l-1}$  at feature location  $(x, y)$  is calculated in Eq. (5.5), where  $m_{l-1}(x, y)$  is the binary output of the resized mask prediction of the previous  $(l-1)$  decoder layer.  $m_0$  is the binary mask prediction obtained from  $X_0$ .

$$\mathcal{M}_{l-1} = \begin{cases} 0 & \text{if } m_{l-1}(x, y) = 1 \\ -\infty & \text{otherwise} \end{cases} \quad (5.5)$$

### 5.3.2 Loss Function

The total training loss for FruitQuery is defined in Eq. (5.6):

$$\mathcal{L} = \lambda_{cl} \cdot \mathcal{L}_{class} + \lambda_m \cdot \mathcal{L}_{mask} + \lambda_{co} \cdot \mathcal{L}_{conv} + \lambda_a \cdot \mathcal{L}_{aux} \quad (5.6)$$

$\lambda$  indicates the different loss weights.

$\mathcal{L}_{class}$  is the cross-entropy loss between the selected class predictions and class targets, as defined in Eq. (4.6).

$\mathcal{L}_{mask}$  is the dice loss between the selected mask predictions and mask targets, as defined in Eq. (3.4).

$\mathcal{L}_{conv}$  is the cross-entropy loss between the output of  $F_{iam}$ -like convolutional module and ground truth, as defined in Eq. (4.6).

As each Transformer decoder layer generates class prediction and mask prediction, the prior predictions are used to calculate auxiliary loss  $\mathcal{L}_{aux}$ , as shown in Eq. (5.7),

$$\mathcal{L}_{aux} = \sum_{i=0}^D (\mathcal{L}_{class}^i + \mathcal{L}_{mask}^i) \quad (5.7)$$

where  $D$  indicates the number of Transformer decoders.  $\mathcal{L}'_{class}$  and  $\mathcal{L}'_{mask}$  use the same loss functions as  $\mathcal{L}_{class}$  and  $\mathcal{L}_{mask}$ .

Based on PointRend [140], which demonstrated that a segmentation model can be effectively trained by calculating its mask loss on a subset of randomly  $K$  sampled points instead of the entire mask, this strategy is incorporated into FruitQuery. Consequently, the mask loss is computed using sampled points both in the matching process and the final loss calculation.

The evaluation metrics include AP, Params, FLOPs and FPS, as defined in Section 3.3.3.

## 5.4 Experiments and Results

### 5.4.1 Experiments

#### Configuration

In this chapter, experiments are conducted based on Detectron2 and have been carried out using Python 3.9.13 and PyTorch 1.13 on a computer with the specifications shown in Table 3.4.

#### Training Details

No pre-trained weights are utilised in this work, and the parameters of all convolution layers are initialised by a standard normal distribution. The training process incorporates diverse data augmentation strategies to improve the model’s robustness and generalisation. These strategies contain random horizontal flips, resizing the input images such that the shortest side is one of 416, 448, 480, 512, 544, 576, 608 or 640 pixels while the longest is at most 768. This not only controls memory usage but also aligns with the original resolution of the dataset images, encouraging the model to adapt to objects of different scales.

Following the original Mask2Former design, the number of mask sampling points  $K$  is set to 12544, corresponding to a grid resolution of  $112 \times 112$ . The loss weights are set to  $\{\lambda_{cl}:2.0, \lambda_m:5.0, \lambda_{co}:20.0, \lambda_a:1.0\}$ . In contrast, the maximum prediction per image  $N$

is set to 100, and The depth of decoder layers  $D$  is set to 6, determined through ablation experiments in Section 5.4.3.

An AdamW optimiser is used with a step learning rate schedule, of which the initial rate is 0.0001, and the weight decay is 0.05. A learning rate multiplier of 0.1 is applied to the backbone, and the learning rate is decayed by 10 at fractions 0.9 and 0.95 of the total number of training iterations. All models are trained for 54k iterations with a batch size of 8.

### Inference Details

The data augmentation strategy used in inference is only resizing the input images such that the shortest side is 640 pixels while the longest is at most 768 pixels. Auxiliary predictions are not used during inference. The top 100 candidates with the highest confidence are selected as final predictions.

During inference, the data augmentation strategy is simplified to resizing the input images. Specifically, each image is resized such that the shortest side is scaled to 640 pixels while ensuring the longest side does not exceed 768 pixels, preserving the aspect ratio. Auxiliary predictions, like outputs from intermediate layers or heads used during training, are not used during inference to streamline the process and focus solely on the final model predictions. After the model generates predictions, the top 100 candidate predictions with the highest confidence scores are selected as the final predictions.

### 5.4.2 Main Results

A comprehensive segmentation comparison of different state-of-the-art backbones on the combined fruit dataset is conducted, using FruitQuery’s architecture shown in Fig. 5.4, and the results are summarised in Table 5.4.

#### Overall Performance

FruitQuery with SRSA-s (FruitQuery-s) achieves the highest overall AP of 67.02, AP<sub>50</sub> of 79.17, and AP<sub>75</sub> of 70.83, significantly outperforming 13 other models with a total of 33 variants. FruitQuery with SRSA-xs (FruitQuery-xs) also delivers a competitive AP of 66.46. This illustrates the superior performance of FruitQuery in fruit segmentation.

Among CNN-based models, the widely used ResNet series shows solid results, with ResNet-50 reaching an AP of 63.92. The recent FasterNet-l also achieves a competitive AP of 65.25. Turning to the YOLO series, YOLOv9-c attains the highest AP of 60.41 among its variants, indicating that the YOLO series has limited performance on fruit segmentation. In comparison, all YOLO variants fall short of FruitQuery.

On the Transformer-based side, models demonstrate more different designs and parameter counts. The variants of NextViT, GroupMixFormer, and PoolFormer generate



Table 5.4: Instance segmentation results on the combined dataset.

Backbone	Type	AP	AP <sub>50</sub>	AP <sub>75</sub>	NinePeach			StrawDI_Db1				Params (M)	FLOPs (G)	FPS
					AP <sub>unripe</sub>	AP <sub>semiripe</sub>	AP <sub>ripe</sub>	AP <sub>rs1</sub>	AP <sub>rs2</sub>	AP <sub>rs3</sub>	AP <sub>rs4</sub>			
CNN-based	ResNet [106]	18	61.74	75.68	65.37	50.61	53.50	65.72	39.52	73.64	75.80	17.54	29.56	19.50
		34	63.17	77.06	66.98	49.42	56.77	67.65	38.52	74.00	77.30	27.64	45.88	34.60
		50	63.92	77.29	67.70	53.62	54.78	<b>69.57</b>	41.61	75.33	77.17	30.52	50.94	31.54
	FasterNet [33]	s	62.56	75.04	66.20	49.72	50.78	63.21	44.76	74.04	78.29	35.82	91.85	17.90
		m	64.52	76.13	67.68	51.46	51.38	64.09	47.15	76.68	81.17	58.06	129.00	16.12
		l	65.25	77.01	69.15	53.24	53.61	66.47	<b>50.14</b>	<b>77.73</b>	77.72	97.70	189.00	15.60
	YOLOv8* [131]	s	57.33	72.44	63.67	48.49	49.66	68.02	32.64	61.44	65.70	11.79	46.12	<b>44.22</b>
		m	58.70	74.36	64.85	54.06	53.15	67.10	32.81	64.43	66.91	27.24	119.24	41.58
		l	59.74	75.50	66.17	55.32	52.63	66.79	34.11	64.14	71.46	45.94	238.48	34.36
	YOLOv9* [131]	s	59.91	75.23	66.12	<b>56.60</b>	54.33	65.10	33.85	65.99	69.95	8.64	82.26	13.4
		m	60.04	75.72	66.57	54.90	54.08	68.63	33.50	65.21	69.34	22.26	142.38	15.78
		c	60.41	76.07	67.13	54.37	55.13	68.13	33.89	66.53	69.13	27.84	171.82	15.58
Transformer-based	YOLOv10* [131]	s	58.17	73.91	64.53	50.86	50.34	63.50	33.34	68.75	66.18	9.20	44.10	21.28
		m	58.60	74.23	65.18	52.56	50.57	65.86	33.23	65.76	68.09	19.37	110.06	18.94
		b	58.69	73.60	64.87	52.95	51.82	65.39	32.71	63.70	69.35	25.52	180.68	17.94
	MobileViT [191]	xxs	46.34	63.25	50.76	28.26	38.27	50.81	20.12	54.66	63.06	<b>7.27</b>	<b>17.27</b>	19.74
		xs	50.29	67.14	54.81	34.17	41.35	54.80	23.85	62.78	62.95	8.28	21.14	19.56
		s	52.90	68.62	56.82	38.08	44.19	56.41	27.65	62.97	70.42	11.36	26.99	19.44
	LightViT [115]	t	54.65	69.90	58.59	40.05	44.47	57.85	30.77	66.00	71.37	14.06	29.11	14.36
		s	56.31	71.25	60.30	42.43	46.42	59.60	31.96	68.74	70.89	23.74	28.03	14.28
		b	58.57	73.17	62.39	45.18	50.00	62.50	34.13	70.31	72.29	39.63	45.06	13.49

\* YOLO series are trained using their segment head.

Table 5.4 (Continued) Instance segmentation results on the combined dataset.

Backbone	Type	AP	AP <sub>50</sub>	AP <sub>75</sub>	NinePeach			StrawDI_Db1				Params (M)	FLOPs (G)	FPS
					AP <sub>unripe</sub>	AP <sub>semiripe</sub>	AP <sub>ripe</sub>	AP <sub>rs1</sub>	AP <sub>rs2</sub>	AP <sub>rs3</sub>	AP <sub>rs4</sub>			
NextViT [155]	s	62.36	74.88	66.02	48.79	49.99	62.38	48.11	73.57	77.61	76.07	37.96	105.00	15.06
	b	62.47	75.03	65.81	47.71	51.79	61.08	45.80	77.66	75.75	77.52	51.02	128.00	13.10
GroupMix- Former [54]	t	62.67	75.45	66.21	47.42	52.55	63.07	42.81	74.79	78.39	79.67	17.62	83.72	9.24
	s	63.50	76.31	67.79	48.81	53.66	64.79	44.53	74.60	79.36	78.75	29.06	96.51	9.02
MetaFormer [302]	id	59.65	73.16	63.43	43.19	48.77	62.98	40.86	71.14	75.95	74.69	18.40	66.92	18.94
SegFormer [293]	b0	58.48	71.53	62.36	42.33	47.76	58.71	40.09	67.83	75.39	77.28	10.19	57.06	17.32
PoolFormer [302]	s12	61.12	74.19	64.48	44.19	47.02	61.16	42.94	73.75	78.19	80.60	18.40	66.92	13.50
	s24	62.61	75.17	66.28	45.27	49.63	64.78	44.69	73.76	79.89	80.24	27.88	80.94	12.34
TransXNet [293]	t	59.26	72.48	62.66	42.31	46.70	63.03	39.81	71.59	73.96	77.44	19.33	70.65	10.88
	s	60.43	73.64	63.87	45.07	49.16	62.24	40.31	71.02	74.53	80.66	33.34	98.84	8.22
CMT [95]	ti	66.00	78.44	69.43	54.80	57.58	68.98	46.84	77.23	78.81	77.79	14.57	67.55	14.06
	xs	66.46	78.60	70.10	55.79	57.45	68.96	47.72	76.35	78.37	80.61	20.21	78.27	13.08
SRSA (FruitQuery)	xs	66.46	78.49	70.27	51.77	56.11	69.03	47.12	77.16	82.12	81.92	10.94	61.56	16.50
	s	<b>67.02</b>	<b>79.17</b>	<b>70.83</b>	52.05	<b>58.68</b>	68.55	47.91	76.47	<b>82.16</b>	<b>83.74</b>	14.08	69.33	16.00

Transformer-based

similar results of AP, ranging from 62.37 to 63.50. Two CMT variants reach APs of 66.00 and 66.46, coming closest to FruitQuery’s performance. These Transformer-based models reflect the trend toward attention-driven backbones, with noticeable performance gains over many CNN counterparts.

However, they still fall short of FruitQuery in AP, AP<sub>50</sub> and AP<sub>75</sub>, suggesting that the proposed query-based design leverages features more effectively for precise fruit instance segmentation.

### Individual Performance

For the NinePeach dataset, YOLOv9-s achieves the highest AP<sub>unripe</sub> of 56.60, while Res50 delivers the highest AP<sub>ripe</sub> of 69.57. However, FruitQuery attains the best performance on semiripe peaches with an AP<sub>semiipe</sub> of 58.68, underscoring its ability to capture the more subtle visual cues present in intermediate ripeness for peaches.

For the StrawDl\_Db1 dataset, CNN-based FasterNet obtains the highest AP<sub>rs1</sub> of 50.14 and AP<sub>rs2</sub> of 77.73, while FruitQuery-s outperforms all counterparts in half of the strawberry ripeness stages, with the highest AP<sub>rs3</sub> of 82.16 and AP<sub>rs4</sub> of 83.74. These gains indicate that FruitQuery can effectively handle the appearance variations in later strawberry growth, where colour, texture, and shape have significant changes compared to earlier stages.

Overall, within seven ripeness stages of the combined dataset, FruitQuery delivers the best AP for three of them, indicating that FruitQuery, with the query-based design, is capable of capturing fine-grained features within different fruit ripeness levels and generating comparable results.

### Model Complexity

The broad range of model sizes is generally related to performance: larger models typically have more parameters, which allows them to capture more complex patterns and relations.

On the CNN-based models, FasterNet-l is the largest CNN-based model with parameters of 97.70M and FLOPs of 189G, and it achieves a competitive AP of 65.25. Notably, the YOLO series is well-known for its lightweight design, with YOLOv9-s having 8.64M parameters and 82.26G FLOPs, and YOLOv10-s having 7.27M parameters and 44.10G FLOPs, but their AP of 59.91 and 58.17 are lower than many other models.

On the Transformer-based models, MobileViT-xxs exhibits the smallest parameter count of 7.27M and FLOPs of 17.27G, while it comes with the lowest AP of 46.34. NextViT-b is the most complex Transformer-based model with 51.02M parameters and 128G FLOPs, delivering an AP of 62.47.

FruitQuery shows a highly cost-efficient design. Specifically, the FruitQuery-xs only utilises 10.94M parameters and 61.56G FLOPs to achieve an AP of 66.46, and FruitQuery-

s attains the highest AP of 67.02 with 14.08M parameters and 69.33G FLOPs.

In contrast, models with similar APs to FruitQuery-xs (66.46), such as CMT-ti (66.00), CMT-xs (66.46) and FasterNet-l (65.25), require larger parameters and FLOPs (14.57M/67.55G, 20.21M/78.27G and 97.70M/189.00G) than FruitQuery-xs (10.94M/61.56G). On the other hand, models that match FruitQuery-xs in parameters and FLOPs (10.94M /61.56G), such as YOLOv8-s (11.79M/46.12G), MobileViT-s (11.36M/26.99G) and SegFormer -s24 (10.19M/57.06G), deliver poorer APs (57.33, 52.90 and 58.48).

## Inference Speed

CNN-based models exhibit higher inference speeds compared to Transformer-based models, consistent with the established efficiency advantages of convolutional architectures. Among all evaluated models, YOLOv8-t achieves the highest FPS at 44.22, followed by YOLOv8-m (41.58) and YOLOv8-l (34.36), highlighting the real-time capabilities.

The proposed FruitQuery achieves relatively high inference speeds (16.5 and 16 FPS), demonstrating competitive inference performance. They outperform all YOLOv9 variants, suggesting improved speed efficiency relative to this recent Transformer-based series. In addition, FruitQuery surpasses a number of widely used Transformer-based backbones such as LightViT-t (14.36), CMT-ti (14.06), and NextViT-s (15.06), which are specifically designed for efficiency.

While slightly slower than MobileViT-xxs (19.74) and MobileViT-xs (19.56), FruitQuery is notably faster than recent models like TransXNet-s (8.22) and GroupMixFormer-s (9.02), positioning them among the faster Transformer-based designs. These results indicate that FruitQuery strikes a favourable balance between inference speed and model complexity.

In summary, the results demonstrate that FruitQuery not only exhibits comparable or even superior results to other segmentation models but also maintains a lightweight model size and higher efficiency.

Table 5.5: Ablation on the pixel decoder.

Module	AP	AP <sub>50</sub>	AP <sub>75</sub>
FPN	64.97	78.41	68.88
PPM-FPN	<b>66.57</b>	<b>78.98</b>	<b>70.22</b>

### 5.4.3 Ablation Experiments

#### Type of Pixel Decoder

Table 5.5 compares two different pixel decoders of FPN and PPM-FPN, in terms of model performance. The baseline FPN achieves an AP of 64.97, AP<sub>50</sub> of 78.41, and AP<sub>75</sub> of 68.88. In contrast, the PPM-FPN variant leads to a consistent performance boost across all metrics, improving AP by 1.6 points from 64.97 to 66.57, AP<sub>50</sub> by 0.57 points from 78.41 to 78.98, and AP<sub>75</sub> by 1.34 points from 68.88 to 70.22. These results indicate that incorporating PPM into the FPN enhances the overall segmentation performance.

Table 5.6: Results of ablation experiments based on FruitQuery-xs.

(a) Ablation on the number of attention head.

Head	AP	AP <sub>50</sub>	AP <sub>75</sub>
2	62.36	75.30	65.80
4	<b>64.46</b>	<b>77.09</b>	<b>68.61</b>
8	64.36	76.98	67.88

(b) Ablation on the number of queries.

Query	AP	AP <sub>50</sub>	AP <sub>75</sub>
80	64.54	77.74	68.39
90	64.45	77.38	68.15
100	<b>66.52</b>	<b>78.84</b>	<b>70.19</b>
110	65.91	78.56	69.48
120	65.48	78.34	69.11

(c) Ablation on the pixel decoder.

Layer	AP	AP <sub>50</sub>	AP <sub>75</sub>
1	62.85	75.78	66.59
2	63.98	76.93	67.72
3	63.91	76.69	67.86
4	66.41	78.98	70.20
5	66.68	79.31	70.40
6	<b>66.80</b>	<b>78.95</b>	<b>70.85</b>
7	65.94	78.49	69.36
8	65.78	78.07	69.26

#### Number of Decoder Attention Head

Table 5.6a compares the effect of different numbers of attention heads on model performance. With just 2 heads, the model attains an AP of 62.36, AP<sub>50</sub> of 75.30, and AP<sub>75</sub> of 65.80, indicating limited representational capacity. Increasing to 4 heads yields the highest AP of 64.46, AP<sub>50</sub> to 77.09 and AP<sub>75</sub> of 68.61. Although further increasing the number of heads to 8 slightly boosts AP to 64.46 and AP<sub>75</sub> to 67.88 compared to 2 heads, it still lags behind the 4-head configuration. These results suggest that 4 attention heads provide an optimal balance, offering richer feature representations without incurring diminishing returns.

### Number of Query

Table 5.6b shows the effect of different numbers of queries on model performance. When the number of queries is set to 100, the model achieves its highest overall AP of 66.52,  $AP_{50}$  of 78.84, and  $AP_{75}$  of 70.19. Reducing the number of queries below 100 (e.g., 80 or 90) results in poorer performance across all metrics. Conversely, increasing the number of queries beyond 100 (e.g., 110 or 120) does not lead to any additional benefits. These results imply that 100 queries is an optimal balance for capturing sufficient object-level features without excessive costs.

### Number of Decoder Layers

Table 5.6c illustrates the effect of different numbers of decoder layers on model performance. With only 1 to 3 layers, AP stays between 62.85 and 63.98, indicating limited representational depth. As more layers are added, accuracy steadily improves, peaking at 6 layers with an AP of 66.80,  $AP_{50}$  of 78.95, and  $AP_{75}$  of 70.85. Beyond 6 layers, model performance begins to decline, suggesting that excessive stacking of decoder blocks may introduce redundancy or complicate training. These findings highlight an optimal spot at 6 decoder layers.

Table 5.7: The AP comparison of training on separate and combined datasets.

Dataset	Category	t/o separate	t/o combined
NinePeach	unripe	43.72	<b>51.27</b>
	semiripe	49.07	<b>53.93</b>
	ripe	62.05	<b>66.94</b>
StrawDlDb1	rs1	<b>45.57</b>	42.29
	rs2	72.55	<b>76.73</b>
	rs3	<b>79.87</b>	78.40
	rs4	77.99	79.64
Average		61.08	<b>64.63</b>

#### 5.4.4 Combined and Separate Training

The performance difference of FruitQuery-xs trained on combined (t/o combined) and separate (t/o separate) datasets is compared, and the results are shown in Table 5.7. For NinePeach, the combined training strategy produces notable improvements across all ripeness levels, with  $AP_{unripe}$  increases of 7.55 points from 43.72 to 51.27, 4.86 points for  $AP_{semiripe}$  from 49.07 to 53.93, and 4.89 points for  $AP_{ripe}$  from 62.05 to 66.94.

In contrast, results on StrawDl\_Db1 are mixed:  $AP_{rs2}$  has a significant gain of 4.18 points from 72.55 to 76.73, and  $AP_{rs4}$  also increases 1.65 points from 77.99 to 79.64. However, the other two categories  $AP_{rs1}$  drops from 45.57 to 42.29 and  $AP_{rs3}$  drops from 79.87 to 78.40.

Overall, training on the combined dataset boosts the model’s overall AP from 61.08 to 64.63, indicating that learning from a broader, integrated fruit distribution can enhance generalisation for the majority of fruit ripeness stages despite limited category-specific trade-offs.

Table 5.8: The parameters comparison of YOLOv9 and FruitQuery.

Aspect	YOLOv9			FruitQuery	
Type	s	m	c	xs	s
Backbone (M)	5.72	15.52	19.95	<b>4.07</b>	7.15
Neck (M)	/	/	/	<b>2.70</b>	2.76
Head (M)	<b>2.92</b>	6.74	7.89	4.18	4.18
Total (M)	<b>8.64</b>	22.26	27.84	10.94	14.08
AP	59.91	60.04	60.41	66.46	<b>67.02</b>

### 5.4.5 Model Parameter Distribution

The parameter distribution of YOLO and FrutiQuery is summarised in Table 5.8. Based on previous results in Table 5.4, YOLOv9 is the best-performing version of the three YOLO series, therefore, it is selected to compare with FruitQuery and also in later comparisons.

YOLOv9-s has the least number of parameters of 8.64M, with a head of 2.92M, but produces the lowest AP of 59.91. When changing the model from YOLOv9-s to YOLOv9-m, the total parameters increase to 22.26M, with a bigger backbone and head, but bring a tiny AP gain from 59.91 to 60.04. YOLOv9-c performs better than YOLOv9-m with the AP of 60.41, but occupies a backbone of 19.95M and a head of 7.89M.

On the other hand, FruitQuery demonstrates its ability to outperform YOLOv9 with fewer parameter counts. Specifically, FruitQuery-xs and FruitQuery-s have an identical head of 4.18M, which is smaller than YOLOv9-m and YOLOv9-c. The main difference between the two variants of FruitQuery lies in the backbone. FruitQuery-s has a more complex backbone and delivers a better AP of 67.02.

These results not only demonstrate that FruitQuery achieves a significantly better balance between the segmentation performance and model size compared to YOLO but also highlight its lightweight design, which enhances the potential for in-field applications.





Figure 5.6: Segmentation visualisations of FruitQuery on NinePeach (top) and StrawDI\_Db1 (bottom).

#### 5.4.6 Visualisation

The segmentation performance of FruitQuery is visualised in Fig. 5.6. First, FruitQuery is capable of simultaneously segmenting peaches and strawberries without requiring separate training for each fruit type. Second, FruitQuery demonstrates strong generalisation ability on fruit size due to effective multi-scale feature fusion. Specifically, the size of



peaches is relatively large compared to that of strawberries, and FruitQuery can accurately segment both large and small fruit. Third, FruitQuery maintains high robustness in complex in-field conditions, such as occlusions from tree trunks and leaves, delivering precise fruit segmentation. These indicate that FruitQuery can accurately predict fruit locations for downstream applications.

