APPLIED AI FOR AGRICULTURE - FRUIT RECOGNITION ALGORITHMS QUALITY ASSESSMENT FOR HARVESTING AUTOMATIC

by

Huynh Le Trung Minh

Phan Duc Hien

Tran Minh Huy

Nguyen Cang Truong

Tran Quoc Loi

FPT UNIVERSITY HO CHI MINH CITY

FPT UNIVERSITY

APPLIED AI FOR AGRICULTURE - FRUIT RECOGNITION ALGORITHMS QUALITY ASSESSMENT FOR HARVESTING AUTOMATIC

by

Huynh Le Trung Minh

Phan Duc Hien

Tran Minh Huy

Nguyen Cang Truong

Tran Quoc Loi

Supervisor: Dr. Truong Hoang Vinh

A final year capstone project submitted in partial fulfillment of the requirement for the Degree of Bachelor of Artificial Intelligence in Computer Science

DEPARTMENT OF ITS

FPT UNIVERSITY HO CHI MINH CITY

November 2024

ACKNOWLEDGMENTS

This project would not have been possible without the invaluable support and knowledge shared by our supervisors, Dr. Truong Hoang Vinh. Their guidance shaped our approach and deepened our understanding, bringing new perspectives that greatly enriched our work. We also acknowledge the critical input of our reviewing teachers, whose observations helped us enhance our findings. Finally, we recognize the encouragement of our families, whose steady support gave us the strength and determination to complete this journey.

AUTHOR CONTRIBUTIONS

The authors confirm their contributions to the paper as follows: Conceptualization, Huynh Le Trung Minh; Methodology, Nguyen Cang Truong and Tran Quoc Loi; Software, Nguyen Cang Truong and Tran Quoc Loi; Formal Analysis, Huynh Le Trung Minh, Tran Minh Huy, and Phan Duc Hien; Investigation, Huynh Le Trung Minh, Tran Minh Huy, and Nguyen Cang Truong; Data Curation, Huynh Le Trung Minh and Tran Minh Huy; Writing—Original Draft Preparation, Huynh Le Trung Minh, Tran Minh Huy, Phan Duc Hien, Nguyen Cang Truong, and Tran Quoc Loi; Writing—Review and Editing, Nguyen Cang Truong and Tran Quoc Loi; Visualization, Huynh Le Trung Minh and Nguyen Cang Truong; Project Administration, Huynh Le Trung Minh. All authors have reviewed and approved the final version of the paper.

ABSTRACT

This study utilizes state-of-the-art models, particularly YOLOS, to detect fruit ripeness and disease stages for efficient agricultural management to reduce the cost of manual labours. YOLOS's precision in identifying small features enables accurate classification of ripeness and early disease symptoms across various fruits. Combined with other models like CNNs, our approach enhances robustness and adapts to diverse conditions in real-time applications. Experimental results show high accuracy in ripeness and disease detection, supporting non-destructive, scalable solutions for optimized crop quality and reduced losses in the supply chain.

CONTENTS

List of Tables

List of Figures

1. INTRODUCTION

1.1. Background and motivation

In the 21st century, where technological advances are overcoming traditional challenges with innovative methods that save a lot of labor and manpower, one of these challenges is the agricultural sector. When growers have to ensure that the quality and yield of fruits are of high quality, this is not a simple task due to its manual nature, fruits often grow in complex environments with many uncertain factors **[1]**. Nevertheless, the high cost of advanced equipment and the technical expertise required to operate these systems pose significant barriers, especially for small-scale farmers and producers in resource-limited settings **[2]**. This problem is even more difficult for local farms where lack of manpower adds to the cost burden.

In this context, to address these challenges, this study proposes a machine learning-based approach to fruit quality assessment, focusing specifically on developing methods that are both accurate and accessible. By leveraging deep learning techniques, especially convolutional neural networks (CNNs), which play a major role in modern agriculture with their powerful computing capabilities, they have been widely applied in agricultural tasks including the advent of fruit identification technology.

A review study has presented that deep learning-based recognition systems, such as CNNs and YOLOs, can achieve human-level accuracy and speed in detecting and assessing fruit quality. The main stages include image acquisition, preprocessing, feature extraction, segmentation, and recognition **[3]**.