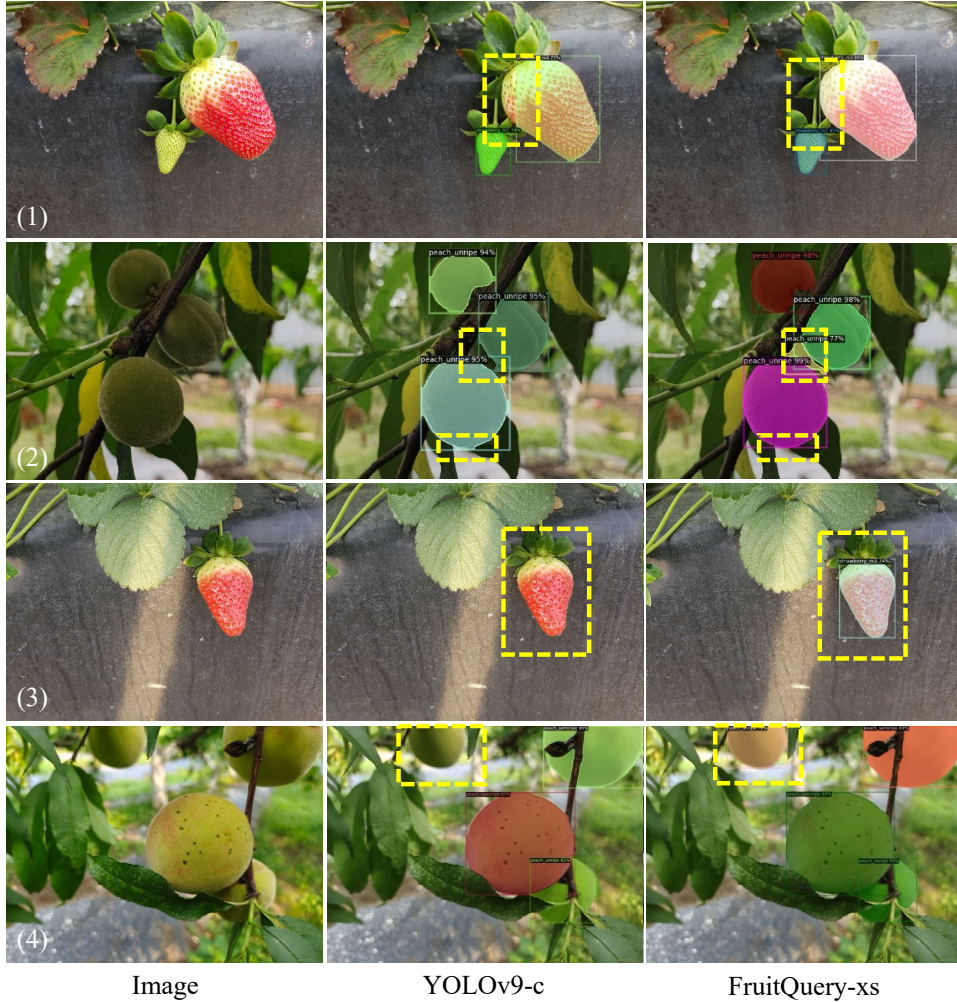


Figure 5.7: The segmentation comparison of YOLOv9 and FruitQuery.

The visualisation of FruitQuery and YOLO is also compared, as shown in Fig. 5.7. In case (1), although YOLOv9-c is not an anchor-based model, it still gives an inaccurate anchor-like prediction on the strawberry, while FruitQuery provides a more precise delineation of the strawberry’s shape. In case (2), YOLO-v9c’s segmentation boundary tends to follow the rectangular outline of the bounding box, while FruitQuery closely tracks the actual peach boundary. Additionally, YOLO-v9c ignores the small peach behind, while FruitQuery correctly detects it. In case (3), YOLO-v9c fails to detect an evidently visible strawberry, while FruitQuery successfully identifies and segments it. In case (4), YOLO-v9c is unable to recognise a peach partially hidden in the background, whereas FruitQuery correctly distinguishes the peach despite the limited visible part.

### 5.4.7 Class Activation Map Analysis

Class Activation Maps (CAM, 318) is a popular visualisation technique that highlights the regions in an image most influential to a model’s prediction. By projecting learned feature weights back onto the original input, CAM reveals where the model allocates its attention and provides an interpretable window into the decision-making process. The CAM comparison of YOLO and FruitQuery is illustrated in Fig. 5.8.

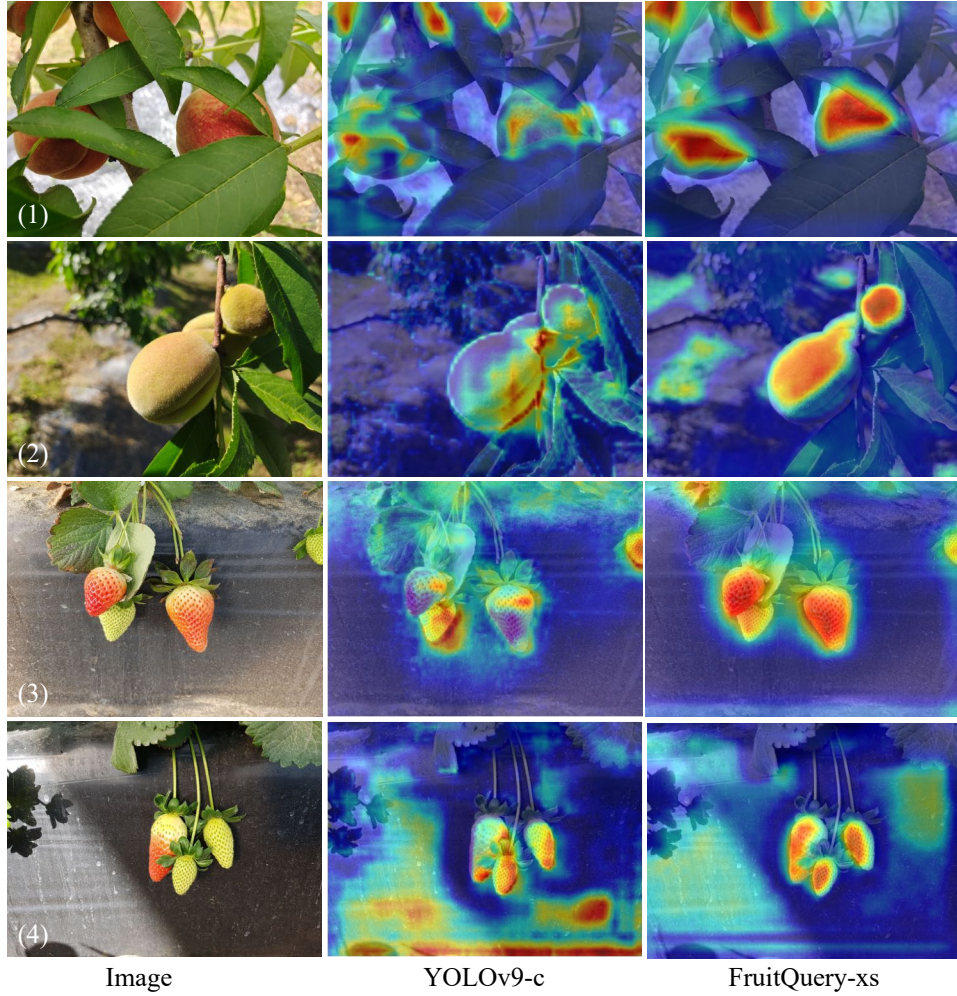


Figure 5.8: The CAM comparison of YOLOv9 and FruitQuery.

In the CAM visualisations, YOLOv9-c exhibits relatively diffuse and occasionally misaligned attention, focusing on broader or less discriminative regions. For example, in cases (1) and (2), YOLOv9-c has uncertain attention on the fruit and is affected by the surrounding leaves. By contrast, FruitQuery maintains a more localised and precise concentration of high-intensity activation around the fruit. This difference is particularly evident in cases (3) and (4). YOLOv9-c looks at a large blur region around fruit and gives attention to the irrelevant background, while FruitQuery accurately distinguishes between fruit and background context, capturing finer textural cues on peaches and strawberries and generating tightly focused activation zones.

Consequently, the visualisations demonstrate the enhanced ability of FruitQuery to learn the discriminative features of peaches and strawberries, such as shape, colour transition, and edge boundaries, eventually resulting in interpretable and improved segmentation performance.

### 5.4.8 Deployment

The inference speed of the FruitQuery was evaluated on three types of hardware platforms, as introduced in Section 4.4.4. For each FruitQuery variant, inference was conducted under two numerical precision settings, FP32 and FP16, and across multiple deployment formats, including PyTorch (.pth), ONNX (.onnx), TensorRT (.trt), and Core ML (.mlmodel). The results are summarised in Table 5.9.

Table 5.9: The FPS of FruitQuery across different devices.

Device	Format	FruitQuery-xs		FruitQuery-s	
		FP32	FP16	FP32	FP16
NVIDIA Tesla V100	.pth	16.18	18.38	16.00	16.81
	.onnx	3.76	3.93	3.74	3.90
	.trt*	<b>39.53</b>	<b>51.02</b>	<b>39.37</b>	<b>50.51</b>
NVIDIA Jetson Orin Nano	.pth	3.02	4.45	2.92	4.44
	.onnx	2.86	3.17	2.80	3.08
	.trt*	8.56	12.63	8.22	11.68
Apple M1	.onnx	0.87	0.71	0.58	0.60
	.mlmodel	0.40	0.43	0.37	0.40

\* Due to hardware limitations, V100 is using TensorRT 8.6.1 while Orin Nano is using TensorRT 10.3.0.

The model formats .pth, .onnx, and .trt have been described in Section 4.4.4. In addition to these, deployment on Apple M1 is extended to include Core ML (.mlmodel), which is a model format specifically optimised for Apple’s ecosystem. It allows models to be executed using the Core ML framework, which internally utilises the Apple Neural Engine to accelerate inference on iOS and macOS devices.

Two numerical precision modes are compared: FP32 (single-precision floating point) and FP16 (half-precision floating point). FP16 reduces memory usage and computation cost by representing floating-point numbers with 16 bits instead of 32, which can accelerate inference on compatible hardware while maintaining adequate accuracy.

Inference results demonstrate that TensorRT achieves the fastest inference across all tested hardware for both precision modes. On the Tesla V100, FruitQuery-xs has a 51.02 FPS using FP16 and TensorRT, which is more than 2.5 times faster than the PyTorch

baseline under FP32. Jetson Orin Nano also benefits significantly from TensorRT acceleration, achieving a 12.63 FPS for FruitQuery-xs in FP16. This highlights the suitability of TensorRT for edge deployment when latency is critical.

Compared to the V100 and Jetson platforms, the Apple M1 incurs higher inference latency. The ONNX-based deployment shows low FPS for both model variants. Moreover, Core ML deployment with .mlmodel results in even longer running time, with FruitQuery-s requiring a 0.40 FPS under FP16. Despite this, the Core ML format enables compatibility with Apple-native applications and can utilise the underlying Neural Engine, which may provide performance gains in future hardware iterations.

Overall, the proposed FruitQuery models demonstrate efficient inference across diverse deployment scenarios. The results affirm the flexibility of the models and their compatibility with multiple deployment backends and precision modes. Particularly, the combination of lightweight architectures, TensorRT optimisation, and half-precision inference enables real-time performance on both cloud GPUs and edge platforms.

## 5.5 Discussion

### 5.5.1 Comparison to PeachSOLO

As shown in Section 3.4.2, PeachSOLO with Swin-FPN achieves the highest 72.12 AP, followed by PeachSOLO with Res50-FPN at 66.33 AP, occupying 46.17M parameters and running at 11.11 FPS. In comparison, FruitQuery-s achieves a slightly lower AP of 59.76, but with a much smaller model size of 14.08M parameters and a higher speed of 16.00 FPS.

This difference in segmentation accuracy is expected, as larger models provide greater capacity to learn detailed and category-specific features. However, FruitQuery-s was designed with lightweight deployment in mind. It provides a favourable trade-off between model performance and efficiency, making it well-suited for real-time applications on edge devices. Its ability to handle multiple fruit types and ripeness stages within a single compact model further highlights its practical value for robotic harvesting systems operating under computational constraints.

### 5.5.2 Limitations

First, FruitQuery still relies on a large quantity of manually annotated data, particularly with instance-level and ripeness-specific labels. This requirement brings a significant bottleneck in extending the model to new fruit types or orchard conditions, where annotation can be time-consuming and labour-intensive.

Second, although FruitQuery is designed to be lightweight, its current inference speed



does not yet meet the requirements of real-time operation on highly resource-constrained edge devices. Achieving true real-time performance remains a challenge, particularly when handling high-resolution inputs in dynamic field environments.

These limitations indicate promising directions for further work, including reducing annotation costs and enhancing inference efficiency for deployment in practical agricultural settings.

### 5.5.3 Future Work

First, FruitQuery will be further compressed and optimised using techniques such as quantisation, pruning, and architecture refinement. The goal is to enable real-time inference on embedded platforms, thereby facilitating in-field deployment for robotic fruit harvesting with immediate ripeness feedback.

Second, future work will explore self-supervised or semi-supervised learning approaches to reduce reliance on manual annotations. The model can be more readily adapted to diverse fruit types and conditions with improved data efficiency. In addition, expanding the current dataset to include a broader spectrum of fruit varieties and ripeness stages will enhance the model’s multi-fruit applicability.

## 5.6 Summary

In this chapter, two in-field fruit datasets of peaches and strawberries are combined, which contain 3 ripeness stages for peaches and 4 ripeness stages for strawberries. Then, a lightweight query-based instance segmentation model for fruit ripeness determination called FruitQuery is introduced.

The combined dataset enables training the model to handle the ripeness determination of two fruits at the same time, reducing the effort to replicate the training. FruitQuery is composed of three main components: a backbone, a pixel decoder, and Transformer decoders. The SRSA module is integrated into the backbone to reduce computational overhead and introduces a PPM in the pixel decoder to improve multi-scale feature fusion. Transformer decoders were employed to learn a fixed number of queries for instance masks, eliminating the need for postprocessing like NMS.

By combining the advantages of convolution and Transformer, FruitQuery runs in an end-to-end way and precisely attends to fruit regions, capturing subtle distinctions in shape and ripeness. The design of FruitQuery leads to state-of-the-art performance, achieving the highest AP of 67.02 with 14.08M parameters and surpassing 13 other CNN-based and Transformed-based models. Notably, it outperforms three series of YOLO, under challenging conditions such as occlusion and varying illumination. However, FruitQuery’s dependence on labelled data makes it challenging for swift adaptation to new

fruit varieties. Additionally, latency issues may be a problem for FruitQuery when applied on embedded platforms.

Moving forward, FruitQuery will be further optimised for in-field applications, exploring strategies like quantisation for edge deployment. The combined dataset is planned to be expanded with more fruit varieties, ultimately building a large-scale fruit instance segmentation dataset with ripeness labels. Through these enhancements, FruitQuery is expected to increase its utility in orchard automation, enabling more accurate and efficient fruit ripeness determination and helping the development of precision agriculture.



# Chapter 6

## AppleSSL: A Novel Self-supervised Method for In-field Occluded Apple Ripeness Determination

### 6.1 Introduction

While the previous chapters have addressed fruit ripeness determination through classification and segmentation using either fully supervised or lightweight deep learning models, they typically require a considerable number of labelled images and often assume a clear view of the fruit surface. These assumptions limit their applicability in real-world orchard environments, where the fruits are frequently occluded by leaves and branches, and the process of annotating ripeness stages remains highly subjective and labour-intensive. Furthermore, most prior models are constrained by the need to pre-define discrete ripeness categories, which may not reflect the nuanced and individualised decision-making processes of end-users in precision agriculture. To address these practical challenges and advance the existing body of work, this chapter introduces a novel approach that incorporates self-supervised learning for in-field occluded apple ripeness determination.

In the context of apple precision agriculture, variations in apple ripening times exist both among trees within the same orchard and even among apples on the same tree, as illustrated in Fig. 6.1. The differences in ripening times are influenced by a combination of environmental conditions, biological traits, and human interventions. This lack of selectivity can lead to reduced apple market value and the need for post-harvest sorting.

#### 6.1.1 Ripeness labelling

Determining apple ripeness from images is usually a subjective and challenging task. Fig. 6.2 shows that the definitions of “ripe” can be different among different users, ranging



Figure 6.1: Apples with distinct ripeness difference can appear simultaneously.

from binary classifications to more granular multi-category classifications. Binary and three-category classifications are the most commonly considered by previous research. However, extending these models to finer classifications, such as five categories, requires re-labelling the images and retraining the model, which introduces unnecessary effort. To solve this, this chapter regards ripeness determination as a regression task rather than a multi-category classification task.

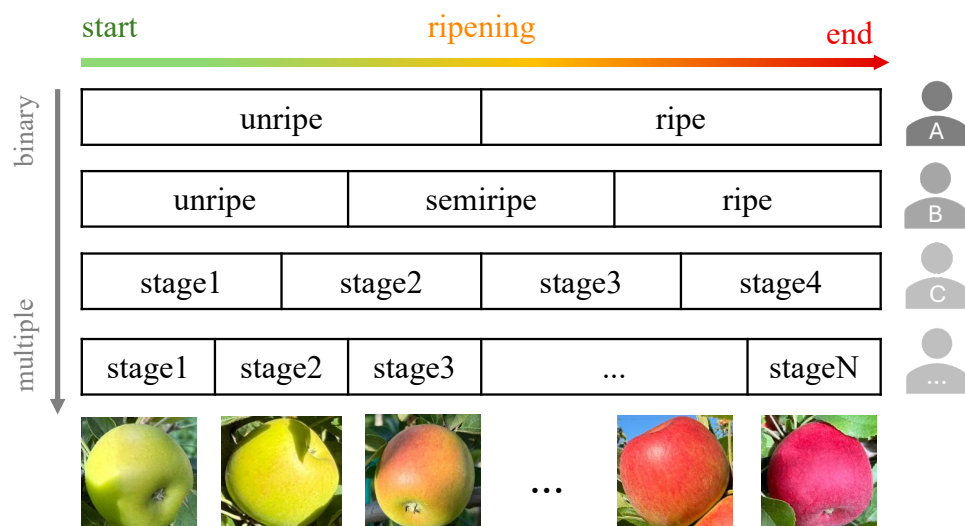


Figure 6.2: Different users have different criteria for apple ripeness.

It is noted that regardless of the number of ripeness stages defined by the users, the fully unripe and fully ripe apples will always remain in the first and last categories, respectively. Based on this, this chapter proposed a self-supervised method which takes few images of fully unripe and fully ripe apples as labels, learns from a large number of unlabelled images, and generates ripeness scores as output.

### 6.1.2 In-field Occlusion

In-field occlusion is the second challenge in this work. Since most of these robots heavily depend on visual perception for fruit identification and localisation, occlusion significantly impacts their decision-making process. As shown in Fig. 6.1, apples are often easily occluded by leaves. Moreover, occlusion can also result in recognition failures, requiring manual leaf removal prior to picking [269].

Some of the previous research has considered the occlusion when training the detection and segmentation models. [265] introduced a YOLO-based model specifically designed for detecting apples at different growth stages in orchards and mitigated apple overlap and occlusion to some extent. [316] proposed a CNN-based vision algorithm for mango instance segmentation and picking point localisation, considering occlusion, overlap, and variations in object scale. [272] replaced the network’s complete-IoU regression loss function with the weighted-IoU loss function to address tomato fruit and leaf occlusion. [35] proposed a YOLO-based lightweight 4-class occlusion detection method for *Camellia oleifera* fruit, introducing a clustering algorithm to select the target dataset. Similarly, [61] proposed a detection model to locate ripe ground-planted strawberries of 4 different occlusion categories.

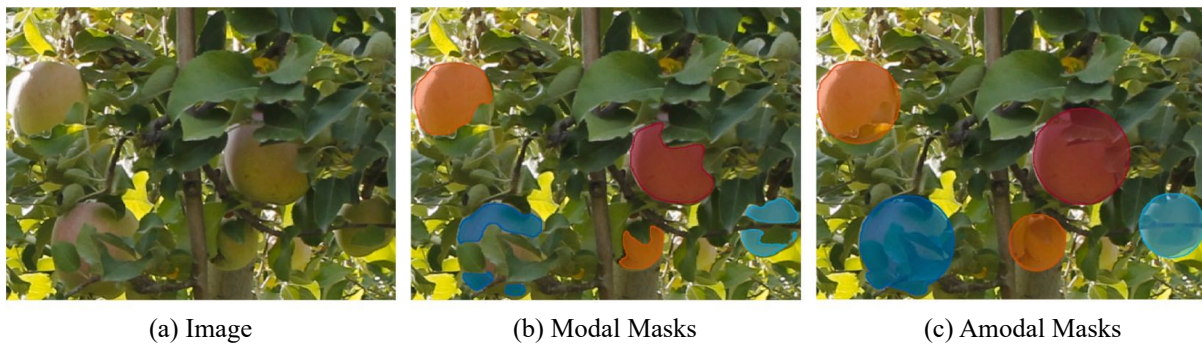


Figure 6.3: Example of modal and amodal masks [79].

Furthermore, some researchers proposed to estimate the shape of partially occluded fruits by means of amodal instance segmentation, which aims to predict the shape of each object of interest in an image [156]. [79] implemented an amodal segmentation model with an end-to-end CNN for accurate Fiji apple detection and sizing, predicting complete shapes (visible and occluded regions) and achieving robust diameter estimation. The examples of modal and amodal masks are shown in Fig. 6.3. [137] employed an amodal segmentation approach using a reconstruction network to perform cucumber occlusion recovery, achieving high accuracy and speed. Besides, some research introduced mathematical methods to estimate the shape of the target fruit. [258] proposed an active deep sensing method to handle occlusions in clustered and single fruit scenarios, utilising a deep network to predict optimal observation positions, and guiding robots to avoid the occlusion. [161] mitigated the challenge of fruit occlusion in complex environments

by leveraging approximately spherical fruit shape priors for improved segmentation and localisation, enabling effective occlusion-aware solutions without reliance on additional data or equipment.

However, all of the above research limits the addressed problem to either classifying the occlusion categories or estimating the shape of the occluded fruit. Taking a step forward, this chapter proposes a self-supervised method to reconstruct the details of the occluded parts of the fruits.

### 6.1.3 Contributions

To address the occlusion and ripeness-labelling challenges, this chapter proposes AppleSSL, a self-supervised method that leverages a small number of labelled examples (less than 1%) and a large pool of unlabelled apple images collected in natural orchard conditions. The method comprises three key components: a reconstructor trained to infer missing apple details in occluded images; a feature extractor designed to learn ripeness-relevant representations from unlabelled data; and a predictor that outputs a continuous ripeness score without requiring rigid classification boundaries.

This chapter presents the most novel contribution of the thesis, offering a flexible, data-efficient, and occlusion-aware solution for apple ripeness estimation in the field. By combining reconstruction, feature learning, and regression-based prediction within a unified self-supervised framework, AppleSSL bridges the gap between the highly controlled assumptions of earlier models and the complex visual challenges encountered in real orchard settings. Its design supports downstream deployment in robotic harvesting systems and large-scale orchard monitoring, thereby contributing to the broader goals of smart and precision agriculture.



Figure 6.4: The apple orchard in New Zealand (left) and samples of apple images (right).



## 6.2 Dataset

### 6.2.1 Image Collection

A number of 2530 apple images ( $4032 \times 3024$  pixels) were captured with a mobile phone in a large Jazz apple orchard located near Hawke’s Bay, New Zealand. The overview of the orchard and samples of the apple images are presented in Fig. 6.4. The collection took several weeks from February to March in 2024, and encompassed the complete apple ripening process from fully unripe to fully ripe.

There were no specific requirements for the image collection. All apple images were taken under natural illumination and in real-world production settings, taken from various angles to simulate every possible scenario for the in-field operation of robots. As a result, the apples exhibited variations such as being isolated, in close proximity to each other, and partially obscured by leaves or stalks.

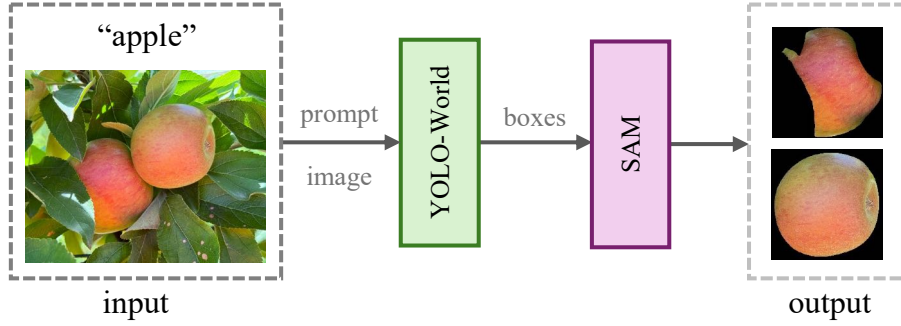


Figure 6.5: The workflow of image preprocessing.

### 6.2.2 Image Preprocessing

YOLO-World [41] is applied to detect the bounding boxes of apples, and then the boxes are used as the input of Segment Anything Model (SAM, 139) to perform the apple instance segmentation. The workflow of the process is shown in Fig. 6.5. The dataset consists of 2530 images, from which 7191 apple instances were detected and segmented following the workflow. From these images, 20 fully unripe and 20 fully ripe apples under diverse conditions were manually selected, using them as labelled instances, as illustrated in Fig. 6.6, while the remaining 7151 apple instances are unlabelled.

The foreground ratio  $Fr$  of all uniformly resized apple instances is computed using Eq. (6.1), where  $N_{apple}$  represents the number of pixels corresponding to apples, and  $N_{img}$  is the total number of pixels in the image.

$$Fr = \frac{N_{apple}}{N_{img}} \quad (6.1)$$



Figure 6.6: The selected 20 fully unripe and 20 fully ripe apples.

The distribution of  $Fr$  is presented in Fig. 6.7. Here, apples with  $Fr \geq 0.6$  are defined as “complete” apples, as they contain sufficient visual information for analysis. In contrast, apples with  $Fr < 0.6$  are categorised as “incomplete” apples, as substantial portions of the apple are occluded, resulting in limited details.

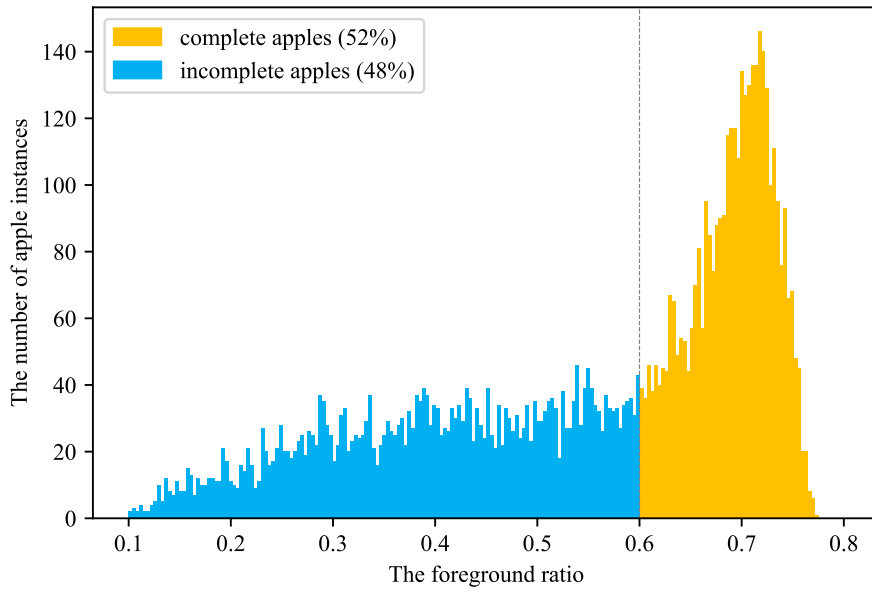


Figure 6.7: The  $Fr$  distribution of the dataset.

### 6.2.3 Image Augmentation

Image augmentation involves applying various transformations to images to artificially increase the size of a dataset and simulate real-world conditions.

For some of the self-supervised learning methods, image augmentation is a cornerstone of training strategies. It serves as a key mechanism to manipulate input data, ensuring that the model learns meaningful representations from a large number of unlabelled data.

Based on the collected apple images, it is assumed that some in-field conditions observed in apples, such as variations in brightness, shadows, viewing angles, and occlusions

caused by leaves, branches, or other fruits, can be regarded as forms of 'natural augmentation'. These natural augmentations do not influence the ripeness of the apples, as ripeness is an intrinsic quality independent of external conditions.

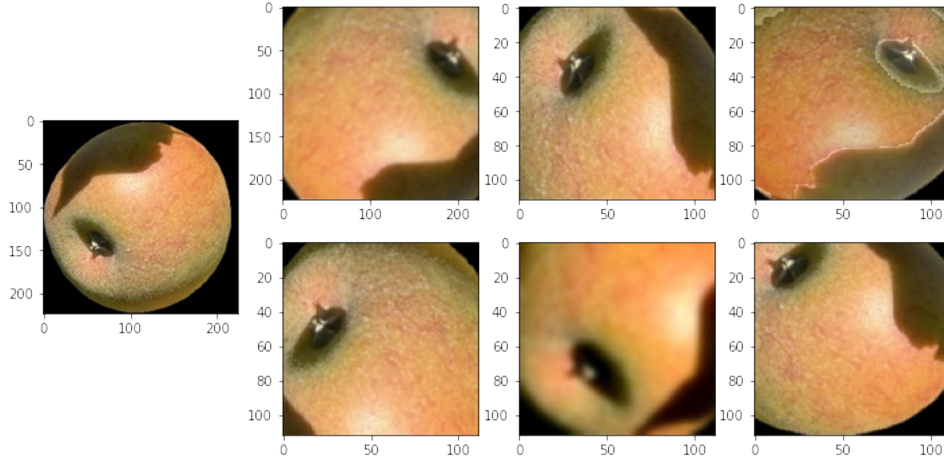


Figure 6.8: The original image (left) and examples generated via augmentation (right).

In this work, a variety of artificial augmentation methods are incorporated, including random cropping, random scaling, random flipping, brightness adjustment, colour jittering and Gaussian blur to simulate natural augmentations. For instance, random cropping and flipping mimic the perspectives of images captured from different angles, while Gaussian blur replicates the effect of images taken when the camera is out of focus on the apples. It is noted that gray-scale conversion is not used in this study, as it results in the loss of colour information. The illustration of augmentations is provided in Fig. 6.8.

By setting different probabilities to each method, a diverse set of variations is generated, enabling the model to robustly learn meaningful features associated with ripeness across different scenarios.

## 6.3 Method

### 6.3.1 Overview

The overall architecture of this study is shown in Fig. 6.9. The collected images first undergo a preprocessing stage, including object detection and instance segmentation. Following this, the apple instances are partitioned based on two criteria: (1) whether they are labelled and (2) whether they are complete or incomplete. Complete apples are utilised for feature extraction and reconstruction, and incomplete apples are used for reconstruction. Finally, labelled apples serve as boundaries for projecting the features onto the final ripeness prediction.

The architecture of the proposed AppleSSL is illustrated in Fig. 6.10. It contains three parts: a missing-part reconstructor, a feature extractor, and a ripeness score predictor.



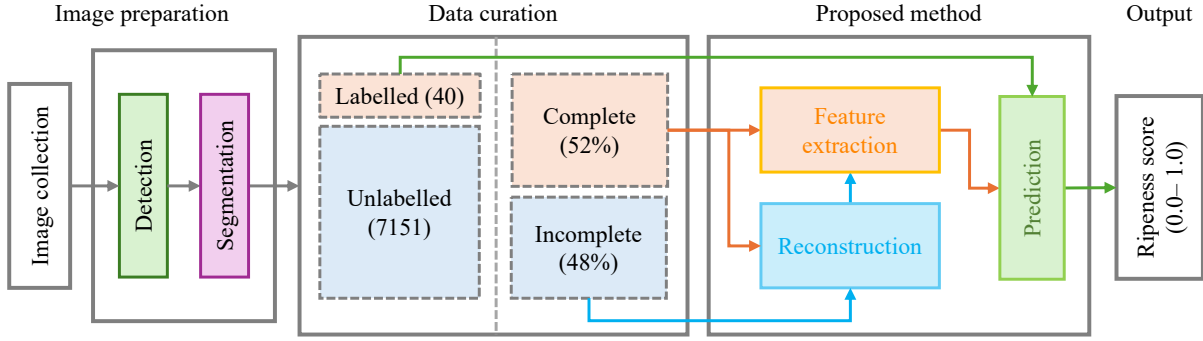


Figure 6.9: The overall architecture of this study.

- **Reconstructor**

The reconstructor is a self-supervised component designed for incomplete apples, which aims to reconstruct missing parts of apples to provide more details.

- **Extractor**

The extractor also operates within a self-supervised paradigm to learn representations related to ripeness from images. Specifically, it is expected to find a feature space in which every apple is separated by its ripeness, and unripe apples are as far as possible from ripe apples.

- **Predictor**

The predictor is a simple Multi-Layer Perceptron (MLP), which takes features from the extractor as input and predicts ripeness scores.

### 6.3.2 Reconstructor

The reconstructor is based on ‘masked image modelling’, which learns by masking portions of the input image and predicting the missing parts. In this context, occlusions caused by leaves or trunks are considered a kind of natural mask, and the task is to reconstruct these occluded apples.

Specifically, the reconstructor employs the SimMIM [294], which consists of an encoder that maps the normalised image to a latent representation and a prediction head that reconstructs the reconstructed image from the latent representation. The illustration is presented in Fig. 6.10.

Given an input image, it is divided into regular and non-overlapping patches. A subset of patches is selected, while the remaining ones are masked. The encoder embeds the visible patches using a linear projection with added positional embeddings and processes them through a series of Transformer blocks. It is noted that the encoder operates exclusively on visible, unmasked patches, as masked patches are removed, and no mask tokens are used. The encoder extracts a latent feature representation of the masked image, which is utilised to predict the original signals in the masked regions. For the

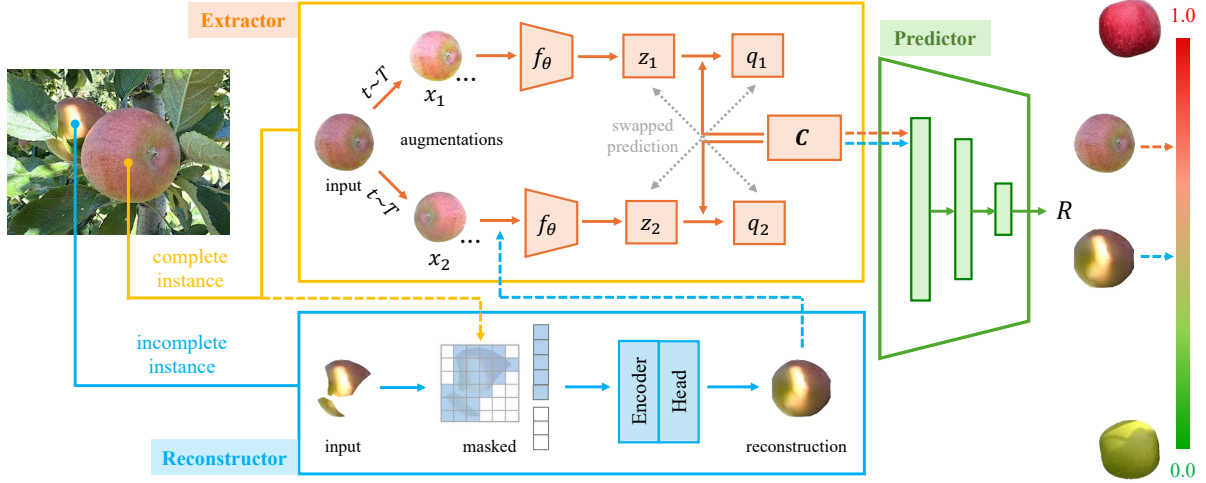


Figure 6.10: The architecture of the proposed AppleSSL.

encoder, two common vision Transformer architectures, Vision Transformer (ViT) and Swin Transformer (SwinT), are considered.

The prediction head processes the latent feature representation to generate a form of the original signals for the masked regions. While the prediction head can have an arbitrary form and capacity, a single  $1 \times 1$  convolutional layer is employed to maintain a small model size. Each output element from the prediction head is a vector of pixel values corresponding to a patch. The final layer of the decoder is a linear projection with the number of output channels equal to the pixel count in a patch. The output of the prediction head is then reshaped to reconstruct the image.

The Mask Autoencoder (MAE, 103) is another state-of-the-art model of masked image modelling, which takes a complete ViT architecture for both the encoder and prediction head. MAE demonstrates that random sampling with a high masking ratio significantly reduces redundancy, creating a task that cannot be easily solved by extrapolation from visible neighbouring patches. Accordingly, the reconstructor adopts a strategy of random masking with a 75% masking ratio, meaning 75% of the input image patches are masked, leaving only 25% visible for the model.

### Training Details

During training, the pre-trained models are fine-tuned on complete apple instances to save training time.

The loss function calculates the Mean Squared Error (MSE) loss between the reconstructed and original images by measuring the average squared difference between their pixel values. It is defined as in Eq. (6.2).

$$\text{MSE}(\mathbf{x}, \mathbf{y}) = \frac{1}{\Omega(\mathbf{x}_M)} \|\mathbf{y}_M - \mathbf{x}_M\|_2^2 \quad (6.2)$$

where  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{3 \times H \times W}$  are the original RGB values and the predicted values, respectively;  $M$  indicates the set of masked pixels;  $\Omega(\cdot)$  is the number of elements.

## Evaluation Details

During the evaluation, another two metrics are introduced to evaluate the reconstruction quality in de-normalised colour values.

- **Peak-Signal-to-Noise Ratio (PSNR)**

PSNR [113] is a widely used metric for evaluating the quality of image reconstruction in computer vision. It measures the similarity between the original and reconstructed images by comparing the ratio of peak signal to noise on a logarithmic scale. PSNR is defined as in Eq. (6.3), where 255 is the maximum pixel value for 8-bit images. A higher PSNR indicates that the reconstructed image is closer to the original, indicating better quality. Conversely, a lower PSNR indicates greater numerical differences between the images, reflecting poorer quality.

$$\text{PSNR}(\mathbf{x}, \mathbf{y}) = 10 \cdot \log_{10}\left(\frac{255^2}{\text{MSE}(\mathbf{x}, \mathbf{y})}\right) \quad (6.3)$$

- **Structural Similarity Index Measure (SSIM)**

SSIM [280] is another well-known metric used to measure the structural similarity between the original and reconstructed images. It focuses on comparing structural information in images, including luminance, contrast, and texture, which aligns more closely with human visual perception. The definition of SSIM is given in Eq. (6.4).

$$\text{SSIM}(\mathbf{x}, \mathbf{y}) = \frac{(2\mu_{\mathbf{x}}\mu_{\mathbf{y}} + C_1)(2\sigma_{\mathbf{xy}} + C_2)}{(\mu_{\mathbf{x}}^2 + \mu_{\mathbf{y}}^2 + C_1)(\sigma_{\mathbf{x}}^2 + \sigma_{\mathbf{y}}^2 + C_2)} \quad (6.4)$$

where  $\mu_{\mathbf{x}}, \mu_{\mathbf{y}}$  and  $\sigma_{\mathbf{x}}^2, \sigma_{\mathbf{y}}^2$  are the average luminance and variance of the original and reconstructed images.  $\sigma_{\mathbf{xy}}$  is the covariance between two images.  $C_1$  and  $C_2$  are small constants to avoid a zero denominator. The SSIM value ranges from  $[-1, 1]$ , and a higher value represents a more accurate replication of the original image.

### 6.3.3 Extractor

The feature extractor is implemented in a self-supervised learning framework, using the online-clustering method SwAV [28]. This method employs two parallel branches to facilitate feature learning. Specifically, the feature extractor is designed to identify representations associated with apple ripeness. The goal is to find a feature space in which fully unripe apples are positioned farthest from fully ripe apples, while ensuring that

a random given apple image and its augmented variants are mapped to closely aligned locations. An overview of this process is presented in Fig. 6.10.

The input image is transformed into multiple augmented views  $\mathbf{x}_{nt}$  (e.g.,  $x_1$  and  $x_2$ ) using transformations  $t$  sampled from a set  $\mathcal{T}$  of image augmentation techniques.

These augmented views  $\mathbf{x}_{nt}$  are then passed through an encoder  $f_\theta$ , which consists of two standard convolutional layers, to generate non-linear feature representations  $\mathbf{z}_{nt}$  (e.g.,  $z_1$  and  $z_2$ ). Then, the feature representations are normalised using  $\ell_2$  normalisation and projected onto the unit sphere.

Next, a code  $\mathbf{q}_{nt}$  (e.g.,  $q_1$  and  $q_2$ ) is computed by mapping the feature  $\mathbf{z}_{nt}$  to a set of prototypes  $\mathbf{C}$ . The prototype  $\mathbf{C}$  consists a set of  $K$  trainable vectors, denoted as  $\{\mathbf{c}_1, \dots, \mathbf{c}_K\}$ . In this work,  $\mathbf{C}$  is represented as a matrix whose columns correspond to the prototype vectors  $\mathbf{c}_1, \dots, \mathbf{c}_K$ . These prototypes are treated as model parameters and are updated iteratively during the training process.

In detail, a code is computed for one augmented version of an image and predicted from other augmented versions of the same image. Given two feature vectors,  $\mathbf{z}_t$  and  $\mathbf{z}_s$ , derived from different augmentations of the same image, their corresponding codes  $\mathbf{q}_t$  and  $\mathbf{q}_s$  are obtained by matching these features to a set of  $K$  prototype vectors,  $\{\mathbf{c}_1, \dots, \mathbf{c}_K\}$ . The computation involves multiplying the feature vector  $\mathbf{z}_{nt}$  with the prototype matrix  $\mathbf{C}$ , followed by applying the Sinkhorn-Knopp algorithm to normalise the result and produce the code  $\mathbf{q}_{nt}$ .

The prototype vectors represent the clustering centres of the apple images. As this method is an online method, the codes are updated only based on the image features within the current batch, distinguishing this method from offline clustering approaches that require the entire dataset to compute the codes. The loss function is defined in Eq. (6.5).

$$L(\mathbf{z}_t, \mathbf{z}_s) = \ell(\mathbf{z}_t, \mathbf{q}_s) + \ell(\mathbf{z}_s, \mathbf{q}_t) \quad (6.5)$$

where the function  $\ell(\mathbf{z}, \mathbf{q})$  quantifies the alignment between features  $\mathbf{z}$  and a code  $\mathbf{q}$ . Conceptually, the method evaluates the similarity between the features  $\mathbf{z}_t$  and  $\mathbf{z}_s$  using the intermediate codes  $\mathbf{q}_t$  and  $\mathbf{q}_s$ . In other words, if these two features are from augmentations of the same input image, and they encode the same or similar information, then it should be feasible to predict the code from the other feature.

The loss function in Eq. (6.5) consists of two terms that define the “swapped” prediction task: predicting the code  $\mathbf{q}_t$  from the feature  $\mathbf{z}_s$ , and vice versa, predicting  $\mathbf{q}_s$  from  $\mathbf{z}_t$ . Each term corresponds to the cross-entropy loss between the predicted code and the probability distribution obtained by applying softmax function to the dot products of  $\mathbf{z}_i$  and all prototypes in  $\mathbf{C}$ . The loss formulation is detailed in Eq. (6.6), where  $\tau$  is a temperature parameter that controls the sharpness of the softmax distribution.

$$\ell(\mathbf{z}_t, \mathbf{q}_s) = - \sum_k \mathbf{q}_s^{(k)} \log \mathbf{p}_t^{(k)}, \quad \mathbf{p}_t^{(k)} = \frac{\exp\left(\frac{1}{\tau} \mathbf{z}_t^\top \mathbf{c}_k\right)}{\sum_{k'} \exp\left(\frac{1}{\tau} \mathbf{z}_t^\top \mathbf{c}_{k'}\right)} \quad (6.6)$$

In contrast to previous self-supervised learning methods, which directly compare the similarity of feature vectors  $\mathbf{z}_{nt}$ . Comparing high-dimensional features (e.g. 2048) usually takes a lot of time and computational overhead. Instead, this study focuses on comparing the codes  $\mathbf{q}_{nt}$  derived from different views, aiming to make them consistent. This strategy allows the model to capture more details of the input. In this work, the code is chosen as the output of the feature extractor, as it provides a more efficient and effective representation for comparison.

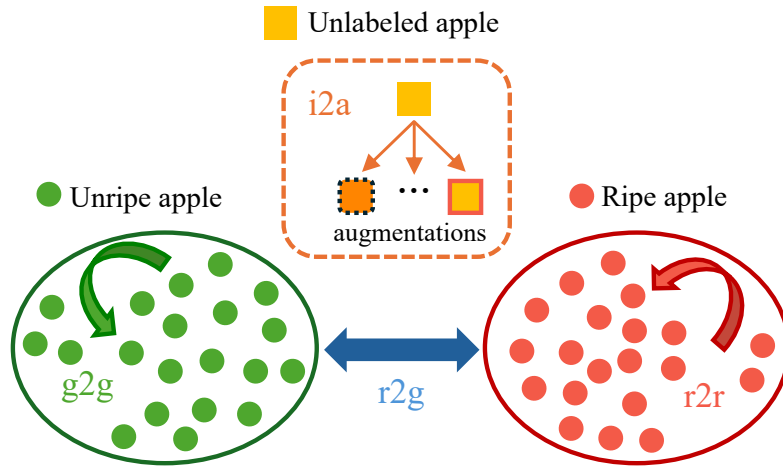


Figure 6.11: The proposed distances for model performance evaluation.

Inspired by SMOG [209], the similarity comparison can happen at the instance-level, and also at the group-level. Building on this idea, two distances are conceptualised as metrics to make the extractor more suitable for the apple ripeness determination task. The considered distances are illustrated in Fig. 6.11.

$$D = A \sum_{i=1}^{N_G} \sum_{j \geq k}^{N_E} \left(1 - \frac{f_i \cdot f_j}{\|f_i\|_2 \|f_j\|_2}\right) \quad (6.7)$$

The definition of the distance is given in Eq. (6.7), where  $f$  is the extracted feature and  $\|f\|_2$  is the Euclidean norm of  $f$ ,  $A$  is a constant.  $k$  is a variable and  $N$  denotes an ordered set. Specifically,  $N_G$  is defined as the set of labelled fully unripe apples and  $N_E$  as the set of labelled fully ripe apples. The sizes of  $N_G$  and  $N_E$  are 20 in this chapter.

- **Intra-distance**

For a random unlabelled apple, it is expected that the distance between the image and its augmentations should be as small as possible. This ensures that the image and its augmentations occupy a stable position between unripe and ripe items in

the feature space. This intra-distance, denoted as  $D_{i2a}$ , is implicitly considered by the loss Eq. (6.6).

For the set of labelled apples, it is assumed that unripe items should be closest to other unripe items, and ripe items should be closest to other ripe items in the feature space. To quantify this, it is defined as:

- The average distance between labelled unripe apples as  $D = D_{g2g}$ , where  $i, j \in N_G, A = \frac{2}{|N_G|(|N_G| + 1)}, k = i$ .
- The average distance between labelled ripe apples as  $D = D_{r2r}$ , where  $i, j \in N_E, A = \frac{2}{|N_E|(|N_E| + 1)}, k = i$ .