Fruit identification, a remarkable identification technology, provides an effective alternative to the traditional and manual methods that are labor-intensive. Texture, shape, color, size, and volume are external quality characteristics **[4]**. Simple access, no additional hardware requirements, and can be easily used by individuals with just a camera. This process eliminates the need for manual fruit identification difficulties, reducing operating costs and regular hardware maintenance.

The benefits of a fruit identification system go beyond ease of use. The systems eliminate the possibility of human error, ensuring accurate and consistent fruit identification. In addition, the system provides valuable data insights, allowing individuals or organizations to monitor fruit conditions, identify patterns, and optimize resource allocation.

This data-driven approach drives improved decision-making and resource management, improving farm performance, helping individuals and organizations proactively intervene and address concerns about fruit quality and harvestability.

With technology constantly evolving, fruit identification systems are a testament to the power of innovation in traditional and manual processes. This helps ensure local farms have a more streamlined, accurate, data-driven approach. Fruit quality refers to a fruit's overall characteristics that determine its desirability, nutritional content, and safety for consumption**[5]**. High-quality fruits benefit growers and sellers economically, promote healthy eating habits, reduce healthcare costs, positively impact the environment, ensure food safety, and promote international trade **[6]**.

1.2. Problem statement

Traditionally, fruit identification is a process where human beings check the quality of the fruit and capture the condition of the fruit manually such as notes or markings. Problems in the inspection process can occur if human beings do not have much ability to assess, leading to situations such as confusion, omission, or mistakes in identifying the fruit. Causing a lot of time, effort, and cost for the farm. This is a serious problem in the fruit identification process leading to losses during harvest such as misjudging the quality of the fruit, which affects the entire quality of the remaining fruit when it is brought to the market. To solve the problem, we propose to use an intelligent and automatic fruit identification system to manage the quality of the fruit when harvested.

The system for recognizing and assessing the quality of harvested fruits works by comparing the characteristics of harvested fruits based on information extracted from a pre-trained database, analyzing the color of the fruit, and then evaluating and determining the quality of the fruit and saving it in the file.

1.3. Solution

The purpose of our project is to classify fruits and identify their quality based on images and videos, and easily record the quality assessment of the fruit. The main objectives of this work are:

- Classify fruits based on images and videos.
- Develop a deep learning model to recognize the quality of fruits from pre-trained data sets.
- Save the assessment.

To ensure that the system complies with the requirements, our team will carefully test, refine, and evaluate their performance. This approach will help to select an effective fruit quality assessment recognition model at harvest.

1.4. Objectives of thesis

- Task 1: Create or clean the dataset.
- Task 2: Enhance the existing model/architecture.
- Task 3: Train with at least 2 models.
- Task 4: Validate models on different datasets.
- Task 5: Deploy the model on computer or mobile devices.
- Task 6: Prepare the required documents.

2. RELATED WORK

This part presents a review of existing relevant literature on fruit detection methods, with a particular emphasis convolutional neural network (CNN) models. It also explores various approaches and solutions that have been proposed in this area.

CNN have also made significant improvements in image classification and object detection. In this regard, the object detection framework R-CNN **[7]**, its derivatives fast R-CNN **[8]**, faster R-CNN **[9]** and mask R-CNN **[10]** are extensively utilized in the literature. CNNs have strong generalization and they rely on a lot of data for training in order to prevent overfitting. A transfer learning technique is often used to achieve good performance with a relatively small set of training data.

Bargoti and Underwood **[11]** trained a CNN, where its output was the likelihood that each pixel belonged to a fruit. Following that, a binary mask for segmentation was created using the output result. Häni et al. **[12]** proposed the CNN segmentation model, named U-NET, for apple segmentation and circular Hough transform for fruit detection. This network was initially designed for the segmentation of medical images. For detection, the author reports an F1 score of 0.858.