- **Inter-distance**

For labelled unripe and ripe apple images, unripe items should be as distant as possible from ripe items in the feature space. To quantify this separation, the average group-level distance between labelled unripe and labelled ripe apples is computed, denoted as:  $D = D_{r2g}$ , where  $i \in N_G, j \in N_E, A = \frac{1}{|N_G||N_E|}, k = 1$ .

Intra-distances evaluate the clustering consistency within each apple and its variants in the feature space. Inter-distance measures the degree of separation between the two labelled groups, while also providing insight into the depth of the feature space. The computation of these two distances serves as a complement to the “swapped” prediction loss, offering additional metrics for assessing the effectiveness of the learned representations. This combination is also particularly useful for comparing the performance of different self-supervised learning methods.

## Training Details

In this chapter, the two views consist of a global view (high-resolution,  $224 \times 224$  pixels) and a local view (low-resolution,  $112 \times 112$  pixels) augmentation. The extractors are trained from scratch on a set of complete apples. The backbone of the extractor is Res18 [106] to save the model size. The dimension of the output feature is set to 256, the number of prototypes is 512, the temperature  $\tau$  is set to 0.1, and the number of Sinkhorn-Knopp iterations is set to 3. No pre-trained weights are used, and the parameters of all convolution layers are initialised by a normal distribution.

## Evaluation Details

During the evaluation, the  $D_{r2r}$ ,  $D_{g2g}$  and  $D_{r2g}$  are reported, each bounded within the range  $[0, 2]$ . Ideally, lower values of  $D_{r2r}$  and  $D_{g2g}$  indicate promising performance, as they reflect the extractor’s ability to effectively process the labelled images under various

augmentations. Besides,  $D_{r2g}$  is expected to be significantly greater than  $D_{r2r}$  and  $D_{g2g}$ , indicating that unripe apples from ripe apples are successfully separated in the feature space. To help better compare the results, a simple distance difference is computed, defined as  $(D_{r2g} - D_{r2r} - D_{g2g})$ , where higher values indicate better overall separation.

These three distances serve as metrics to evaluate how closely the extractor aligns with the proposed aim outlined in 6.3.1. Specifically, extractors generate high-dimensional features instead of final outputs. If the extractor has lower  $D_{r2r}$ ,  $D_{g2g}$  values and a higher  $D_{r2g}$  value, then it is promising but does not promise to produce better final results. Because high-dimensional features are then processed by the predictor for final results, the design of the predictor is also a big factor that influences final results.

### 6.3.4 Predictor

A simple 3-layer MLP predictor is employed to predict the ripeness score from the extracted features. The network consists of three fully connected layers, with dimensions set to  $[N, 128, 100, 1]$ , where  $N$  represents the feature dimension from the extractor. Each layer, except the final one, is followed by a ReLU activation function. The final layer is a fully connected output layer with a single neuron, which produces the ripeness score  $R$ . This score is then normalised to fall within the range  $[0.0, 1.0]$ . This architecture effectively reduces the dimensionality from input space to a single scalar value while leveraging ReLU non-linearity to capture complex relationships between the features, ensuring robust and accurate predictions. The illustration is shown in Fig. 6.10.

### Training Details

During training, the weights of the feature extractors are frozen, and only the weights of the predictor are updated. The predictor is trained from scratch using the labelled images only. The loss function calculates the MSE between the one-hot encoded predictions and the ground truths from labelled images.

### Evaluation Details

The mean values  $\bar{x}_{green}$  and  $\bar{x}_{red}$ , along with the variances  $s_{green}^2$  and  $s_{red}^2$  of labelled fully unripe and fully ripe apples are selected as evaluation metrics. Ideally, the model is expected to predict a score of 0.0 for fully unripe apples and 1.0 for fully ripe apples. These metrics align with human sense, where higher values correspond to riper apples.

The range of prediction values indicates that the apple ripeness prediction is treated as a regression task rather than a multi-class classification task. As a result, the AppleSSL generates continuous predictions instead of discrete ones. It avoids the inherent discontinuities of discrete classification and allows for a smooth representation of the apple ripeness distribution, providing a more nuanced understanding of ripeness levels.



Additionally and importantly, the distribution of ripeness score predictions is plotted along with dense apple images, and these predictions on the extracted features are visualised for better interpretation.

## 6.4 Experiments and Results

### 6.4.1 Experiments

In this chapter, experiments are conducted based on PyTorch Lightning 2.0.0 [66] and have been carried out using Python 3.9.13 and PyTorch 1.13 on a computer with the specifications shown in Table 3.4..

An SGD optimiser was employed with a weight decay of  $5 \times 10^{-5}$  and a momentum of 0.9. Different initial learning rates, ranging from 0.0001 to 0.06 were explored across different models to identify the optimal value for achieving the best performance.

The default patience setting for the reconstructor and extractor is set to 30 epochs to optimise training time, meaning the model will terminate training if no improvement in metrics is observed after 30 epochs. In contrast, the patience for the predictor is set to 3 epochs to minimise the risk of overfitting.

Table 6.1: The results of reconstruction.

	Model	Image/Mask Size	PSNR $\uparrow$ (dB)	SSIM $\uparrow$	Params (M)
MAE	ViT-B	224/16	25.14	0.73	111
		224/32	22.00	0.67	
	ViT-L	224/16	<b>25.71</b>	0.74	329
		224/32	21.24	0.67	
SimMIM	SwinT	192 $^\dagger$ /16	24.40	0.74	89.9
		192 $^\dagger$ /32	21.47	0.72	
	ViT-B (AppleSSL)	<b>224/16</b>	<b>25.36</b>	<b>0.75</b>	<b>86.3</b>
		224/32	21.27	0.69	

$^\dagger$  follows the pre-trained SwinT setting with a window size of 6.

### 6.4.2 Reconstructor

#### Comparison

The proposed reconstructor is compared with MAE. The numerical results and visual comparison are shown in Table 6.1 and Fig. 6.12.

For MAE models, ViT-Base (ViT-B) with a mask size of 16 achieves a PSNR of 25.14 and an SSIM of 0.73. When the model size increases to ViT-Large (ViT-L), the performance improves, with ViT-L achieving the highest PSNR of 25.71 and an SSIM of 0.74. However, this improvement comes at the cost of significantly larger parameters, increasing from 111M to 329M.

For SimMIM models, the proposed reconstructor with ViT-B achieves the highest SSIM of 0.75 and the second-highest PSNR of 25.36, while utilising only 86.3M parameters. Notably, the proposed reconstructor has the smallest parameter count, requiring less than one-third of the parameters of MAE with ViT-L, but delivering very comparable performance.

From the visual comparison, it is observed that with the same input image and masking strategy, ViT-L produces the best reconstructions, while ViT-B delivers similar but reasonable results.

It is well-known that larger models deliver better performance, as they can learn and store more information. However, the small performance difference observed here is acceptable when considering the significant disparity in model size. Increasing the model size excessively for tiny marginal performance gains is not a practical choice for this study.

Compared to the standard ViT, using SwinT as the backbone yields inferior performance in this study. It is hypothesised that this is due to the hierarchical structure of the SwinT, which processes image patches locally using smaller patches and gradually expands the receptive field. This local processing may disrupt the consistency of information within the expanded receptive field, as illustrated in the first row of Fig. 6.12.

The results highlight the significant impact of mask size on performance, with larger mask sizes consistently leading to degradation across all models. The original SimMIM identifies a mask size of 32 as optimal, but based on the experiments, the performance drops substantially with a mask size of 32 compared to 16. A mask size of 16 proves to be the most suitable for reconstructing missing apple parts. It is suggested that a mask size of 32 lacks flexibility, as it is too large to effectively cover the missing patches and introduces excessive noise into the visible patches.

Overall, the proposed reconstructor achieves a favourable balance between performance and efficiency, providing valuable information for subsequent ripeness prediction.

## Visualisation

Then, the reconstructor is tested with incomplete apple images under different settings. The visualisation is shown in Fig. 6.13. The ground truths of these input images are unknown, but the detailed progress for each reconstruction is shown in the visualisations.

The various cases show diverse environmental and lighting conditions affecting the visibility and appearance of apples:

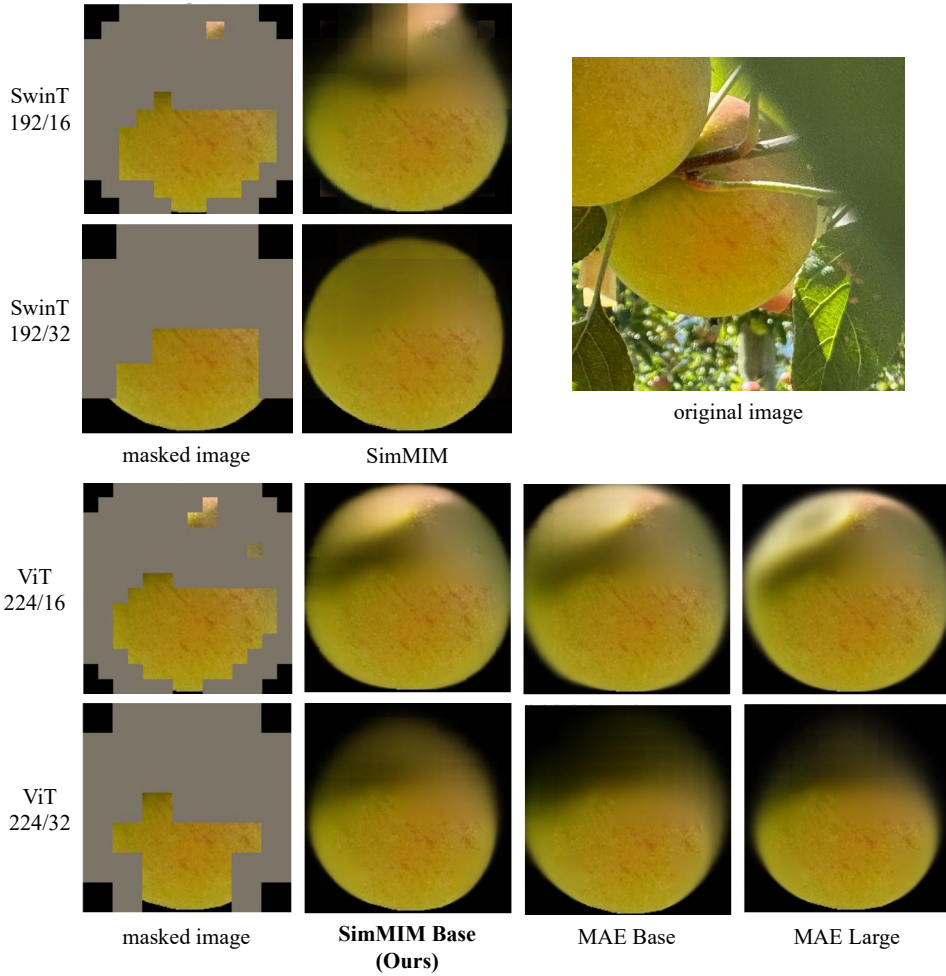


Figure 6.12: The reconstruction comparison using different models and mask sizes.

- Very limited visibility

In cases (a) and (c), the majority of the apples are obscured, resulting in visible rates of less than 30%.

- Different occlusion sources

In case (g), the apple is hidden by the trunk, while apples in other cases are covered by leaves.

- Lighting conditions

In cases (a), (c), (i), and (j), the apples are shaded from direct sunlight, while in cases (d), (e), (k), and (l), they are exposed to direct sunlight.

- Shadows and light patterns

In cases (e), (g), (h), and (k), direct shadows, light-stripes or light-spots are observed on the apples, creating complex light patterns.

- High contrast conditions

In cases (e), (k), and (l), the apples exhibit strong contrasts between light and



Figure 6.13: The visualisation of reconstruction, the numbers in masked input indicate visible rates for the model. Detailed analysis of (a)~(l) are in 6.4.2.



shadow, presenting challenging illumination scenarios.

- Backlighting effects

In cases (b) and (g), the apples are positioned against the light source, resulting in unique lighting angles and potential silhouette effects.

- Uniform colour

In cases (a), (c), (d), and (f), the apples are predominantly of a single colour.

- Gradual colour transitions

In cases (b), (e), and (h), the apples showcase significant continuous colour variations, introducing additional complexity in visual features.

The proposed reconstructor demonstrates its reliable ability to effectively predict occluded apple parts under various conditions, including different illumination levels, occlusions, and ripeness stages in the above cases.

It is suggested that the model trained on a diverse set of apple images in various settings is able to accurately predict the occluded parts of incomplete apples. This enables the use of the trained model to reconstruct missing parts without the need for manually designed fruit shapes or handcrafted features.

### 6.4.3 Extractor

The extractor serves as a critical component of the proposed method, acting as a bridge between the input images and the predictor. To evaluate the performance, the proposed extractor is compared against 15 other self-supervised methods and a supervised binary classification model. For the binary classification model, MSE loss is employed as the loss function, while the self-supervised methods utilise their respective original loss functions, including negative cosine similarity loss, normalised temperature-scaled cross-entropy loss (NT-Xent loss), and other customised loss functions. The comparative results are presented in Table 6.2.

Res18 is selected as the backbone for most of the self-supervised methods, as it is more lightweight compared to the commonly used Res50. For MSN and PMSN, ViT-Small is used, following their respective model designs. The output dimensions for each method are kept consistent with their original configurations.

The results demonstrate that supervised binary classification and several self-supervised methods demonstrate strong performance in separating fully unripe and fully ripe apples within the feature space. However, certain self-supervised methods, such as DCL, DCLW, MSN, PMSN, and VICRegL, fail to meet expectations for this task. Their  $D_{r2g}$  values are smaller than  $D_{r2r}$  and  $D_{g2g}$ , indicating an insufficient separation between unripe and ripe apples, thus they are excluded from being incorporated with the predictor.

Table 6.2: The distance results of different extractors.

Extractor	Backbone	Loss	Dimension	$D_{r2r} \downarrow$	$D_{g2g} \downarrow$	$D_{r2g} \uparrow$	$(D_{r2g} - D_{r2r} - D_{g2g}) \uparrow$	Params (M)
Binary <sup>1</sup>	Res18	MSE loss	512	0.1488	0.0609	0.2391	0.0293	11.2
BYOL [90]	Res18	NCS <sup>2</sup> loss	256	0.0002	0.0036	0.0038	-0.0001	12.5
FastSiam [221]	Res18	NCS loss	128	0.0504	0.0032	0.1370	0.0833	11.8
SimSiam [38]	Res18	NCS loss	256	0.0118	0.1594	0.7084	<b>0.5373</b>	13.5
DenseCL [278]	Res18	NT-Xent loss	2048	0.2488	0.2035	0.6969	0.2446	23.7
MoCo [104]	Res18	NT-Xent loss	512	0.5583	0.8731	0.9668	-0.4646	11.5
NNCLR [62]	Res18	NT-Xent loss	128	0.3879	0.8371	<b>1.1086</b>	-0.1163	11.9
SimCLR [36]	Res18	NT-Xent loss	512	0.2769	0.2366	0.6583	0.1448	11.5
DCL [300]	Res18	DCL loss	512	0.8192	1.0193	0.7207	-1.1179	11.5
DCLW [300]	Res18	DCL weighted loss	512	1.0032	1.4614	0.8880	-1.5767	11.5
DINO [29]	Res18	DINO loss	2048	0.5046	0.2689	1.1046	0.3311	23.7
MSN [5]	ViT-S	MSN loss	256	0.6997	1.1675	1.0978	-0.7694	27.8
PMSN [4]	ViT-S	PMSN loss	384	<b>0.0001</b>	<b>0.0003</b>	0.0001	-0.0003	27.8
TiCo [322]	Res18	TiCo loss	256	0.4499	0.3836	0.5986	-0.2349	23.9
VICReg [11]	Res18	VICReg loss	512	0.1828	0.1722	0.1849	-0.1701	16.4
VICRegL [12]	Res18	VICRegL loss	2048	0.6578	0.6641	0.6012	-0.7208	20.7
SwAV(AppleSSL)	Res18	SwAV loss	256	<b>0.3816</b>	<b>0.2418</b>	<b>0.8844</b>	<b>0.2610</b>	<b>11.7</b>

<sup>1</sup> Binary classification is the only supervised model, using features extracted from the layer before the fully connected layer.<sup>2</sup> NCS: Negative cosine similarity.

The binary classification model achieves the  $D_{r2g}$  (0.2391) greater than both  $D_{r2r}$  (0.1488) and  $D_{g2g}$  (0.0609). These results suggest that unripe apples are distributed more densely than ripe apples. The proposed method achieves the  $D_{r2g}$  of 0.8844, which is significantly greater than  $D_{g2g}$  (0.3816) and  $D_{r2r}$  (0.2418), demonstrating a better balance between the clustering of ripe and unripe apples compared to binary classification.

While PMSN achieves the smallest  $D_{g2g}$  and  $D_{r2r}$ , its  $D_{r2g}$  equals  $D_{r2r}$ , indicating that it does not effectively separate unripe and ripe apples in the feature space. NNCLR achieves the highest  $D_{r2g}$  of 1.1086, but the margin relative to its  $D_{r2r}$  and  $D_{g2g}$  is insufficient to ensure a clear separation.

SimSiam achieves the highest distance difference of 0.5373, with a remarkably low  $D_{r2r}$  of 0.0118. It is noted that DINO also demonstrates a balanced distribution between unripe and ripe apples, reflecting its ability to achieve meaningful separation. In contrast, the binary classification method yields a distance difference of only 0.0293 due to imbalanced  $D_{g2g}$  and  $D_{r2r}$ . The proposed extractor achieves a distance difference of 0.2610, significantly outperforming the binary classification approach by a large margin. It also surpasses several other self-supervised methods, showcasing robust performance in separating unripe and ripe apples.

Regarding model size, introducing complex backbones, such as ViT-Small (ViT-S) with 27.8M parameters, does not bring noticeable improvements. It is suggested that this is because the task of this study is relatively simple, making heavy backbones prone to over-fitting. Additionally, the binary classification model only occupies 11.2M parameters as a result of no extra modules being introduced. The proposed method is with 11.7M parameters, incorporating additional parameters for the extra branch and prototypes **C**. Despite this, the proposed model remains more compact than many other self-supervised methods while delivering superior performance.

## 6.4.4 Predictor

### Comparison

The 12 extractors with  $D_{r2g} > D_{r2r}$  and  $D_{r2g} > D_{g2g}$  were selected to extract image features for the predictor. The performance of the predictor is summarised in Table 6.3. The proposed method demonstrates the best overall performance, achieving the lowest  $\bar{x}_{green}$  of 0.0127 and  $s_{green}^2$  of 0.0001, along with the highest  $\bar{x}_{red}$  of 0.8933 and the second-highest  $s_{red}^2$  of 0.0094. In contrast, the binary classification model yields a  $\bar{x}_{green}$  of 0.2460 and a  $\bar{x}_{red}$  of 0.7258, indicating its comparatively weaker capability in predicting ripeness scores.

The results further highlight that some self-supervised methods outperform the binary classification model. For example, TiCO achieves competitive results with the lowest  $s_{red}^2$  of 0.0034 and the second-lowest  $\bar{x}_{green}$  of 0.0127. DINO delivers a  $\bar{x}_{green}$  of 0.1798 and a



Table 6.3: The results of predictor using features from extractors.

Extractor	$\bar{x}_{green} \downarrow$	$s_{green}^2 \downarrow$	$\bar{x}_{red} \uparrow$	$s_{red}^2 \downarrow$
Binary	0.2460	0.0037	0.7258	0.0098
BYOL	0.0636	0.0017	0.6383	0.0414
FastSiam	0.5336	0.0014	0.8011	0.0052
SimSiam	0.2375	0.0011	0.7302	0.0047
DenseCL	0.3329	0.0031	0.7440	0.0185
MoCo	0.1964	0.0055	0.6536	0.0145
NNCLR	0.1194	0.0012	0.7821	0.0162
SimCLR	0.0607	0.0012	0.7548	0.0169
DINO	0.1798	0.0037	0.8208	0.0161
TiCo	0.0444	0.0005	0.7606	<b>0.0034</b>
VICReg	0.0893	0.0011	0.7121	0.0070
SwAV(AppleSSL)	<b>0.0127</b>	<b>0.0001</b>	<b>0.8933</b>	<b>0.0094</b>

$x_{red}$  of 0.8208. Similarly, VICReg and SimCLR produce relatively low  $\bar{x}_{green}$  values and high  $\bar{x}_{red}$  values.

## Visualisation

To present the results more clearly, the ripeness score predictions are visualised in Fig. 6.14. The analysis of these predictions is conducted from the following three perspectives:

- **Prediction continuity**

The dataset contains apples at various ripeness stages, with 40 labelled fully unripe and fully ripe apples used for training. Consequently, the predictions are expected to span the entire range of scores, from 0.0 (unripe) to 1.0 (ripe), reflecting a continuous progression.

Among the evaluated methods, the proposed approach uniquely achieves seamless and continuous predictions across the entire score range, accurately representing all ripeness stages. Other methods, including NNCLR, DINO, SimCLR, VICReg, and the binary classification model, also approximate full-score predictions but exhibit gaps, with certain score intervals missing in their outputs. This discontinuity indicates limitations in capturing the smooth progression of ripeness.

- **Prediction distribution**

Like many large image datasets, including the previous NinePeach dataset, the apple dataset should exhibit a “long-tail” distribution. This reflects the natural

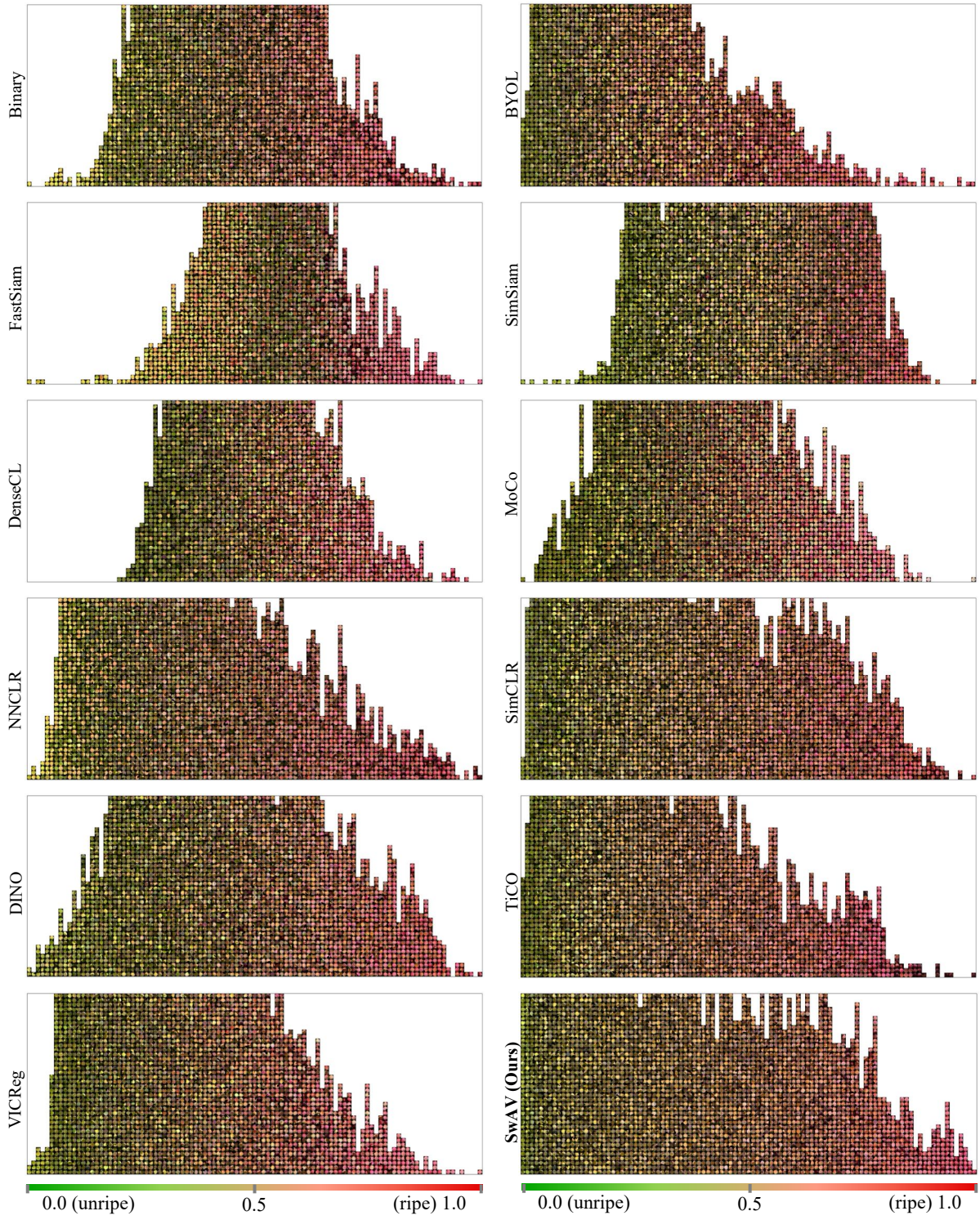


Figure 6.14: Ripeness score  $R$  predictions for complete apple instances, with intervals of 0.1 and at most 40 items displayed per score.

tendency for unripe apples to outnumber ripe ones due to factors such as natural fruit-falling and artificial fruit-thinning.