Some more examples includes: Sa et al. **[13]** proposed faster R-CNN to detect peppers, apples, avocados, mangoes, strawberries, and oranges. But only the pepper images come from the orchard. The remaining fruit photos are taken from the internet (Google Images). Linjuan Ma et al. **[14]** proposed faster R-CNN with AlexNet backbone for fruit detection and achieved an accuracy of 92% on the Fruits-360 dataset. Longsheng Fu et al. **[15]** demonstrated a faster R-CNN model with ZFNet that achieved an Average Precision (AP) of 89.3% for detecting kiwifruit. Their results showed that the highest detection rate for separated fruits is 96.7%, followed by adjacent fruits with 94.3% and overlapped fruits with 85.6%. Their validation on 100 images containing 5,918 kiwifruits showed an overall detection accuracy of 92.3%, with an average detection time of 0.274 seconds per image (at a resolution of 2352×1568 pixels). Xiaochun Mai et al. **[16]** proposed a faster R-CNN model with classifier fusion for small fruit detection. At an IoU threshold of 0.2, the model achieved a recall of 0.9037, precision of 0.7539, and an average precision (AP) of 0.8490, resulting in an F1-score of 0.8221. This represents a significant improvement over the F1-score of 0.7750 from the previous method **[11]**. At an IoU threshold of 0.5, the model showed a recall of 0.8354, precision of 0.7586, AP of 0.7668, and a mean IoU of 0.5762, highlighting its strong performance in detecting almonds under various conditions.

Liu et al. **[17]** employed the mask R-CNN method to detect cucumber fruits, using ResNet-101 as the backbone. They enhanced the model by adjusting the scales and aspect ratios of the anchor boxes and obtained an F1 score of 0.894 for the test images. Weikuan Jia et al. **[18]**

proposed a mask R-CNN model enhanced with ResNet and DenseNet backbones for apple detection in orchards. The model achieved high precision of 97.31% and recall of 95.70% on a test set of 120 images, with improved recognition speed, while transfer learning with an ImageNet pre-trained model resulted in a correct recognition rate of 86.14% on 368 apple fruits. HongJun Wang et al. **[19]** introduced a mask R-CNN model enhanced with a bottom-up horizontal connection path and an improved feature pyramid structure for for fruit surface lesions detection on apple, peach, orange, and pear. The model achieved 95% detection accuracy with a detection speed of 2.6 frames per second on a GPU, outperforming fast R-CNN and SSD algorithms in both accuracy and speed. Ganesh et al. **[20]** presented a deep learning approach named Deep Orange, based on the state-of-the-art instance segmentation framework mask R-CNN to detect and and pixel-wise segment fruits, specifically orange. The approach uses multi-modal input data comprising of RGB and HSV images of the scene. The evaluation on 200 images showed that the RGB+HSV model achieved the highest precision (0.9753) and F1-score (0.8867) but had a lower recall (0.8128) compared to the RGB-only model (precision: 0.8947, recall: 0.8673, F1-score: 0.8808). The HSV-only model performed poorly with many false positives. The average inference time per image was 11ms. The RGB+HSV model also provided better pixel-wise segmentation (mask) compared to the RGB model. Their future work will further compare segmentation performance across those models.

In recent years, you only look once (YOLO), a CNN-based object detection framework, has gained popularity for fruit detection tasks. Koirala et al. **[21]** applied this model to identify mangoes in orchard images, achieving impressive evaluation metrics with F1 score and average precision (AP) of 0.96 and 0.98, respectively.

During our research, we have found that R-CNN, its derivatives, and YOLO models yield excellent results on object detection tasks in general and fruit detection tasks in particular (as listed above). So, in this section, we will summarize our findings about these state-of-the-art models:

- **Region-based Convolutional Neural Network (R-CNN)** serves as a foundational model that use region proposals to detect objects within images in the context of fruit detection. It applies a CNN to each proposed region to classify objects and refine bounding boxes, making it effective for precise fruit detection. However, due to its computational intensity and slower processing time, R-CNN is less commonly used in real-time applications, being more suited for scenarios requiring high accuracy over speed.

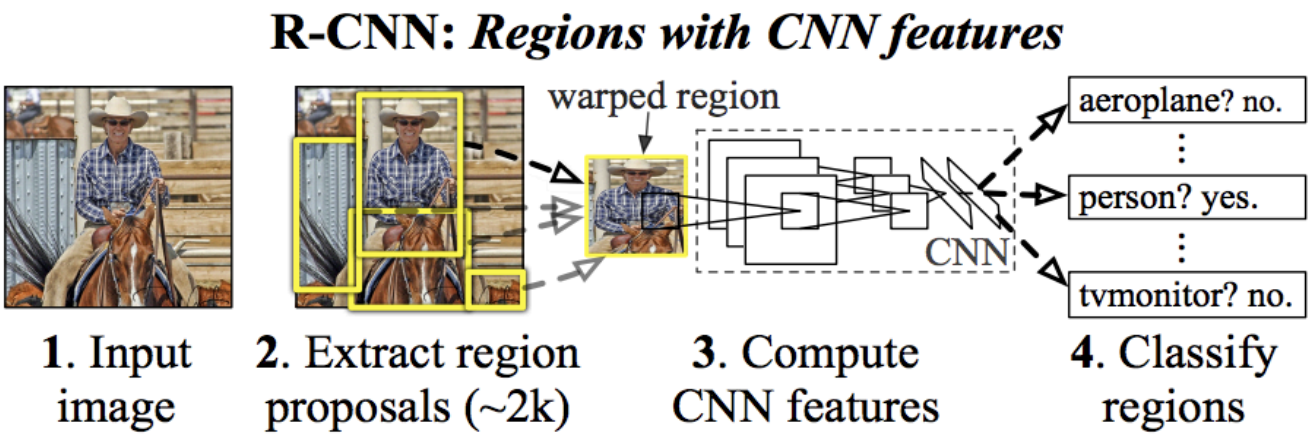


Figure 1. R-CNN Architecture

- **Fast Region-based Convolutional Neural Network (Fast R-CNN)** is applied in fruit detection to improve efficiency compared to the original R-CNN. It integrates region proposals and classification into a single network, reducing computation time. While it delivers accurate fruit detection results, its reliance on external region proposal algorithms, like Selective Search, can limit its speed and scalability compared to more advanced methods like Faster R-CNN or YOLO.

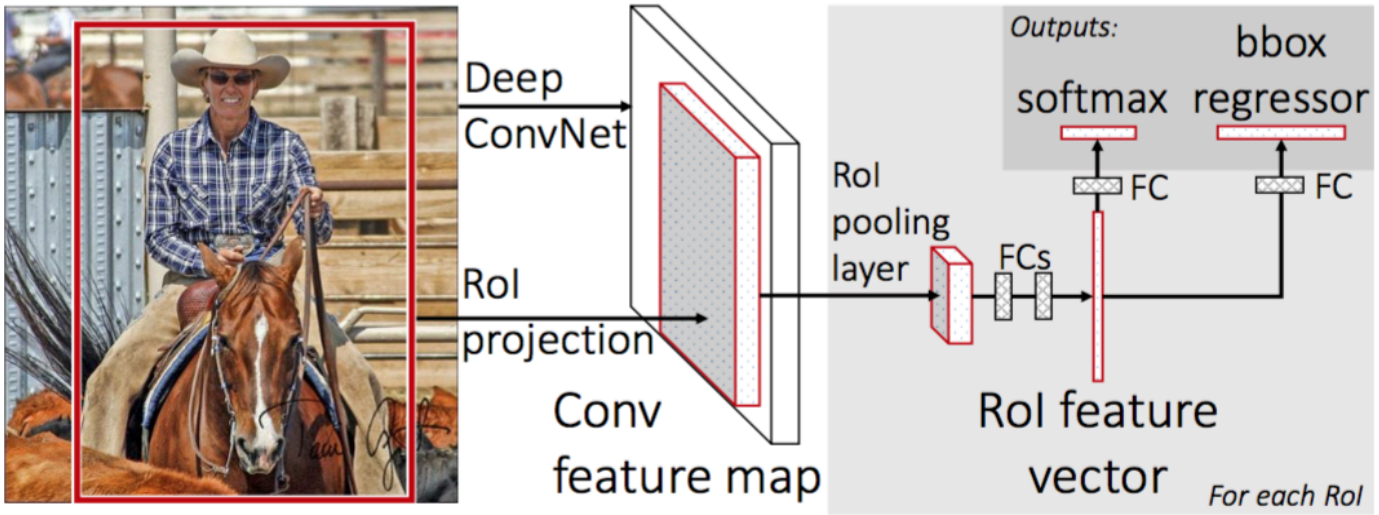


Figure 2. Fast R-CNN Architecture

- **Faster Region-based Convolutional Neural Network (Faster R-CNN)** is commonly used in fruit detection due to its integration of a Region Proposal Network (RPN), which streamlines the generation of region proposals. This makes it faster and more efficient than its predecessors while maintaining high accuracy. It is particularly effective in detecting fruits in complex environments and has been adapted with fine-tuned backbones and

anchor box adjustments to further enhance its performance. However, it is still slower than real-time detection models like YOLO.