Several methods, including binary classification, FastSiam, SimSiam, DenseCL, MoCo, and DINO, produce predictions with a Gaussian-like distribution. These

methods do not generate sufficient predictions for unripe apples. Most predictions fall in the semi-ripe range, indicating poor separation between unripe and ripe apples. In contrast, the proposed method, along with BYOL, NNCLR, SimCLR, TiCO, and VICReg, predicts ripeness scores following the expected “long-tail” distribution. The predicted number of apples gradually decreases from unripe to ripe, effectively reflecting the natural progression of apple ripening.

- **Colour gradient**

A smooth colour gradient from unripe to ripe is an essential indicator of the accuracy of ripeness predictions. Ideally, the gradient should transition smoothly from green for unripe apples to red for fully ripe ones.

Some methods, including BYOL, FastSiam, MoCo, and VICReg, exhibit obvious inconsistencies, as some green apples are incorrectly assigned scores over 0.5, suggesting outliers in prediction. SimCLR and TiCO also face challenges, with semi-ripe and ripe apples often mixed, making it difficult to tell. Notably, the proposed method delivers a smooth and consistent colour gradient. The predictions start with green on the left and gradually transition to red on the right, accurately reflecting the natural ripening process. This demonstrates the robustness and precision of AppleSSL in ripeness estimation.

3D Principal Component Analysis (PCA) is used to reduce the dimensionality of the extracted features to three dimensions, with the visualisation presented in Fig. 6.15.

Among all of the visualisations, the proposed predictor stands out by generating a smooth manifold where apple ripeness increases progressively. In the space, the labelled unripe and ripe apples are distinctly separated, indicating high explainability for the ripeness score predictions.

Since ripeness score prediction is a subjective topic, several volunteers including apple-picking robot professionals and normal apple consumers, were invited to help evaluate the performance. They were required to independently choose the best prediction from their perspectives. The test was conducted anonymously, and ground truths were not disclosed before test. All of the participants agreed that the proposed predictor and TiCO are the top-performing methods. However, compared to the proposed predictor, although TiCO shows a good colour gradient, it is unconfident with accurate predictions for ripe apples, as a result of  $\bar{x}_{red}$  of 0.7606.

The results further highlight that self-supervised methods can outperform supervised binary classification. This underscores the ability of self-supervised models to learn latent ripeness-related features from a large number of unlabelled images, significantly reducing the need for manual labelling.



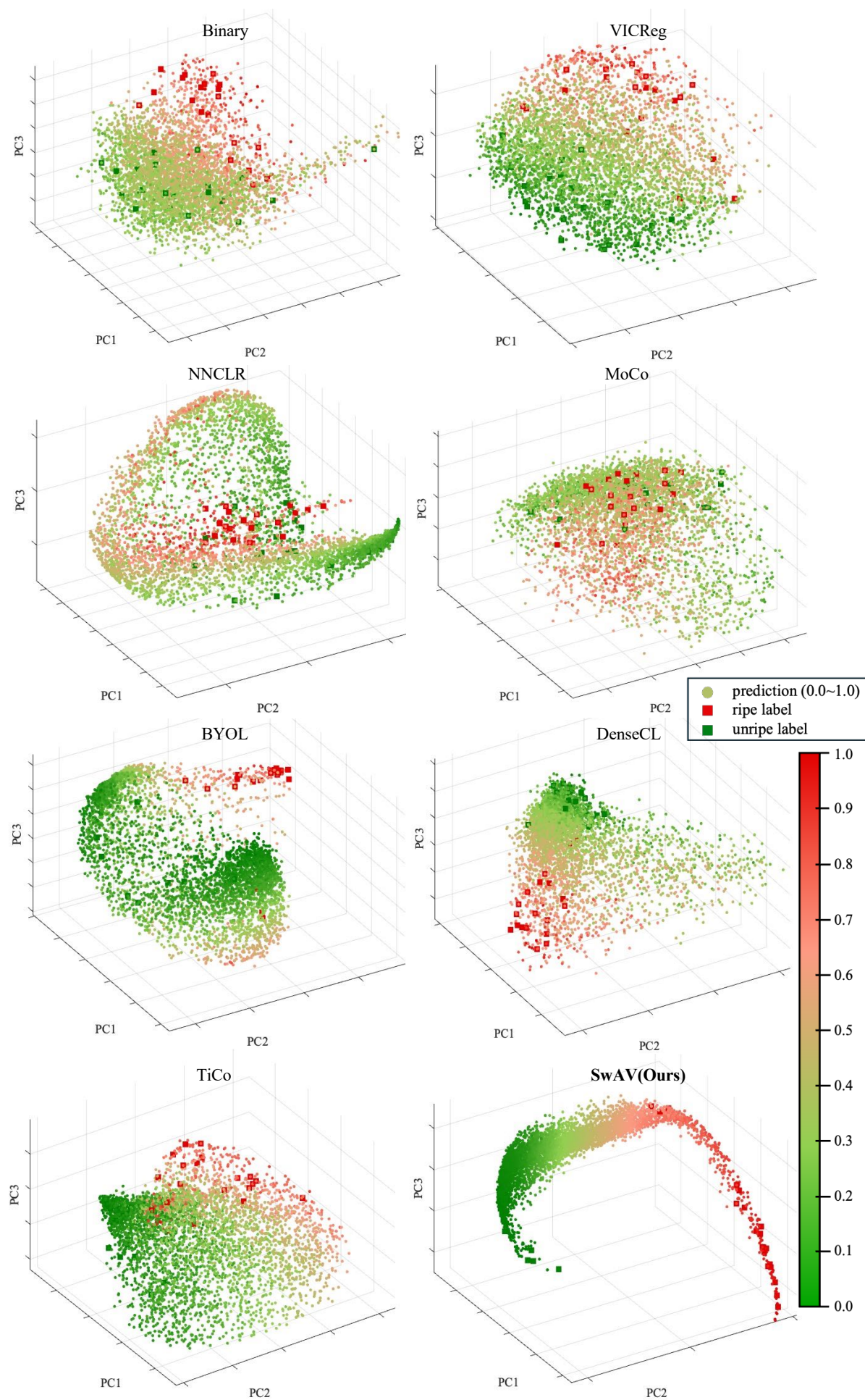


Figure 6.15: 3D PCA visualisations of ripeness scores on extracted features.

## 6.5 Discussion

### 6.5.1 Transfer to Other Fruits

The proposed framework is not specific to apples and can be generalised to other fruits. It is likely to transfer more easily to round fruits such as oranges or apricots, as the reconstructor in this work is based on round shape estimation. Additional adaptation may be needed when applying to non-round fruits like mangoes or strawberries, which have more varied shapes. In such cases, it is necessary to design shape-estimating methods to draw the possible shape of occluded fruits.

Examples of “fully unripe” and “fully ripe” fruits are also required to be specified. If annotations at different ripeness levels (e.g., 25%, 50%, and 75%) are available, they can be added to provide reference milestones. The proposed distance metrics can be easily adjusted to incorporate these levels.

Overall, the framework is designed to minimise reliance on annotations, making it feasible to transfer, particularly for round fruits with clear colour change during ripening.

### 6.5.2 Limitations

The apple images in this study were collected from a single Jazz apple orchard, which may not represent the different varieties of apples. Despite extensive searches, no public datasets that met the research requirements were found. This constraint has led to reliance solely on the collected dataset.



(a) Peduncle and Calyx Prediction Deficiency.

(b) Excessive-Occlusion Prediction Deficiency.

Figure 6.16: Two prediction deficiencies in the proposed reconstructor.

In terms of reconstruction, there are two prediction deficiencies, as shown in Fig. 6.16. The first is peduncle and calyx prediction deficiency, the model cannot predict the apple peduncle and calyx as expected. The second deficiency appears when excessive occlusion occurs, with very limited visible information, the reconstructor can not perform well and generate reasonable results.



AppleSSL is designed for in-field apples that have significant colour changes during their ripening process. Therefore, it is not suitable for certain apple cultivars like Granny Smith, which remain green throughout all ripening stages. Additionally, it cannot be applied to fruits that ripen after harvesting like bananas, or to those evaluated based on softness like avocados.

### 6.5.3 Future Work

To improve the applicability of AppleSSL, the dataset is expected to be expanded by including a more diverse range of apple varieties, capturing a broader representation across different types.

AppleSSL demonstrates that it is feasible to use a single-view image to predict apple ripeness. The next work is proposed to focus on extending this method to work with multi-view images, which would allow more accurate ripeness estimation. This method has the potential to be extended to other fruits that exhibit significant colour changes during the ripening process, such as peaches.



Figure 6.17: The digital simulation of a large orchard, with apple locations and ripeness monitored.

Besides, the proposed method is promising for deployment on in-field robots to capture both the ripeness and spatial information of apples, making it possible to monitor the ripeness distribution across a large orchard. This information can facilitate data-driven decision-making for orchard management and then be used to guide autonomous picking robots to selectively harvest ripe apples. Fig. 6.17 simulates such an apple orchard in a 3D digital environment.

## 6.6 Summary

Developing apple-harvesting robots capable of identifying the ripeness stage of apples is a challenging task, particularly because in-field apples are often obscured by leaves, branches, or trunks. Determining apple ripeness is also challenging as it is subjective to define the number of ripeness stages. Under this context, a novel self-supervised method called AppleSSL is proposed, utilising 40 labelled and 7151 unlabelled apple images for two problems: ripeness determination and in-field occlusion.

AppleSSL consists of three key parts: a reconstructor, a feature extractor, and a predictor. The reconstructor is trained to restore the missing details of occluded apples, enabling more complete visual representations. The feature extractor leverages a vast number of unlabelled images to learn ripeness-related features effectively, reducing the reliance on labelled images. Finally, the predictor uses the extracted features to generate flexible ripeness scores between 0.0 and 1.0, eliminating the need for subjectively pre-defined ripeness stages. This flexibility allows end-users to make customised decisions according to their specific needs and criteria.

Experimental results highlight that AppleSSL achieves the highest SSIM of 0.75 and the second-highest PSNR of 25.36 for reconstructing incomplete apples, with the fewest 86.3M parameters. Besides, AppleSSL outperforms 15 other self-supervised methods and even a supervised method in ripeness score prediction, achieving the lowest score of 0.0127 for fully unripe apples and the highest score of 0.8933 for fully ripe apples.

AppleSSL is promising for integration into in-field robotic systems, enabling them to determine ripeness effectively and selectively harvest only ripe fruits. Furthermore, it can be used to monitor overall ripeness trends across large orchards, helping managers make informed decisions about harvest timing and orchard management. AppleSSL contributes to the goals of smart precision agriculture.



# Chapter 7

## Conclusions and Future Work

### 7.1 Research Summary

Precision agriculture is undergoing rapid development with the help of deep learning and automation. This thesis sets out to explore how deep neural networks can be used to estimate fruit ripeness based on images, aiming to create accurate and efficient models that work directly in the field. Through the development of novel models designed for peaches, strawberries, and apples, this research addresses practical challenges such as heavy occlusion, limited labelled data, and the computational constraints of edge devices.

Looking back at the original objectives, designing high-performance, lightweight models for fruit classification and ripeness grading, this work has largely met its goals. The proposed models demonstrate strong performance across species and conditions, and the focus on efficiency supports future integration into mobile or robotic systems. However, no approach is without its limitations. The reliance on fruit-specific datasets and the need for detailed labels can restrict how easily the proposed models transfer to new fruits or orchard settings. Moreover, while occlusion handling improved considerably, certain edge cases remain challenging, such as peduncle and calyx prediction deficiency.

These reflections suggest areas for future improvement. Making models more general, reducing the need for manual labelling, and improving performance on edge devices will help move this work closer to everyday use in agriculture. With further development, these tools can support growers in making better harvest decisions while saving time and resources.

The comprehensive overview of this research is presented in Table. 7.1.

### 7.2 Key Contributions

This research delivers some contributions to deep learning for precision agriculture, offering practical and efficient solutions for in-field fruit ripeness estimation, as outlined

Table 7.1: The overview of this research.

Chapter	3	4	5	6
Name	NinePeach	StrawDL_Db1	NinePeach+StrawDL_Db1	Apple Images
Images	4599	3100	7499	2530
Catergories	3 (unripe, semiripe, ripe)	1 (no ripeness)	7 in total. peaches (un-, semi-, ripe), strawberries (rs1,-2,-3,-4)	Not predefined
Labels	Fully instance annotation	Fully instance annotation	Fully instance annotation	Manual selection (20 unripe + 20 ripe)
Name	PeachSOLO	LightStraw	FruitQuery	AppleSSL
Type	-	Tiny	xs	s
Performance	72.12 AP	66.82 AP	66.46 AP	67.02 AP
Params (M)	43.93	17.42	10.94	14.08
FLOPs (G)	174.9	78.3	61.53	69.33
Speed (FPS)	11.11	12.29	16.5	16
Strengths	1. 9 peach cultivars 2. Higher precision	1. Embedded device friendly 2. Multiple model choices	1. Combined CNN and Transformer 2. Smaller & faster models	1. Very few labels needed 2. Occluded fruit part reconstruction
Limitations	1. Large model size 2. Predefined ripeness stages	1. No ripeness information 2. Models can be improved	1. Predefined ripeness stages 2. Not real-time inference	1. Based on colour changes 2. Two listed deficiencies

below:

- **PeachSOLO:** Developed a fast, lightweight instance segmentation model for peach ripeness detection under natural occlusion and field complexity. It introduces the NinePeach dataset (4599 images) and uses CBAM attention to achieve a 4.55% AP gain over the baseline, enabling accurate pick-point estimation with lower resource demands than Mask R-CNN.
- **LightStraw:** Proposed a compact CNN for strawberry segmentation on edge devices, combining efficient self-attention and bipartite matching. It achieves an AP of 70.22, over 21 points higher than Mask R-CNN, while significantly reducing parameters and computation, supporting real-time robotic harvesting.
- **FruitQuery:** Designed a unified, lightweight Transformer-based model for ripeness segmentation across peaches and strawberries. With 14.08M parameters, it achieves 67.02 AP and outperforms 13 existing CNN and Transformer models, especially under occlusion and variable lighting. The shared training approach reduces redundancy and supports cross-species generalisation.
- **AppleSSL:** Introduced a self-supervised framework using minimal labels to estimate continuous apple ripeness scores (0.0–1.0) under occlusion. With 86.3M parameters, it achieves 0.75 SSIM and 25.36 PSNR, outperforming 15 self-supervised methods and enabling scalable orchard monitoring and harvest planning.

Together, these contributions provide a flexible and efficient toolkit for real-world fruit analysis, advancing sustainable and automated agriculture through deep learning.

## 7.3 Limitations

While this research presents significant advancements in deep learning for fruit ripeness determination, several limitations remain:

- **Dataset Dependency and Generalisation:** The models in this thesis rely on well-annotated datasets, such as the NinePeach and apple datasets, which limit their ability to generalise seamlessly to new fruit varieties or varying orchard conditions. This necessitates further annotation efforts for every new scenario or fruit type.
- **Computational Complexity:** Some of the proposed models, particularly those involving transformer-based architectures or self-supervised learning frameworks, exhibit considerable computational overhead. This may limit their real-time applicability on edge devices or agricultural robots, where both inference speed and resource efficiency are crucial.

- **Occlusion Handling:** Despite improvements from attention mechanisms and self-supervised techniques, occlusion remains a challenge, especially when parts of the fruit, like the peduncle or calyx, are obstructed by environmental factors such as branches or poor lighting. The models currently struggle to reliably detect and segment occluded regions.
- **Fruit Variety Limitation:** The models are designed primarily for fruits with clear visual cues for ripeness, such as peaches and apples. This restricts their applicability to fruits that rely on internal characteristics (e.g., bananas or avocados) or those that do not undergo significant colour change during ripening (e.g., Granny Smith apples).

## 7.4 Future Work

To address these limitations and further improve fruit ripeness determination in real-world applications, the following directions for future research are suggested:

- **Multi-fruit Generalisation:** Extending the proposed models to additional fruit types and incorporating multi-modal data, such as hyperspectral or thermal imaging, would enhance robustness and improve generalisation. This would make the models more adaptable to diverse fruit species and varying orchard conditions, which are common in practical agriculture.
- **Real-time Deployment with In-field Robots:** Optimising inference speed and reducing the model size will be key to deploying these models on edge devices and robots. Future work could integrate deep learning models with in-field robots for selective harvesting. These robots need to operate under resource constraints and require fast, reliable predictions to support selective harvesting.
- **Active and Semi-supervised Learning:** To reduce annotation requirements, future work could explore active learning or semi-supervised learning methods. These approaches could help reduce the need for extensive labelled datasets, enabling the models to generalise better to new fruit types or orchard conditions without requiring as much manual annotation.
- **Advanced Occlusion Recovery:** Although current attention mechanisms and self-supervised techniques have shown promise, more advanced generative models, such as diffusion models or neural radiance fields (NeRF), could be explored to further enhance occlusion recovery. These methods have the potential to better reconstruct occluded fruit regions and improve segmentation accuracy under challenging field conditions.

- **Integration with Foundation Models:** Foundation models pre-trained on large-scale datasets could be explored as backbones for feature extraction or reconstruction. These models may provide strong general-purpose representations that transfer well to different fruit types or orchard environments, further reducing the need for task-specific data collection and training from scratch.

By addressing these limitations, future research can enhance the scalability, efficiency, and applicability of fruit ripeness estimation models, driving more sustainable practices in precision agriculture.

# References

- [1] S. E. Adebayo, N. Hashim, K. Abdan, M. Hanafi, and K. Mollazade. Prediction of quality attributes and ripeness classification of bananas using optical properties. *Scientia Horticulturae*, 212:171–182, Nov. 2016.
- [2] G. Agati, C. DOnofrio, E. Ducci, A. Cuzzola, D. Remorini, L. Tuccio, F. Lazzini, and G. Mattii. Potential of a Multiparametric Optical Sensor for Determining in Situ the Maturity Components of Red and White *Vitis vinifera* Wine Grapes. *Journal of Agricultural and Food Chemistry*, 61(50):12211–12218, Dec. 2013.
- [3] M. L. Amodio, F. Ceglie, M. M. A. Chaudhry, F. Piazzolla, and G. Colelli. Potential of NIR spectroscopy for predicting internal quality and discriminating among strawberry fruits from different production systems. *Postharvest Biology and Technology*, 125:112–121, Mar. 2017.
- [4] M. Assran, R. Balestrieri, Q. Duval, F. Bordes, I. Misra, P. Bojanowski, P. Vincent, M. Rabbat, and N. Ballas. The Hidden Uniform Cluster Prior in Self-Supervised Learning, Oct. 2022. arXiv:2210.07277.
- [5] M. Assran, M. Caron, I. Misra, P. Bojanowski, F. Bordes, P. Vincent, A. Joulin, M. Rabbat, and N. Ballas. Masked Siamese Networks for Label-Efficient Learning. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXI*, pages 456–473, Berlin, Heidelberg, Oct. 2022. Springer-Verlag.
- [6] Y. Bai, Y. Guo, Q. Zhang, B. Cao, and B. Zhang. Multi-network fusion algorithm with transfer learning for green cucumber segmentation and recognition under complex natural environment. *Computers and Electronics in Agriculture*, 194:106789, Mar. 2022.
- [7] A. Baiano, C. Terracone, G. Peri, and R. Romaniello. Application of hyperspectral imaging for prediction of physico-chemical and sensory characteristics of table grapes. *Computers and Electronics in Agriculture*, 87:142–151, Sept. 2012.
- [8] B. H. A. Bakar, A. J. Ishak, R. Shamsuddin, and W. Z. W. Hassan. Ripeness level classification for pineapple using RGB and HSI colour maps. *Journal of Theoretical and Applied Information Technology*, 57, Nov. 2013.
- [9] D. Bank, N. Koenigstein, and R. Giryes. Autoencoders, 2020. Version Number: 2.



- [10] H. Bao, L. Dong, S. Piao, and F. Wei. BEiT: BERT Pre-Training of Image Transformers, Sept. 2022. arXiv:2106.08254 [cs].
- [11] A. Bardes, J. Ponce, and Y. LeCun. VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning, Jan. 2022. arXiv:2105.04906.
- [12] A. Bardes, J. Ponce, and Y. LeCun. VICRegL: Self-Supervised Learning of Local Visual Features, Oct. 2022. arXiv:2210.01571.
- [13] S. Bargoti and J. P. Underwood. Image Segmentation for Fruit Detection and Yield Estimation in Apple Orchards. *Journal of Field Robotics*, 34(6):1039–1060, Sept. 2017.
- [14] R. Barth, J. Hemming, and E. J. Van Henten. Angle estimation between plant parts for grasp optimisation in harvest robots. *Biosystems Engineering*, 183:26–46, July 2019.
- [15] C. E. Basson, J.-H. Groenewald, J. Kossmann, C. Cronjé, and R. Bauer. Sugar and acid-related quality attributes and enzyme activities in strawberry fruits: Invertase is the main sucrose hydrolysing enzyme. *Food Chemistry*, 121(4):1156–1162, Aug. 2010.
- [16] A. Batu. Determination of acceptable firmness and colour values of tomatoes. *Journal of Food Engineering*, 61(3):471–475, Feb. 2004.
- [17] S. K. Behera, A. K. Rath, and P. K. Sethy. Maturity status classification of papaya fruits based on machine learning and transfer learning approach. *Information Processing in Agriculture*, 8(2):244–250, June 2021.
- [18] A. Benelli, C. Cevoli, A. Fabbri, and L. Ragni. Ripeness evaluation of kiwifruit by hyperspectral imaging. *Biosystems Engineering*, 223:42–52, Nov. 2022.
- [19] D. L. Betemps, J. C. Fachinello, S. P. Galarca, N. M. Portela, D. Remorini, R. Massai, and G. Agati. Non-destructive evaluation of ripening and quality traits in apples using a multiparametric fluorescence sensor. *Journal of the Science of Food and Agriculture*, 92(9):1855–1864, July 2012.
- [20] C. M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer New York, NY, 1 edition, 2006.
- [21] L. Bodria, M. Fiala, R. Guidetti, and R. Oberti. Optical techniques to estimate the ripeness of red-pigmented fruits. *Transactions of the ASAE*, 47(3):815–820, 2004.
- [22] N. Boudhrioua, P. Giampaoli, and C. Bonazzi. Changes in aromatic components of banana during ripening and air-drying. *LWT - Food Science and Technology*, 36(6):633–642, Sept. 2003.
- [23] J. Brezmes, E. Llobet, X. Vilanova, G. Saiz, and X. Correig. Fruit ripeness monitoring using an Electronic Nose. *Sensors and Actuators B: Chemical*, 69(3):223–229, Oct. 2000.

- [24] L. Bu, C. Chen, G. Hu, A. Sugirbay, H. Sun, and J. Chen. Design and evaluation of a robotic apple harvester using optimized picking patterns. *Computers and Electronics in Agriculture*, 198:107092, July 2022.
- [25] C. Camps and D. Christen. Non-destructive assessment of apricot fruit quality by portable visible-near infrared spectroscopy. *LWT - Food Science and Technology*, 42(6):1125–1131, July 2009.
- [26] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-End Object Detection with Transformers. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I*, pages 213–229, Berlin, Heidelberg, Aug. 2020. Springer-Verlag.
- [27] P. Carlini, R. Massantini, and F. Mencarelli. Vis-NIR Measurement of Soluble Solids in Cherry and Apricot by PLS Regression and Wavelength Selection. *Journal of Agricultural and Food Chemistry*, 48(11):5236–5242, Nov. 2000.
- [28] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments, Jan. 2021. arXiv:2006.09882 [cs].
- [29] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging Properties in Self-Supervised Vision Transformers, May 2021. arXiv:2104.14294 [cs].
- [30] A. M. Cavaco, P. Pinto, M. D. Antunes, J. M. D. Silva, and R. Guerra. ‘Rocha’ pear firmness predicted by a Vis/NIR segmented model. *Postharvest Biology and Technology*, 51(3):311–319, Mar. 2009.
- [31] C. Cederberg and U. Sonesson. *Global food losses and food waste: extent, causes and prevention*. Food and Agriculture Organization of the United Nations, Rome, 2011. Meeting Name: International Congress Save Food!
- [32] I. Chandrasekaran, S. S. Panigrahi, L. Ravikanth, and C. B. Singh. Potential of Near-Infrared (NIR) Spectroscopy and Hyperspectral Imaging for Quality and Safety Assessment of Fruits: an Overview. *Food Analytical Methods*, 12(11):2438–2458, Nov. 2019.
- [33] J. Chen, S.-h. Kao, H. He, W. Zhuo, S. Wen, C.-H. Lee, and S.-H. G. Chan. Run, Don’t Walk: Chasing Higher FLOPS for Faster Neural Networks. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12021–12031. IEEE Computer Society, June 2023.
- [34] M. Chen, Z. Chen, L. Luo, Y. Tang, J. Cheng, H. Wei, and J. Wang. Dynamic visual servo control methods for continuous operation of a fruit harvesting robot working throughout an orchard. *Computers and Electronics in Agriculture*, 219:108774, Apr. 2024.

- [35] S. Chen, X. Zou, X. Zhou, Y. Xiang, and M. Wu. Study on fusion clustering and improved YOLOv5 algorithm based on multiple occlusion of *Camellia oleifera* fruit. *Computers and Electronics in Agriculture*, 206:107706, Mar. 2023.
- [36] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A Simple Framework for Contrastive Learning of Visual Representations, June 2020. arXiv:2002.05709 [cs, stat].
- [37] X. Chen, M. Ding, X. Wang, Y. Xin, S. Mo, Y. Wang, S. Han, P. Luo, G. Zeng, and J. Wang. Context Autoencoder for Self-supervised Representation Learning. *International Journal of Computer Vision*, 132(1):208–223, Jan. 2024.
- [38] X. Chen and K. He. Exploring Simple Siamese Representation Learning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15745–15753, Nashville, TN, USA, June 2021. IEEE.
- [39] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar. Masked-Attention Mask Transformer for Universal Image Segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1290–1299, 2022.
- [40] B. Cheng, A. G. Schwing, and A. Kirillov. Per-Pixel Classification is Not All You Need for Semantic Segmentation, Oct. 2021. arXiv:2107.06278 [cs].
- [41] T. Cheng, L. Song, Y. Ge, W. Liu, X. Wang, and Y. Shan. YOLO-World: Real-Time Open-Vocabulary Object Detection. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16901–16911, Seattle, WA, USA, June 2024. IEEE.
- [42] T. Cheng, X. Wang, S. Chen, W. Zhang, Q. Zhang, C. Huang, Z. Zhang, and W. Liu. Sparse Instance Activation for Real-Time Instance Segmentation. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4423–4432, New Orleans, LA, USA, June 2022. IEEE.
- [43] K. S. Chia, H. Abdul Rahim, and R. Abdul Rahim. Prediction of soluble solids content of pineapple via non-invasive low cost visible and shortwave near infrared spectroscopy and artificial neural network. *Biosystems Engineering*, 113(2):158–165, Oct. 2012.
- [44] B.-H. Cho, K. Koyama, E. Olivares Díaz, and S. Koseki. Determination of “Hass” Avocado Ripeness During Storage Based on Smartphone Image and Machine Learning Model. *Food and Bioprocess Technology*, 13(9):1579–1587, Sept. 2020.
- [45] T. Choi, O. Would, A. Salazar-Gomez, X. Liu, and G. Cielniak. Channel randomisation: Self-supervised representation learning for reliable visual anomaly detection in speciality crops. *Computers and Electronics in Agriculture*, 226:109416, Nov. 2024.
- [46] J. Ci, E. J. Van Henten, X. Wang, A. K. Burusa, and G. Kootstra. SSL-NBV: A self-supervised-learning-based next-best-view algorithm for efficient 3D plant reconstruction by a robot. *Computers and Electronics in Agriculture*, 233:110121, June 2025.

- [47] A. Clement, M. Dorais, and M. Vernon. Multivariate Approach to the Measurement of Tomato Maturity and Gustatory Attributes and Their Rapid Assessment by Vis-NIR Spectroscopy. *Journal of Agricultural and Food Chemistry*, 56(5):1538–1544, Mar. 2008.
- [48] A. Clement, M. Dorais, and M. Vernon. Nondestructive Measurement of Fresh Tomato Lycopene Content and Other Physicochemical Characteristics Using Visible-NIR Spectroscopy. *Journal of Agricultural and Food Chemistry*, 56(21):9813–9818, Nov. 2008.
- [49] O. Community. Open Neural Network Exchange (ONNX), 2017.
- [50] N. Corporation. NVIDIA TensorRT: Programmable Inference Accelerator, May 2016.
- [51] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sept. 1995.
- [52] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, Jan. 1967.
- [53] C. H. Crisosto, G. M. Crisosto, and M. A. Ritenour. Testing the reliability of skin color as an indicator of quality for early season ‘Brooks’ (*Prunus avium* L.) cherry. *Postharvest Biology and Technology*, 24(2):147–154, Mar. 2002.
- [54] Y. Cui, C. Jiang, G. Wu, and L. Wang. MixFormer: End-to-End Tracking with Iterative Mixed Attention, Feb. 2023. arXiv:2302.02814 [cs].
- [55] M. Dadwal and V. K. Banga. Estimate Ripeness Level of fruits Using RGB Color Space and Fuzzy Logic Technique. *International Journal of Engineering and Advanced Technology (IJEAT)*, 2(1), Oct. 2012.
- [56] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable Convolutional Networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 764–773, 2017.
- [57] P. Das and J. P. Singh Yadav. Transfer Learning based Tomato Ripeness Classification. In *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pages 423–428, Oct. 2020.
- [58] V. de Sa. Learning Classification with Unlabeled Data. In *Advances in Neural Information Processing Systems*, volume 6. Morgan-Kaufmann, 1993.
- [59] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations (ICLR)*, Oct. 2020.
- [60] N. R. Draper and H. Smith. *Applied Regression Analysis*. Wiley Series in Probability and Statistics. Wiley, 1 edition, Apr. 1998.

- [61] X. Du, H. Cheng, Z. Ma, W. Lu, M. Wang, Z. Meng, C. Jiang, and F. Hong. DSW-YOLO: A detection method for ground-planted strawberry fruits under different occlusion levels. *Computers and Electronics in Agriculture*, 214:108304, Nov. 2023.
- [62] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman. With a Little Help From My Friends: Nearest-Neighbor Contrastive Learning of Visual Representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9588–9597, 2021.
- [63] I. Eduardo, G. Chietera, D. Bassi, L. Rossini, and A. Vecchietti. Identification of key odor volatile compounds in the essential oil of nine peach accessions: Identification of odor volatile compounds in peach accessions. *Journal of the Science of Food and Agriculture*, 90(7):1146–1154, May 2010.
- [64] N. El-Bendary, E. El Hariri, A. E. Hassanien, and A. Badr. Using machine learning techniques for evaluating tomato ripeness. *Expert Systems with Applications*, 42(4):1892–1905, Mar. 2015.
- [65] G. ElMasry, N. Wang, A. ElSayed, and M. Ngadi. Hyperspectral imaging for nondestructive determination of some quality attributes for strawberry. *Journal of Food Engineering*, 81(1):98–107, July 2007.
- [66] W. Falcon. PyTorch Lightning, Mar. 2019.
- [67] FAO. *Fruit and vegetables – your dietary essentials. The International Year of Fruits and Vegetables*. FAO, Rome, Italy, 1 edition, 2020.
- [68] J. Fenoll, A. Manso, P. Hellin, L. Ruiz, and P. Flores. Changes in the aromatic composition of the *Vitis vinifera* grape Muscat Hamburg during ripening. *Food Chemistry*, 114(2):420–428, May 2009.
- [69] G. Ferre, G. Massol, G. Le Fur, and F. Villeneuve. Apple Colour and Ripeness. Use of a Colorimeter: Prospects. *Infos CTIFL, France*, 30:19–24, 1987.
- [70] A. Ferrer, S. Remon, A. I. Negueruela, and R. Oria. Changes during the ripening of the very late season Spanish peach cultivar Calanda. *Scientia Horticulturae*, 105(4):435–446, July 2005.
- [71] L. Fu, Y. Feng, Y. Majeed, X. Zhang, J. Zhang, M. Karkee, and Q. Zhang. Kiwifruit detection in field images using Faster R-CNN with ZFNet. *IFAC-PapersOnLine*, 51(17):45–50, Jan. 2018.
- [72] R.-L. Gai, K. Wei, and P.-F. Wang. SSMDA: Self-Supervised Cherry Maturity Detection Algorithm Based on Multi-Feature Contrastive Learning. *Agriculture*, 13(5):939, May 2023. Number: 5 Publisher: Multidisciplinary Digital Publishing Institute.

- [73] J. Gao, A. P. French, M. P. Pound, Y. He, T. P. Pridmore, and J. G. Pieters. Deep convolutional neural networks for image-based *Convolvulus sepium* detection in sugar beet fields. *Plant Methods*, 16(1):29, Dec. 2020.
- [74] X. Gao, W. Xue, C. Lennox, M. Stevens, and J. Gao. Developing a hybrid convolutional neural network for automatic aphid counting in sugar beet fields. *Computers and Electronics in Agriculture*, 220:108910, May 2024.
- [75] Z. Gao, Y. Shao, G. Xuan, Y. Wang, Y. Liu, and X. Han. Real-time hyperspectral imaging for the in-field estimation of strawberry ripeness with deep learning. *Artificial Intelligence in Agriculture*, 4:31–38, Jan. 2020.
- [76] C. V. Garcia, R. J. Stevenson, R. G. Atkinson, R. A. Winz, and S.-Y. Quek. Changes in the bound aroma profiles of ‘Hayward’ and ‘Hort16A’ kiwifruit (*Actinidia* spp.) during ripening and GC-olfactometry analysis. *Food Chemistry*, 137(1-4):45–54, Apr. 2013.
- [77] A. Gastelum-Barrios, R. A. Borquez-Lopez, E. Rico-Garcia, and G. M. Soto-Zarazua. Tomato quality evaluation with image processing: A review. *African Journal of Agricultural Research*, 6(14):3333–3339, July 2011.
- [78] Y. Ge, P. J. From, and Y. Xiong. Multi-view gripper internal sensing for the regression of strawberry ripeness using a mini-convolutional neural network for robotic harvesting. *Computers and Electronics in Agriculture*, 216:108474, Jan. 2024.
- [79] J. Gené-Mola, M. Ferrer-Ferrer, E. Gregorio, P. M. Blok, J. Hemming, J.-R. Morros, J. R. Rosell-Polo, V. Vilaplana, and J. Ruiz-Hidalgo. Looking behind occlusions: A study on amodal segmentation for robust on-tree apple fruit size estimation. *Computers and Electronics in Agriculture*, 209:107854, June 2023.
- [80] A. Ghanbari, G. H. Shirdel, and F. Maleki. Semi-Self-Supervised Domain Adaptation: Developing Deep Learning Models with Limited Annotated Data for Wheat Head Segmentation. *Algorithms*, 17(6):267, June 2024.
- [81] J. J. Giovannoni. Genetic Regulation of Fruit Development and Ripening. *THE PLANT CELL ONLINE*, 16(suppl\_1):S170–S180, Mar. 2004.
- [82] D. Girod, J. A. Landry, G. Doyon, J. A. Osuna-Garcia, S. Salazar-Garcia, and R. Geonaga. Evaluating Hass Avocado Maturity Using Hyperspectral Imaging. In *Caribbean Food Crops Society 44th Annual Meeting*, page 12, Miami, Florida, USA, 2008. Unknown.
- [83] A. H. Gomez, Y. He, and A. G. Pereira. Non-destructive measurement of acidity, soluble solids and firmness of Satsuma mandarin using Vis/NIR-spectroscopy techniques. *Journal of Food Engineering*, 77(2):313–319, Nov. 2006.
- [84] A. H. Gomez, J. Wang, and A. G. Pereira. Mandarin Ripeness Monitoring and Quality Attribute Evaluation Using an Electronic Nose Technique. *Transactions of the ASABE*, 50(6):2137–2142, 2007.

- [85] J. Gomez-Sanchis, D. Lorente, E. Soria-Olivas, N. Aleixos, S. Cubero, and J. Blasco. Development of a Hyperspectral Computer Vision System Based on Two Liquid Crystal Tuneable Filters for Fruit Inspection. Application to Detect Citrus Fruits Decay. *Food and Bioprocess Technology*, 7(4):1047–1056, Apr. 2014.
- [86] M. Gonzalez-Aguero, S. Troncoso, O. Gudenschwager, R. Campos-Vargas, M. A. Moya-Leon, and B. G. Defilippi. Differential expression levels of aroma-related genes during ripening of apricot (*Prunus armeniaca* L.). *Plant Physiology and Biochemistry*, 47(5):435–440, May 2009.
- [87] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.
- [88] C. V. Greensill and K. B. Walsh. Calibration Transfer between Miniature Photodiode Array-Based Spectrometers in the near Infrared Assessment of Mandarin Soluble Solids Content. *Journal of Near Infrared Spectroscopy*, 10(1):27–35, Jan. 2002. Publisher: SAGE Publishing.
- [89] I. Grijalva, B. J. Spiesman, and B. McCornack. Image classification of sugarcane aphid density using deep convolutional neural networks. *Smart Agricultural Technology*, 3:100089, Feb. 2023.
- [90] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko. Bootstrap your own latent a new approach to self-supervised learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20*, pages 21271–21284, Red Hook, NY, USA, Dec. 2020. Curran Associates Inc.
- [91] Z. Gu, D. He, J. Huang, J. Chen, X. Wu, B. Huang, T. Dong, Q. Yang, and H. Li. Simultaneous detection of fruits and fruiting stems in mango using improved YOLOv8 model deployed by edge device. *Computers and Electronics in Agriculture*, 227:109512, Dec. 2024.
- [92] Z. Gu, X. Ma, H. Guan, Q. Jiang, H. Deng, B. Wen, T. Zhu, and X. Wu. Tomato fruit detection and phenotype calculation method based on the improved RTDETR model. *Computers and Electronics in Agriculture*, 227:109524, Dec. 2024.
- [93] J. Gui, T. Chen, J. Zhang, Q. Cao, Z. Sun, H. Luo, and D. Tao. A Survey on Self-supervised Learning: Algorithms, Applications, and Future Trends, Sept. 2023. arXiv:2301.05712 [cs].
- [94] Y. Gulzar, Y. Hamid, A. B. Soomro, A. A. Alwan, and L. Journaux. A Convolution Neural Network-Based Seed Classification System. *Symmetry*, 12(12):2018, Dec. 2020.



- [95] J. Guo, K. Han, H. Wu, Y. Tang, X. Chen, Y. Wang, and C. Xu. CMT: Convolutional Neural Networks Meet Vision Transformers. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12165–12175, New Orleans, LA, USA, June 2022. IEEE.
- [96] D. Gustafson and W. Kessel. Fuzzy clustering with a fuzzy covariance matrix. In *1978 IEEE Conference on Decision and Control including the 17th Symposium on Adaptive Processes*, pages 761–766, San Diego, CA, USA, Jan. 1978. IEEE.
- [97] J. Guthrie and K. Walsh. Non-invasive assessment of pineapple and mango fruit quality using near infra-red spectroscopy. *Australian Journal of Experimental Agriculture*, 37(2):253, 1997.
- [98] J. A. Guthrie, D. J. Reid, and K. B. Walsh. Assessment of internal quality attributes of mandarin fruit. 2. NIR calibration model robustness. *Australian Journal of Agricultural Research*, 56(4):417, 2005.
- [99] S. Gutierrez, A. Wendel, and J. Underwood. Spectral filter design based on in-field hyperspectral imaging and machine learning for mango ripeness estimation. *Computers and Electronics in Agriculture*, 164:104890, Sept. 2019.
- [100] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York, New York, NY, 2009.
- [101] S. S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999. Google-Books-ID: bX4pAQAAMAAJ.
- [102] S. S. Haykin. *Neural networks and learning machines*. Prentice-Hall, New York Munich, 3. ed edition, 2009.
- [103] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked Autoencoders Are Scalable Vision Learners, Dec. 2021. arXiv:2111.06377 [cs].
- [104] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.
- [105] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, Oct. 2017. ISSN: 2380-7504.
- [106] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016. ISSN: 1063-6919.
- [107] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural Collaborative Filtering, 2017. Version Number: 2.

- [108] Z. He, S. R. Khanal, X. Zhang, M. Karkee, and Q. Zhang. Real-time Strawberry Detection Based on Improved YOLOv5s Architecture for Robotic Harvesting in open-field environment, Oct. 2023. arXiv:2308.03998 [cs].
- [109] A. Herrero-Langreo, L. Lunadei, L. Lleó, B. Diezma, and M. Ruiz-Altisent. Multispectral Vision for Monitoring Peach Ripeness. *Journal of Food Science*, 76(2):E178–E187, 2011. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1750-3841.2010.02000.x>.
- [110] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine*, 29(6):82–97, Nov. 2012.
- [111] A. M. Hoffmann, G. Noga, and M. Hunsche. Fluorescence indices for monitoring the ripening of tomatoes in pre- and postharvest phases. *Scientia Horticulturae*, 191:74–81, Aug. 2015.
- [112] T. Hong, J. Qiao, N. Wang, M. Ngadi, Z. Zhao, and Z. Li. Non-destructive inspection of Chinese pear quality based on hyperspectral imaging technique. *Transactions of the Chinese Society of Agricultural Engineering*, 23(2):151–155, 2007.
- [113] A. Hore and D. Ziou. Image Quality Metrics: PSNR vs. SSIM. In *2010 20th International Conference on Pattern Recognition*, pages 2366–2369, Istanbul, Turkey, Aug. 2010. IEEE.
- [114] C. Hu, X. Liu, Z. Pan, and P. Li. Automatic Detection of Single Ripe Tomato on Plant Combining Faster R-CNN and Intuitionistic Fuzzy Set. *IEEE Access*, 7:154683–154696, 2019.
- [115] T. Huang, L. Huang, S. You, F. Wang, C. Qian, and C. Xu. LightViT: Towards Light-Weight Convolution-Free Vision Transformers, July 2022. arXiv:2207.05557 [cs].
- [116] D. P. Hughes and M. Salathe. An open access repository of images on plant health to enable the development of mobile disease diagnostics, Apr. 2016. arXiv:1511.08060 [cs].
- [117] G. Hui, Y. Wu, D. Ye, and W. Ding. Fuji Apple Storage Time Predictive Method Using Electronic Nose. *Food Analytical Methods*, 6(1):82–88, Feb. 2013.
- [118] G. Ian, B. Yoshua, and C. Aaron. *Deep Learning*. MIT Press, 2016.
- [119] R. Infante, L. Contador, P. Rubio, K. Mesa, and C. Meneses. Non-destructive monitoring of flesh softening in the black-skinned Japanese plums ‘Angeleno’ and ‘Autumn beaut’ on-tree and postharvest. *Postharvest Biology and Technology*, 61(1):35–40, July 2011.
- [120] S. Ioffe and C. Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37, ICML’15*, pages 448–456, Lille, France, July 2015. JMLR.org.

- [121] P. Jaiswal, S. N. Jha, and R. Bharadwaj. Non-destructive prediction of quality of intact banana using spectroscopy. *Scientia Horticulturae*, 135:14–22, Feb. 2012.
- [122] R. Jangali, B. McGuinness, H. Lim, H. Williams, A. H. Qureshi, D. Smith, B. A. MacDonald, and M. Duke. Development of a Novel Multipurpose Robotic End Effector for Fruitlet Thinning and Fruit Harvesting of Apples. In *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*, pages 2073–2078, Aug. 2024. ISSN: 2161-8089.
- [123] L. Janik, D. Cozzolino, R. Damberg, W. Cynkar, and M. Gishen. The prediction of total anthocyanin concentration in red-grape homogenates using visible-near-infrared spectroscopy and artificial neural networks. *Analytica Chimica Acta*, 594(1):107–118, June 2007.
- [124] P. Jannok, Y. Kamitani, and S. Kawano. Development of a Common Calibration Model for Determining the Brix Value of Intact Apple, Pear and Persimmon Fruits by near Infrared Spectroscopy. *Journal of Near Infrared Spectroscopy*, 22(5):367–373, Oct. 2014. Publisher: SAGE Publishing.
- [125] S. Jha, S. Chopra, and A. Kingsly. Modeling of color values for nondestructive evaluation of maturity of mango. *Journal of Food Engineering*, 78(1):22–26, Jan. 2007.
- [126] S. Jha, A. Kingsly, and S. Chopra. Non-destructive Determination of Firmness and Yellowness of Mango during Growth and Storage using Visual Spectroscopy. *Biosystems Engineering*, 94(3):397–402, July 2006.
- [127] W. Jia, Y. Tian, R. Luo, Z. Zhang, J. Lian, and Y. Zheng. Detection and segmentation of overlapped fruits based on optimized mask R-CNN application in apple harvesting robot. *Computers and Electronics in Agriculture*, 172:105380, May 2020.
- [128] W. Jia, Z. Zhang, W. Shao, S. Hou, Z. Ji, G. Liu, and X. Yin. FoveaMask: A fast and accurate deep learning model for green fruit instance segmentation. *Computers and Electronics in Agriculture*, 191:106488, Dec. 2021.
- [129] W. Jia, Z. Zhang, W. Shao, Z. Ji, and S. Hou. RS-Net: robust segmentation of green overlapped apples. *Precision Agriculture*, 23(2):492–513, Apr. 2022.
- [130] Z. Jiao, K. Huang, G. Jia, H. Lei, Y. Cai, and Z. Zhong. An effective litchi detection method based on edge devices in a complex scene. *Biosystems Engineering*, 222:15–28, Oct. 2022.
- [131] Jocher Glenn, Jing Qiu, and Ayush Chaurasia. Ultralytics YOLO, Jan. 2023.
- [132] A. Kader. *Postharvest Technology of Horticulture Crops*. Postharvest Technology Center, One Shields Avenue, Davis, CA, 3 edition, 2002.

- [133] H. Kang and C. Chen. Fruit detection, segmentation and 3D visualisation of environments in apple orchards. *Computers and Electronics in Agriculture*, 171:105302, Apr. 2020.
- [134] H. Kang, H. Zhou, X. Wang, and C. Chen. Real-Time Fruit Recognition and Grasping Estimation for Robotic Apple Harvesting. *Sensors*, 20(19):5670, Jan. 2020. Number: 19 Publisher: Multidisciplinary Digital Publishing Institute.
- [135] R. Kestur, A. Meduri, and O. Narasipura. MangoNet: A deep semantic segmentation architecture for a method to detect and count mangoes in an open orchard. *Engineering Applications of Artificial Intelligence*, 77:59–69, Jan. 2019.
- [136] K. Kiilu, G. Okeyo, R. Rimiru, and K. Ogada. Using Naïve Bayes Algorithm in detection of Hate Tweets. *International Journal of Scientific and Research Publications (IJSRP)*, 8, Mar. 2018.
- [137] S. Kim, S.-J. Hong, J. Ryu, E. Kim, C.-H. Lee, and G. Kim. Application of amodal segmentation on cucumber segmentation and occlusion recovery. *Computers and Electronics in Agriculture*, 210:107847, July 2023.
- [138] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization, 2014. Version Number: 9.
- [139] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick. Segment Anything. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3992–4003, Oct. 2023. ISSN: 2380-7504.
- [140] A. Kirillov, Y. Wu, K. He, and R. Girshick. PointRend: Image Segmentation as Rendering, Feb. 2020. arXiv:1912.08193 [cs].
- [141] K. Ko, I. Jang, J. H. Choi, J. H. Lim, and D. U. Lee. Stochastic Decision Fusion of Convolutional Neural Networks for Tomato Ripeness Detection in Agricultural Sorting Systems. *Sensors*, 21(3):917, Jan. 2021. Number: 3 Publisher: Multidisciplinary Digital Publishing Institute.
- [142] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.
- [143] F. Kurtulmus. Identification of sunflower seeds with deep convolutional neural networks. *Journal of Food Measurement and Characterization*, 15(2):1024–1033, Apr. 2021.
- [144] K. Lammers, K. Zhang, K. Zhu, P. Chu, Z. Li, and R. Lu. Development and evaluation of a dual-arm robotic apple harvesting system. *Computers and Electronics in Agriculture*, 227:109586, Dec. 2024.

- [145] M. Lechaudel, G. Damour, L. Urban, and J. Joas. Chlorophyll fluorescence as an indicator of when to harvest mango 'Cogshall' fruit according to the market (export or domestic). *Acta Horticulturae*, 992:159–165, May 2013.
- [146] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov. 1998.
- [147] J. S. Lee, S.-C. Kim, K. C. Seong, C.-H. Kim, Y. C. Um, and S.-K. Lee. Quality Prediction of Kiwifruit Based on Near Infrared Spectroscopy. *Korean Journal of Horticultural Science and Technology*, 30(6):709–717, Dec. 2012.
- [148] C. Lehnert, A. English, C. McCool, A. W. Tow, and T. Perez. Autonomous Sweet Pepper Harvesting for Protected Cropping Systems. *IEEE Robotics and Automation Letters*, 2(2):872–879, Apr. 2017.
- [149] S. Lenk, C. Buschmann, and E. E. Pfundel. In vivo assessing flavonols in white grape berries (*Vitis vinifera* L. cv. Pinot Blanc) of different degrees of ripeness using chlorophyll fluorescence imaging. *Functional Plant Biology*, 34(12):1092, 2007.
- [150] B. Li, M. Cobo-Medina, J. Lecourt, N. Harrison, R. J. Harrison, and J. V. Cross. Application of hyperspectral imaging for nondestructive measurement of plum quality attributes. *Postharvest Biology and Technology*, 141:8–15, July 2018.
- [151] G. Li, L. Jiao, P. Chen, K. Liu, R. Wang, S. Dong, and C. Kang. Spatial convolutional self-attention-based transformer module for strawberry disease identification under complex background. *Computers and Electronics in Agriculture*, 212:108121, Sept. 2023.
- [152] H. Li, J. Chen, Z. Gu, T. Dong, J. Chen, J. Huang, J. Gai, H. Gong, Z. Lu, and D. He. Optimizing edge-enabled system for detecting green passion fruits in complex natural orchards using lightweight deep learning model. *Computers and Electronics in Agriculture*, 234:110269, July 2025.
- [153] H. Li, Z. Gu, D. He, X. Wang, J. Huang, Y. Mo, P. Li, Z. Huang, and F. Wu. A lightweight improved YOLOv5s model and its deployment for detecting pitaya fruits in daytime and nighttime light-supplement environments. *Computers and Electronics in Agriculture*, 220:108914, May 2024.
- [154] H. Li, W. S. Lee, and K. Wang. Identifying blueberry fruit of different growth stages using natural outdoor color images. *Computers and Electronics in Agriculture*, 106:91–101, Aug. 2014.
- [155] J. Li, X. Xia, W. Li, H. Li, X. Wang, X. Xiao, R. Wang, M. Zheng, and X. Pan. Next-ViT: Next Generation Vision Transformer for Efficient Deployment in Realistic Industrial Scenarios, Aug. 2022. arXiv:2207.05501 [cs].

- [156] K. Li and J. Malik. Amodal Instance Segmentation. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, pages 677–693, Cham, 2016. Springer International Publishing.
- [157] L. Li, W. Hu, J. Lu, and C. Zhang. Leaf vein segmentation with self-supervision. *Computers and Electronics in Agriculture*, 203:107352, Dec. 2022.
- [158] Q. Li, W. Jia, M. Sun, S. Hou, and Y. Zheng. A novel green apple segmentation algorithm based on ensemble U-Net under complex orchard environment. *Computers and Electronics in Agriculture*, 180:105900, Jan. 2021.
- [159] Y. Li, Q. Feng, C. Liu, Z. Xiong, Y. Sun, F. Xie, T. Li, and C. Zhao. MTA-YOLACT: Multitask-aware network on fruit bunch identification for cherry tomato robotic harvesting. *European Journal of Agronomy*, 146:126812, May 2023.
- [160] C. Liang, J. Xiong, Z. Zheng, Z. Zhong, Z. Li, S. Chen, and Z. Yang. A visual detection method for nighttime litchi fruits and fruiting stems. *Computers and Electronics in Agriculture*, 169:105192, Feb. 2020.
- [161] J. Liang, K. Huang, H. Lei, Z. Zhong, Y. Cai, and Z. Jiao. Occlusion-aware fruit segmentation in complex natural environments under shape prior. *Computers and Electronics in Agriculture*, 217:108620, Feb. 2024.
- [162] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature Pyramid Networks for Object Detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, July 2017. ISSN: 1063-6919.
- [163] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal Loss for Dense Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327, July 2018. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.
- [164] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common Objects in Context. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, Lecture Notes in Computer Science, pages 740–755, Cham, 2014. Springer International Publishing.
- [165] C. Liu, W. Liu, W. Chen, J. Yang, and L. Zheng. Feasibility in multispectral imaging for predicting the content of bioactive compounds in intact tomato fruit. *Food Chemistry*, 173:482–488, Apr. 2015.
- [166] C. Liu, W. Liu, X. Lu, F. Ma, W. Chen, J. Yang, and L. Zheng. Application of Multispectral Imaging to Determine Quality Attributes and Ripeness Stage in Strawberry Fruit. *PLoS ONE*, 9(2):e87818, Feb. 2014.

- [167] H. Liu, Y. Zhan, H. Xia, Q. Mao, and Y. Tan. Self-supervised transformer-based pre-training method using latent semantic masking auto-encoder for pest and disease classification. *Computers and Electronics in Agriculture*, 203:107448, Dec. 2022.
- [168] M. Liu, P. Fu, and C. Renfa. Non-destructive estimation peach SSC and firmness by mutispectral reflectance imaging. *New Zealand Journal of Agricultural Research*, 50(5):601–608, Dec. 2007.
- [169] R. Liu, J. Lehman, P. Molino, F. P. Such, E. Frank, A. Sergeev, and J. Yosinski. An intriguing failing of convolutional neural networks and the CoordConv solution. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS’18, pages 9628–9639, Red Hook, NY, USA, Dec. 2018. Curran Associates Inc.
- [170] X. Liu, D. Zhao, W. Jia, W. Ji, C. Ruan, and Y. Sun. Cucumber Fruits Detection in Greenhouses Based on Instance Segmentation. *IEEE Access*, 7:139635–139642, 2019.
- [171] Y. Liu, S. Zhou, H. Wu, W. Han, C. Li, and H. Chen. Joint optimization of autoencoder and Self-Supervised Classifier: Anomaly detection of strawberries using hyperspectral imaging. *Computers and Electronics in Agriculture*, 198:107007, July 2022.
- [172] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9992–10002. IEEE Computer Society, Oct. 2021.
- [173] D. Ljubobratovic, M. Vukovic, M. Brkic Bakaric, T. Jemric, and M. Matetic. Assessment of Various Machine Learning Models for Peach Maturity Prediction Using Non-Destructive Sensor Data. *Sensors*, 22(15):5791, Aug. 2022.
- [174] L. Lleo, P. Barreiro, M. Ruiz-Altisent, and A. Herrero. Multispectral images of peach related to firmness and maturity at harvest. *Journal of Food Engineering*, 93(2):229–235, July 2009.
- [175] P. Lobit, P. Soing, M. Genard, and R. Habib. Theoretical Analysis of Relationships Between Composition, pH, and Titratable Acidity of Peach Fruit. *Journal of Plant Nutrition*, 25(12):2775–2792, Nov. 2002.
- [176] R. L. Long and K. B. Walsh. Limitations to the measurement of intact melon total soluble solids using near infrared spectroscopy. *Australian Journal of Agricultural Research*, 57(4):403, 2006.
- [177] I. Loshchilov and F. Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts, 2016. Version Number: 5.
- [178] I. Loshchilov and F. Hutter. Decoupled Weight Decay Regularization, Jan. 2019. arXiv:1711.05101 [cs, math].



- [179] J. Lu, P. Chen, C. Yu, Y. Lan, L. Yu, R. Yang, H. Niu, H. Chang, J. Yuan, and L. Wang. Lightweight green citrus fruit detection method for practical environmental applications. *Computers and Electronics in Agriculture*, 215:108205, Dec. 2023.
- [180] R. Lu. Predicting firmness and sugar content of sweet cherries using near-infrared diffuse reflectance spectroscopy. *Transactions of the ASAE*, 44(5), 2001.
- [181] R. Lu and Y. Peng. Hyperspectral Scattering for assessing Peach Fruit Firmness. *Biosystems Engineering*, 93(2):161–171, Feb. 2006.
- [182] L. Luchsinger and C. Walsh. Development of an objective and non-destructive harvest maturity index for peaches and nectarines. *Acta Horticulturae*, 465:679–688, Apr. 1998.
- [183] W. Luo, J. Zhang, S. Liu, H. Huang, B. Zhan, G. Fan, and H. Zhang. Prediction of soluble solid content in Nanfeng mandarin by combining hyperspectral imaging and effective wavelength selection. *Journal of Food Composition and Analysis*, 126:105939, Feb. 2024.
- [184] T. Ma, T. Inagaki, and S. Tsuchikawa. Development of a sensitivity-enhanced chlorophyll fluorescence lifetime spectroscopic method for nondestructive monitoring of fruit ripening and postharvest decay. *Postharvest Biology and Technology*, 198:112231, Apr. 2023.
- [185] Z. Ma, N. Dong, J. Gu, H. Cheng, Z. Meng, and X. Du. STRAW-YOLO: A detection method for strawberry fruits targets and key points. *Computers and Electronics in Agriculture*, 230:109853, Mar. 2025.
- [186] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, volume 5.1, pages 281–298. University of California Press, Jan. 1967.
- [187] J. Mattheis, D. Felicetti, and D. R. Rudell. Pithy brown core in ‘d’Anjou’ pear (*Pyrus communis* L.) fruit developing during controlled atmosphere storage at pO<sub>2</sub> determined by monitoring chlorophyll fluorescence. *Postharvest Biology and Technology*, 86:259–264, Dec. 2013.
- [188] V. McGlone and S. Kawano. Firmness, dry-matter and soluble-solids assessment of postharvest kiwifruit by NIR spectroscopy. *Postharvest Biology and Technology*, 13(2):131–141, Apr. 1998.
- [189] V. A. McGlone, D. G. Fraser, R. B. Jordan, and R. Kunnemeyer. Internal Quality Assessment of Mandarin Fruit by vis/NIR Spectroscopy. *Journal of Near Infrared Spectroscopy*, 11(5):323–332, Oct. 2003. Publisher: SAGE Publishing.
- [190] B. J. McGuinness, M. D. Duke, K. C. Au, and H. S. Lim. Field factory for the automated harvesting of forestry tree stock. *Biosystems Engineering*, 227:52–67, Mar. 2023.
- [191] S. Mehta and M. Rastegari. MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer, Mar. 2022. arXiv:2110.02178 [cs].

- [192] F. Mendoza and J. Aguilera. Application of Image Analysis for Classification of Ripening Bananas. *Journal of Food Science*, 69(9), Dec. 2004.
- [193] F. Mendoza, P. Dejmek, and J. M. Aguilera. Calibrated color measurements of agricultural foods using image analysis. *Postharvest Biology and Technology*, 41(3):285–295, Sept. 2006.
- [194] F. Mendoza, R. Lu, D. Ariana, H. Cen, and B. Bailey. Integrated spectral and image analysis of hyperspectral scattering data for prediction of apple fruit firmness and soluble solids content. *Postharvest Biology and Technology*, page S0925521411001268, July 2011.
- [195] F. Meng, J. Li, Y. Zhang, S. Qi, and Y. Tang. Transforming unmanned pineapple picking with spatio-temporal convolutional neural networks. *Computers and Electronics in Agriculture*, 214:108298, Nov. 2023.
- [196] M. N. Merzlyak, A. E. Solovchenko, and A. A. Gitelson. Reflectance spectral features and non-destructive estimation of chlorophyll, carotenoid and anthocyanin content in apple fruit. *Postharvest Biology and Technology*, 27(2):197–211, Feb. 2003.
- [197] S.-H. Miraei Ashtiani, S. Javanmardi, M. Jahanbanifard, A. Martynenko, and F. J. Verbeek. Detection of Mulberry Ripeness Stages Using Deep Learning Models. *IEEE Access*, 9:100380–100394, 2021. Conference Name: IEEE Access.
- [198] I. A. Mohtar, N. S. S. Ramli, and Z. Ahmad. Automatic Classification of Mangosteen Ripening Stages using Deep Learning. In *2019 1st International Conference on Artificial Intelligence and Data Sciences (AiDAS)*, pages 44–47, Ipoh, Perak, Malaysia, Sept. 2019. IEEE.
- [199] L. Mu, G. Cui, Y. Liu, Y. Cui, L. Fu, and Y. Gejima. Design and simulation of an integrated end-effector for picking kiwifruit by robot. *Information Processing in Agriculture*, 7(1):58–71, Mar. 2020.
- [200] Muharfiza, D. Al Riza, Y. Saito, K. Itakura, Y. Kohno, T. Suzuki, M. Kuramoto, and N. Kondo. Monitoring of Fluorescence Characteristics of Satsuma Mandarin (Citrus unshiu Marc.) during the Maturation Period. *Horticulturae*, 3(4):51, Oct. 2017.
- [201] S. Munera, J. M. Amigo, J. Blasco, S. Cubero, P. Talens, and N. Aleixos. Ripeness monitoring of two cultivars of nectarine using VIS-NIR hyperspectral reflectance imaging. *Journal of Food Engineering*, 214:29–39, Dec. 2017.
- [202] X. Ni, C. Li, H. Jiang, and F. Takeda. Deep learning image segmentation and extraction of blueberry fruit traits associated with harvestability and yield. *Horticulture Research*, 7:110, Jan. 2020.
- [203] B. M. Nicolai, E. Lotze, A. Peirs, N. Scheerlinck, and K. I. Theron. Non-destructive measurement of bitter pit in apple fruit using NIR hyperspectral imaging. *Postharvest Biology and Technology*, 40(1):1–6, Apr. 2006.

- [204] S. Nie, D. F. Al Riza, Y. Ogawa, T. Suzuki, M. Kuramoto, N. Miyata, and N. Kondo. Potential of a double lighting imaging system for characterization of 'Hayward' kiwifruit harvest indices. *Postharvest Biology and Technology*, 162:111113, Apr. 2020.
- [205] Z. Ning, L. Luo, X. Ding, Z. Dong, B. Yang, J. Cai, W. Chen, and Q. Lu. Recognition of sweet peppers and planning the robotic picking sequence in high-density orchards. *Computers and Electronics in Agriculture*, 196:106878, May 2022.
- [206] B. Niu, Q. Feng, B. Chen, C. Ou, Y. Liu, and J. Yang. HSI-TransUNet: A transformer based semantic segmentation model for crop mapping from UAV hyperspectral imagery. *Computers and Electronics in Agriculture*, 201:107297, Oct. 2022.
- [207] O. O. Olarewaju, I. Bertling, and L. S. Magwaza. Non-destructive evaluation of avocado fruit maturity using near infrared spectroscopy and PLS regression models. *Scientia Horticulturae*, 199:229–236, Feb. 2016.
- [208] U. L. Opara and P. B. Pathare. Bruise damage measurement and analysis of fresh horticultural produce—A review. *Postharvest Biology and Technology*, 91:9–24, May 2014.
- [209] B. Pang, Y. Zhang, Y. Li, J. Cai, and C. Lu. Unsupervised Visual Representation Learning by Synchronous Momentum Grouping. In *Computer Vision – ECCV 2022*, pages 265–282. Springer, Cham, 2022. ISSN: 1611-3349.
- [210] S. Parsa, B. Debnath, M. A. Khan, and A. G. E. Modular autonomous strawberry picking robotic system. *Journal of Field Robotics*, 41(7):2226–2246, Oct. 2024.
- [211] P. Paz, M.-T. Sanchez, D. Perez-Marín, J.-E. Guerrero, and A. Garrido-Varo. Nondestructive Determination of Total Soluble Solid Content and Firmness in Plums Using Near-Infrared Reflectance Spectroscopy. *Journal of Agricultural and Food Chemistry*, 56(8):2565–2570, Apr. 2008.
- [212] M. Peebles, S. H. Lim, M. Duke, and B. McGuinness. Investigation of Optimal Network Architecture for Asparagus Spear Detection in Robotic Harvesting. *IFAC-PapersOnLine*, 52(30):283–287, Jan. 2019.
- [213] A. Peirs, N. Scheerlinck, and B. M. Nicolai. Temperature compensation for near infrared reflectance measurement of apple fruit soluble solids contents. *Postharvest Biology and Technology*, 30(3):233–248, Dec. 2003.
- [214] Y. Peng and R. Lu. Prediction of apple fruit firmness and soluble solids content using characteristics of multispectral scattering images. *Journal of Food Engineering*, 82(2):142–152, Sept. 2007.
- [215] Y. Peng and R. Lu. Analysis of spatially resolved hyperspectral scattering images for assessing apple fruit firmness and soluble solids content. *Postharvest Biology and Technology*, 48(1):52–62, Apr. 2008.

- [216] I. Perez-Borrero, D. Marin-Santos, M. J. Vasallo-Vazquez, and M. E. Gegúndez-Arias. A new deep-learning strawberry instance segmentation methodology based on a fully convolutional neural network. *Neural Computing and Applications*, 33(22):15059–15071, Nov. 2021.
- [217] I. Perez-Borrero, D. Marín-Santos, M. E. Gegúndez-Arias, and E. Cortés-Ancos. A fast and accurate deep learning method for strawberry instance segmentation. *Computers and Electronics in Agriculture*, 178:105736, Nov. 2020.
- [218] D. Perez-Marin, M.-T. Sanchez, P. Paz, M.-A. Soriano, J.-E. Guerrero, and A. Garrido-Varo. Non-destructive determination of quality parameters in nectarines during on-tree ripening and postharvest storage. *Postharvest Biology and Technology*, 52(2):180–188, May 2009.
- [219] B. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, Jan. 1964.
- [220] V. Portnoy, Y. Benyamini, E. Bar, R. Harel-Beja, S. Gepstein, J. J. Giovannoni, A. A. Schaffer, J. Burger, Y. Tadmor, E. Lewinsohn, and N. Katzir. The molecular and biochemical basis for varietal variation in sesquiterpene content in melon (*Cucumis melo* L.) rinds. *Plant Molecular Biology*, 66(6):647–661, Apr. 2008.
- [221] D. Pototzky, A. Sultan, and L. Schmidt-Thieme. FastSiam: Resource-Efficient Self-supervised Learning on a Single GPU. In *Pattern Recognition: 44th DAGM German Conference, DAGM GCPR 2022, Konstanz, Germany, September 27–30, 2022, Proceedings*, pages 53–67, Berlin, Heidelberg, Sept. 2022. Springer-Verlag.
- [222] D. Prasetio and D. Harlili. Predicting football match results with logistic regression. In *2016 International Conference On Advanced Informatics: Concepts, Theory And Application (ICAICTA)*, pages 1–5, 2016.
- [223] S. Qiu, J. Wang, and L. Gao. Discrimination and Characterization of Strawberry Juice Based on Electronic Nose and Tongue: Comparison of Different Juice Processing Approaches by LDA, PLSR, RF, and SVM. *Journal of Agricultural and Food Chemistry*, 62(27):6426–6434, July 2014.
- [224] P. Rajkumar, N. Wang, G. Elmasry, G. Raghavan, and Y. Garipey. Studies on banana fruit quality and maturity stages using hyperspectral imaging. *Journal of Food Engineering*, 108(1):194–200, Jan. 2012.
- [225] R. P. Ramos, J. S. Gomes, R. M. Prates, E. F. Simas Filho, B. J. Teruel, and D. dos Santos Costa. Non-invasive setup for grape maturation classification using deep learning. *Journal of the Science of Food and Agriculture*, 101(5):2042–2051, 2021. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/jsfa.10824>.

- [226] R. Ren, S. Zhang, H. Sun, N. Wang, S. Yang, H. Zhao, and M. Xin. YOLO-RCS: A method for detecting phenological period of 'Yuluxiang' pear in unstructured environment. *Computers and Electronics in Agriculture*, 229:109819, Feb. 2025.
- [227] H. Robbins and S. Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400–407, Sept. 1951.
- [228] M. Rosbi, Z. Omar, U. Khairuddin, A. P. Majeed, and S. A. Bakar. Machine learning for automated oil palm fruit grading: The role of fuzzy C-means segmentation and textural features. *Smart Agricultural Technology*, 9:100691, Dec. 2024.
- [229] P. Roy, A. Kislay, P. A. Plonski, J. Luby, and V. Isler. Vision-based preharvest yield mapping for apple orchards. *Computers and Electronics in Agriculture*, 164:104897, Sept. 2019.
- [230] S. Ruder. An overview of gradient descent optimization algorithms, 2016. Version Number: 2.
- [231] I. Sa, Z. Ge, F. Dayoub, B. Upcroft, T. Perez, and C. McCool. DeepFruits: A Fruit Detection System Using Deep Neural Networks. *Sensors*, 16(8):1222, Aug. 2016.
- [232] A. M. Saad, A. Ibrahim, and N. El-Biale. Internal quality assessment of tomato fruits using image color analysis. *Agricultural Engineering International: CIGR Journal*, 18(1):339–352, Mar. 2016. Number: 1.
- [233] T. T. Santos, L. L. de Souza, A. A. dos Santos, and S. Avila. Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association. *Computers and Electronics in Agriculture*, 170:105247, Mar. 2020.
- [234] I. Saranwong, J. Sornsriwichai, and S. Kawano. On-Tree Evaluation of Harvesting Quality of Mango Fruit Using a Hand-Held NIR Instrument. *Journal of Near Infrared Spectroscopy*, 11(4):283–293, Aug. 2003. Publisher: SAGE Publishing.
- [235] N. Saranya, K. Srinivasan, and S. K. P. Kumar. Banana ripeness stage identification: a deep learning approach. *Journal of Ambient Intelligence and Humanized Computing*, May 2021.
- [236] T.-P. Schlie, T. Rath, D. Köpcke, and W. Dierend. The Impact of Fruit Ripeness on the Lower Oxygen Limit, Chlorophyll Fluorescence and Fermentation Behavior in Apples. *Erwerbs-Obstbau*, 65(5):1227–1237, Oct. 2023.
- [237] R. M. Schmidt. Recurrent Neural Networks (RNNs): A gentle Introduction and Overview, 2019. Version Number: 1.
- [238] Z. Schmiloitch, A. Mizrach, A. Hoffman, H. Egozi, and Y. Fuchs. Determination of mango physiological indices by near-infrared spectrometry. *Postharvest Biology and Technology*, 19(3):245–252, July 2000.

- [239] R. E. Schouten, T. P. Huijben, L. Tijskens, and O. Van Kooten. Modelling quality attributes of truss tomatoes: Linking colour and firmness maturity. *Postharvest Biology and Technology*, 45(3):298–306, Sept. 2007.
- [240] K. P. Seng, L.-M. Ang, L. M. Schmidtke, and S. Y. Rogiers. Computer Vision and Machine Learning for Viticulture Technology. *IEEE Access*, 6:67494–67510, 2018.
- [241] G. B. Seymour, L. Østergaard, N. H. Chapman, S. Knapp, and C. Martin. Fruit Development and Ripening. *Annual Review of Plant Biology*, 64(1):219–241, Apr. 2013.
- [242] D. Shah, V. Trivedi, V. Sheth, A. Shah, and U. Chauhan. ResTS: Residual Deep interpretable architecture for plant disease detection. *Information Processing in Agriculture*, 9(2):212–223, June 2022.
- [243] Y. Shao, Y. Bao, and Y. He. Visible/Near-Infrared Spectra for Linear and Nonlinear Calibrations: A Case to Predict Soluble Solids Contents and pH Value in Peach. *Food and Bioprocess Technology*, 4(8):1376–1383, Nov. 2011.
- [244] S. Sharma, C. Baran, A. Tripathi, A. Awasthi, A. Tiwari, S. Sharma, A. Jaiswal, R. Utam, P. Tandon, R. Singh, and K. N. Uttam. Non-Destructive Monitoring of the Ripening of Plums Using Confocal Micro-Raman and Laser Induced Fluorescence Spectroscopy. *Analytical Letters*, 57(4):531–548, Mar. 2024.
- [245] Q. Shen, X. Zhang, M. Shen, and D. Xu. Multi-scale adaptive YOLO for instance segmentation of grape pedicels. *Computers and Electronics in Agriculture*, 229:109712, Feb. 2025.
- [246] X. Sheng, C. Kang, J. Zheng, and C. Lyu. An edge-guided method to fruit segmentation in complex environments. *Computers and Electronics in Agriculture*, 208:107788, May 2023.
- [247] A. Silwal, J. R. Davidson, M. Karkee, C. Mo, Q. Zhang, and K. Lewis. Design, integration, and field evaluation of a robotic apple harvester. *Journal of Field Robotics*, 34(6):1140–1159, 2017. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21715>.
- [248] D. C. Slaughter and C. H. Crisosto. Nondestructive internal quality assessment of kiwifruit using near-infrared spectroscopy. *Seminars in Food Analysis*, pages 131–140, 1998.
- [249] J. L. Slavin and B. Lloyd. Health Benefits of Fruits and Vegetables. *Advances in Nutrition*, 3(4):506–516, July 2012.
- [250] A. Solovchenko, O. Chivkunova, A. Gitelson, and M. Merzlyak. Non-Destructive Estimation Pigment Content Ripening Quality and Damage in Apple Fruit with Spectral Reflectance in the Visible Range. *School of Natural Resources: Faculty Publications*, Jan. 2010.

- [251] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, Jan. 2014.
- [252] S.S. Sivakumar, J. Qiao, N. Wang, Y. Gariépy, G.S.V. Raghavan, and James McGill. Detecting Maturity Parameters of Mango Using Hyperspectral Imaging Technique. In *2006 Portland, Oregon, July 9-12, 2006*. American Society of Agricultural and Biological Engineers, 2006.
- [253] J. Steinbrener, K. Posch, and R. Leitner. Hyperspectral fruit and vegetable classification using convolutional neural networks. *Computers and Electronics in Agriculture*, 162:364–372, July 2019.
- [254] P. Subedi and K. Walsh. Assessment of sugar and starch in intact banana and mango fruit by SWNIR spectroscopy. *Postharvest Biology and Technology*, 62(3):238–245, Dec. 2011.
- [255] P. Subedi, K. Walsh, and P. Purdy. Determination of optimum maturity stages of mangoes using fruit spectral signatures. *Acta Horticulturae*, 992:521–527, May 2013.
- [256] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. Jorge Cardoso. Generalised Dice Overlap as a Deep Learning Loss Function for Highly Unbalanced Segmentations. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 240–248, Cham, 2017. Springer International Publishing.
- [257] M. Sun, R. Zhao, X. Yin, L. Xu, C. Ruan, and W. Jia. FBoT-Net: Focal bottleneck transformer network for small green apple detection. *Computers and Electronics in Agriculture*, 205:107609, Feb. 2023.
- [258] T. Sun, W. Zhang, X. Gao, W. Zhang, N. Li, and Z. Miao. Efficient occlusion avoidance based on active deep sensing for harvesting robots. *Computers and Electronics in Agriculture*, 225:109360, Oct. 2024.
- [259] M.-T. Sánchez, M. J. De La Haba, M. Benitez-Lopez, J. Fernandez-Novales, A. Garrido-Varo, and D. Perez-Marin. Non-destructive characterization and quality control of intact strawberries based on NIR spectral data. *Journal of Food Engineering*, 110(1):102–108, May 2012.
- [260] J. F. Sánchez-Sevilla, J. G. Vallarino, S. Osorio, A. Bombarely, D. Posé, C. Merchante, M. A. Botella, I. Amaya, and V. Valpuesta. Gene expression atlas of fruit ripening and transcriptome assembly from RNA-seq data in octoploid strawberry (*Fragaria* × *ananassa*). *Scientific Reports*, 7(1):13737, Oct. 2017. Publisher: Nature Publishing Group.
- [261] J. G. Tallada, M. Nagata, and T. Kobayashi. Non-Destructive Estimation of Firmness of Strawberries (*Fragaria\*ananassa* Duch.) Using NIR Hyperspectral Imaging. *Environment Control in Biology*, 44(4):245–255, 2006.



- [262] C. Tang, D. Chen, X. Wang, X. Ni, Y. Liu, Y. Liu, X. Mao, and S. Wang. A fine recognition method of strawberry ripeness combining Mask R-CNN and region segmentation. *Frontiers in Plant Science*, 14, July 2023. Publisher: Frontiers.
- [263] Y. Tang, M. Chen, C. Wang, L. Luo, J. Li, G. Lian, and X. Zou. Recognition and Localization Methods for Vision-Based Fruit Picking Robots: A Review. *Frontiers in Plant Science*, 11:510, May 2020.
- [264] H.-T. Thai, K.-H. Le, and N. L.-T. Nguyen. FormerLeaf: An efficient vision transformer for Cassava Leaf Disease detection. *Computers and Electronics in Agriculture*, 204:107518, Jan. 2023.
- [265] Y. Tian, G. Yang, Z. Wang, H. Wang, E. Li, and Z. Liang. Apple detection during different growth stages in orchards using the improved YOLO-V3 model. *Computers and Electronics in Agriculture*, 157:417–426, Feb. 2019.
- [266] M. Tkachenko, M. Malyuk, A. Holmanyuk, and N. Liubimov. Label Studio: Data labeling software, 2020.
- [267] O. P. Toon, M. A. Zakaria, A. F. Ab. Nasir, A. P.P. Abdul Majeed, C. Y. Tan, and L. C. Y. Ng. Autonomous Tomato Harvesting Robotic System in Greenhouses: Deep Learning Classification. *MEKATRONIKA*, 1(1):80–86, Jan. 2019.
- [268] I. Urbano Bron, R. Vasconcelos Ribeiro, M. Azzolini, A. Pedro Jacomino, and E. Caruso Machado. Chlorophyll fluorescence as a tool to evaluate the ripening of ‘Golden’ papaya fruit. *Postharvest Biology and Technology*, 33(2):163–173, Aug. 2004.
- [269] L. Van Herck, P. Kurtser, L. Wittemans, and Y. Edan. Crop design for improved robotic harvesting: A case study of sweet pepper harvesting. *Biosystems Engineering*, 192:294–308, Apr. 2020.
- [270] L. A. Varga, J. Makowski, and A. Zell. Measuring the Ripeness of Fruit with Hyperspectral Imaging and Deep Learning. *arXiv:2104.09808 [cs]*, Apr. 2021. arXiv: 2104.09808.
- [271] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is All you Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, volume 30 of *NIPS’17*, pages 6000–6010, Red Hook, NY, USA, Dec. 2017. Curran Associates, Inc.
- [272] A. Wang, W. Qian, A. Li, Y. Xu, J. Hu, Y. Xie, and L. Zhang. NVW-YOLOv8s: An improved YOLOv8s network for real-time detection and segmentation of tomato fruits at different ripeness stages. *Computers and Electronics in Agriculture*, 219:108833, Apr. 2024.
- [273] C. Wang, F. Ding, Y. Wang, R. Wu, X. Yao, C. Jiang, and L. Ling. Real-Time Detection and Instance Segmentation of Strawberry in Unstructured Environment. *Computers, Materials & Continua*, 78(1):1481–1501, 2024.

- [274] D. Wang and D. He. Fusion of Mask RCNN and attention mechanism for instance segmentation of apples under complex background. *Computers and Electronics in Agriculture*, 196:106864, May 2022.
- [275] J. Wang, Z. Zhang, L. Luo, H. Wei, W. Wang, M. Chen, and S. Luo. DualSeg: Fusing transformer and CNN structure for image segmentation in complex vineyard environment. *Computers and Electronics in Agriculture*, 206:107682, Mar. 2023.
- [276] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao. Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 548–558, Montreal, QC, Canada, Oct. 2021. IEEE.
- [277] X. Wang, R. Zhang, T. Kong, L. Li, and C. Shen. SOLOv2: Dynamic and Fast Instance Segmentation. In *Advances in Neural Information Processing Systems*, volume 33, pages 17721–17732. Curran Associates, Inc., 2020.
- [278] X. Wang, R. Zhang, C. Shen, T. Kong, and L. Li. Dense Contrastive Learning for Self-Supervised Visual Pre-Training. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3023–3032, Nashville, TN, USA, June 2021. IEEE.
- [279] Y. Wang, C. Yang, S. Li, L. Yang, Y. Wang, J. Zhao, and Q. Jiang. Volatile characteristics of 50 peaches and nectarines evaluated by HP-SPME with GC-MS. *Food Chemistry*, 116(1):356–364, Sept. 2009.
- [280] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, Apr. 2004.
- [281] Z. Wang, K. Walsh, and A. Koirala. Mango Fruit Load Estimation Using a Video Based MangoYOLO—Kalman Filter—Hungarian Algorithm Method. *Sensors*, 19(12):2742, June 2019.
- [282] G. I. Webb. Naïve Bayes. In C. Sammut and G. I. Webb, editors, *Encyclopedia of Machine Learning*, pages 713–714. Springer US, Boston, MA, 2011.
- [283] C.-B. Wei, S.-H. Liu, Y.-G. Liu, L.-L. Lv, W.-X. Yang, and G.-M. Sun. Characteristic Aroma Compounds from Different Pineapple Parts. *Molecules*, 16(6):5104–5112, June 2011.
- [284] X. Wei, F. Liu, Z. Qiu, Y. Shao, and Y. He. Ripeness Classification of Astringent Persimmon Using Hyperspectral Imaging Technique. *Food and Bioprocess Technology*, 7(5):1371–1380, May 2014.
- [285] H. Williams, D. Smith, J. Shahabi, T. Gee, M. Nejati, B. McGuinness, K. Black, J. Tobias, R. Jangali, H. Lim, M. Duke, O. Bachelor, J. McCulloch, R. Green, M. O’Connor,

- S. Gounder, A. Ndaka, K. Burch, J. Fourie, J. Hsiao, A. Werner, R. Agnew, R. Oliver, and B. A. MacDonald. Modelling wine grapevines for autonomous robotic cane pruning. *Biosystems Engineering*, 235:31–49, Nov. 2023.
- [286] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon. CBAM: Convolutional Block Attention Module. In *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part VII*, pages 3–19, Berlin, Heidelberg, Sept. 2018. Springer-Verlag.
- [287] D. Worasawate, P. Sakunasinha, and S. Chiangga. Automatic Classification of the Ripeness Stage of Mango Fruit Using a Machine Learning Approach. *AgriEngineering*, 4(1):32–47, Jan. 2022.
- [288] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and Ross Girshick. Detectron2, 2019.
- [289] WWF. Reducing Food Waste is Imperative: Solutions for Food Loss, 2024.
- [290] WWF-UK. *Driven to waste: The Global Impact of Food Loss and Waste on Farms*. WWF-UK, June 2021.
- [291] X. Xia, X. Chai, Z. Li, N. Zhang, and T. Sun. MTYOLOX: Multi-transformers-enabled YOLO for tree-level apple inflorescences detection and density mapping. *Computers and Electronics in Agriculture*, 209:107803, June 2023.
- [292] B. Xiao, M. Nguyen, and W. Q. Yan. Fruit ripeness identification using YOLOv8 model. *Multimedia Tools and Applications*, Aug. 2023.
- [293] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo. SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers, Oct. 2021. arXiv:2105.15203 [cs].
- [294] Z. Xie, Z. Zhang, Y. Cao, Y. Lin, J. Bao, Z. Yao, Q. Dai, and H. Hu. SimMIM: a Simple Framework for Masked Image Modeling. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9643–9653, June 2022. ISSN: 2575-7075.
- [295] R. Xin, X. Liu, C. Wei, C. Yang, H. Liu, X. Cao, D. Wu, B. Zhang, and K. Chen. E-Nose and GC-MS Reveal a Difference in the Volatile Profiles of White- and Red-Fleshed Peach Fruit. *Sensors*, 18(3):765, Mar. 2018.
- [296] Y. Xiong, Y. Ge, L. Grimstad, and P. J. From. An autonomous strawberry-harvesting robot: Design, development, integration, and field evaluation. *Journal of Field Robotics*, 37(2):202–224, Mar. 2020.
- [297] P. Xu, N. Fang, N. Liu, F. Lin, S. Yang, and J. Ning. Visual recognition of cherry tomatoes in plant factory based on improved deep instance segmentation. *Computers and Electronics in Agriculture*, 197:106991, June 2022.

- [298] Y. Peng and R. Lu. An LCTF-based multispectral imaging system for estimation of apple fruit firmness: Part II. Selection of optimal wavelengths and development of prediction models. *Transactions of the ASABE*, 49(1):269–275, 2006.
- [299] S. Yang, W. Wang, S. Gao, and Z. Deng. Strawberry ripeness detection based on YOLOv8 algorithm fused with LW-Swin Transformer. *Computers and Electronics in Agriculture*, 215:108360, Dec. 2023.
- [300] C.-H. Yeh, C.-Y. Hong, Y.-C. Hsu, T.-L. Liu, Y. Chen, and Y. LeCun. Decoupled Contrastive Learning. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVI*, pages 668–684, Berlin, Heidelberg, Oct. 2022. Springer-Verlag.
- [301] J. Yu, Y. Bai, S. Yang, and J. Ning. Stolon-YOLO: A detecting method for stolon of strawberry seedling in glass greenhouse. *Computers and Electronics in Agriculture*, 215:108447, Dec. 2023.
- [302] W. Yu, M. Luo, P. Zhou, C. Si, Y. Zhou, X. Wang, J. Feng, and S. Yan. MetaFormer is Actually What You Need for Vision. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10809–10819, June 2022. ISSN: 2575-7075.
- [303] X. Yu, H. Lu, and D. Wu. Development of deep learning method for predicting firmness and soluble solid content of postharvest Korla fragrant pear using Vis/NIR hyperspectral reflectance imaging. *Postharvest Biology and Technology*, 141:39–49, July 2018.
- [304] Y. Yu, K. Zhang, L. Yang, and D. Zhang. Fruit detection for strawberry harvesting robot in non-structural environment based on Mask-RCNN. *Computers and Electronics in Agriculture*, 163:104846, Aug. 2019.
- [305] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. Barlow Twins: Self-Supervised Learning via Redundancy Reduction, June 2021. arXiv:2103.03230 [cs].
- [306] T. Zeng, S. Li, Q. Song, F. Zhong, and X. Wei. Lightweight tomato real-time detection method based on improved YOLO and mobile deployment. *Computers and Electronics in Agriculture*, 205:107625, Feb. 2023.
- [307] Y. Zhai, Z. Gao, Y. Zhou, J. Li, Y. Zhang, and Y. Xu. Green fruit detection methods: Innovative application of camouflage object detection and multilevel feature mining. *Computers and Electronics in Agriculture*, 225:109356, Oct. 2024.
- [308] J. Zhang, L. He, M. Karkee, Q. Zhang, X. Zhang, and Z. Gao. Branch detection for apple trees trained in fruiting wall architecture using depth features and Regions-Convolutional Neural Network (R-CNN). *Computers and Electronics in Agriculture*, 155:386–393, Dec. 2018.
- [309] K. Zhang, K. Lammers, P. Chu, Z. Li, and R. Lu. System design and control of an apple harvesting robot. *Mechatronics*, 79:102644, Nov. 2021.

- [310] L. Zhang, J. Jia, G. Gui, X. Hao, W. Gao, and M. Wang. Deep Learning Based Improved Classification System for Designing Tomato Harvesting Robot. *IEEE Access*, 6:67940–67950, 2018.
- [311] M. Zhang, B. Zhang, H. Li, M. Shen, S. Tian, H. Zhang, X. Ren, L. Xing, and J. Zhao. Determination of bagged ‘Fuji’ apple maturity by visible and near-infrared spectroscopy combined with a machine learning algorithm. *Infrared Physics & Technology*, 111:103529, Dec. 2020.
- [312] Q. Zhang and Y.-B. Yang. ResT: An Efficient Transformer for Visual Recognition. In *Proceedings of the 35th International Conference on Neural Information Processing Systems (NeurIPS 2021)*, volume 34, pages 15475–15485. Curran Associates, Inc., 2021.
- [313] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid Scene Parsing Network, Apr. 2017. arXiv:1612.01105 [cs].
- [314] R. Zhao, Y. Zhu, and Y. Li. CLA: A self-supervised contrastive learning method for leaf disease identification with domain adaptation. *Computers and Electronics in Agriculture*, 211:107967, Aug. 2023.
- [315] Z. Zhao, Y. Hicks, X. Sun, and C. Luo. Peach ripeness classification based on a new one-stage instance segmentation model. *Computers and Electronics in Agriculture*, 214:108369, Nov. 2023.
- [316] C. Zheng, P. Chen, J. Pang, X. Yang, C. Chen, S. Tu, and Y. Xue. A mango picking vision algorithm on instance segmentation and key point detection from RGB images in an open orchard. *Biosystems Engineering*, 206:32–54, June 2021.
- [317] Y.-Y. Zheng, J.-L. Kong, X.-B. Jin, X.-Y. Wang, T.-L. Su, and M. Zuo. CropDeep: The Crop Vision Dataset for Deep-Learning-Based Classification and Detection in Precision Agriculture. *Sensors*, 19(5):1058, Mar. 2019.
- [318] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2921–2929, Las Vegas, NV, USA, June 2016. IEEE.
- [319] L. Zhou, Y. Zhang, H. Chen, G. Sun, L. Wang, M. Li, X. Sun, P. Feng, L. Yan, L. Qiu, Y. Li, and Y. Ma. Soybean yield estimation and lodging classification based on UAV multi-source data and self-supervised contrastive learning. *Computers and Electronics in Agriculture*, 230:109822, Mar. 2025.
- [320] W. Zhou, Y. Cui, H. Huang, H. Huang, and C. Wang. A fast and data-efficient deep learning framework for multi-class fruit blossom detection. *Computers and Electronics in Agriculture*, 217:108592, Feb. 2024.

- [321] X. Zhou, W. S. Lee, Y. Ampatzidis, Y. Chen, N. Peres, and C. Fraisse. Strawberry Maturity Classification from UAV and Near-Ground Imaging Using Deep Learning. *Smart Agricultural Technology*, 1:100001, Dec. 2021.
- [322] J. Zhu, R. M. Moraes, S. Karakulak, V. Sobol, A. Canziani, and Y. LeCun. TiCo: Transformation Invariance and Covariance Contrast for Self-Supervised Visual Representation Learning, June 2022. arXiv:2206.10698.
- [323] X. Zhu, F. Chen, Y. Zheng, C. Chen, and X. Peng. Detection of *Camellia oleifera* fruit maturity in orchards based on modified lightweight YOLO. *Computers and Electronics in Agriculture*, 226:109471, Nov. 2024.
- [324] V. Ziosi, M. Noferini, G. Fiori, A. Tadiello, L. Trainotti, G. Casadoro, and G. Costa. A new index based on vis spectroscopy to characterize the progression of ripening in peach fruit. *Postharvest Biology and Technology*, 49(3):319–329, Sept. 2008.
- [325] M. Zude. Comparison of indices and multivariate models to non-destructively predict the fruit chlorophyll by means of visible spectrometry in apple fruit. *Analytica Chimica Acta*, 481(1):119–126, Mar. 2003.
- [326] M. Zude, B. Herold, J.-M. Roger, V. Bellon-Maurel, and S. Landahl. Non-destructive tests on the prediction of apple fruit flesh firmness and soluble solids content on tree and in shelf life. *Journal of Food Engineering*, 77(2):254–260, Nov. 2006.
- [327] M. Zude-Sasse, I. Truppel, and B. Herold. An approach to non-destructive apple fruit chlorophyll determination. *Postharvest Biology and Technology*, 25(2):123–133, June 2002.

# Appendix

This appendix includes several technical points raised during the viva examination.

## LightStraw

- **Question:** Your architecture borrows a lot from VTs, is MRCNN a fair comparison benchmark?

**Answer:** Although Mask R-CNN was proposed in 2017, it is still a widely used instance segmentation model in various applications. On the other hand, similar work on StrawDI\_Db1 was based on Mask R-CNN, which makes it a suitable baseline for comparison. Therefore, Mask R-CNN is a fair comparison benchmark.

## AppleSSL

- **Question:** What is the occlusion ratio that can be confidently dealt with and makes practical sense?

**Answer:** It is assumed that there was no clear boundary between confident and unconfident occlusion ratios, as it depended on the specific occasion. As a result, in general, a ratio of less than 60% was considered workable for most occluded applications.

- **Question:** How robust are the proposed distance metrics (e.g. have they been used in other applications), and are there any alternatives?

**Answer:** The proposed distance metrics were based on cosine similarity, which was robust and effective in self-supervised learning. As the distance metrics were custom-designed for apple ripeness estimation with limited labelled data, they have not been applied to other tasks. Alternatives include Mahalanobis distance and Silhouette score, but they do not align well with this task.

- **Question:** There was some attempt at involving experts in the ripeness analysis, but it was not very clear what the purpose of that exercise was. This should be described in more detail and with clear outcomes/results.

**Answer:** There were some volunteers, including professionals and normal consumers, involved in the ripeness analysis. The purpose of this exercise was to collect subjective opinions on apple ripeness, which were then used to validate the proposed self-supervised learning framework. As this test was simple and causal, the results were not included.



- **Question:** There was a lack of comparisons to annotated instances corresponding to different ripeness levels. At least some commentary on that would provide additional insights.

**Answer:** In this work, only 20 fully unripe and 20 fully ripe apples were annotated. It was very subjective to annotate different ripeness levels (e.g. 50%), which was the problem this work aimed to solve. This work focused more on the global overview instead of the local individual comparison. Therefore, there were no comparisons to annotated instances at different ripeness levels.

- **Question:** Is 1% a good figure to cite? Is 1% not dependent on the length of the dataset?

**Answer:** Yes, 1% is a valid and impactful figure to cite. It indicated that the manual annotation can be reduced to 1% of the original dataset size, which was a significant reduction. 1% was also dependent on the dataset, as it only made sense when paired with the dataset size.