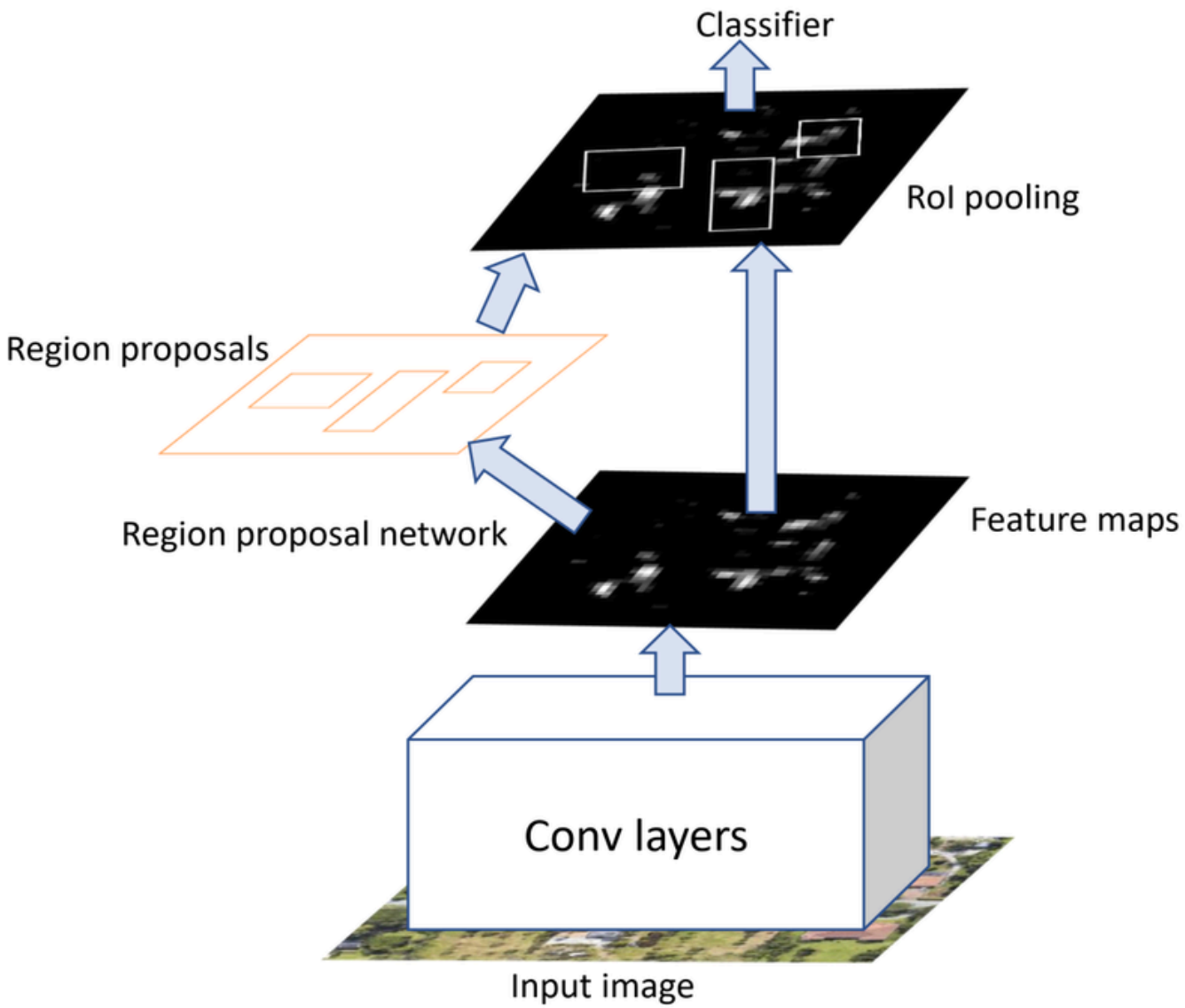


Figure 3. Faster R-CNN Architecture

- **Mask Region-based Convolutional Neural Network (Mask R-CNN)** is also extensively used in fruit detection for its ability to perform object detection and instance segmentation simultaneously. It provides precise localization and segmentation of fruits, making it ideal for complex environments. Researchers often optimize it with tailored backbones and anchor box adjustments to achieve high accuracy in identifying and segmenting fruits. However, its computational demands limit its use in real-time applications.

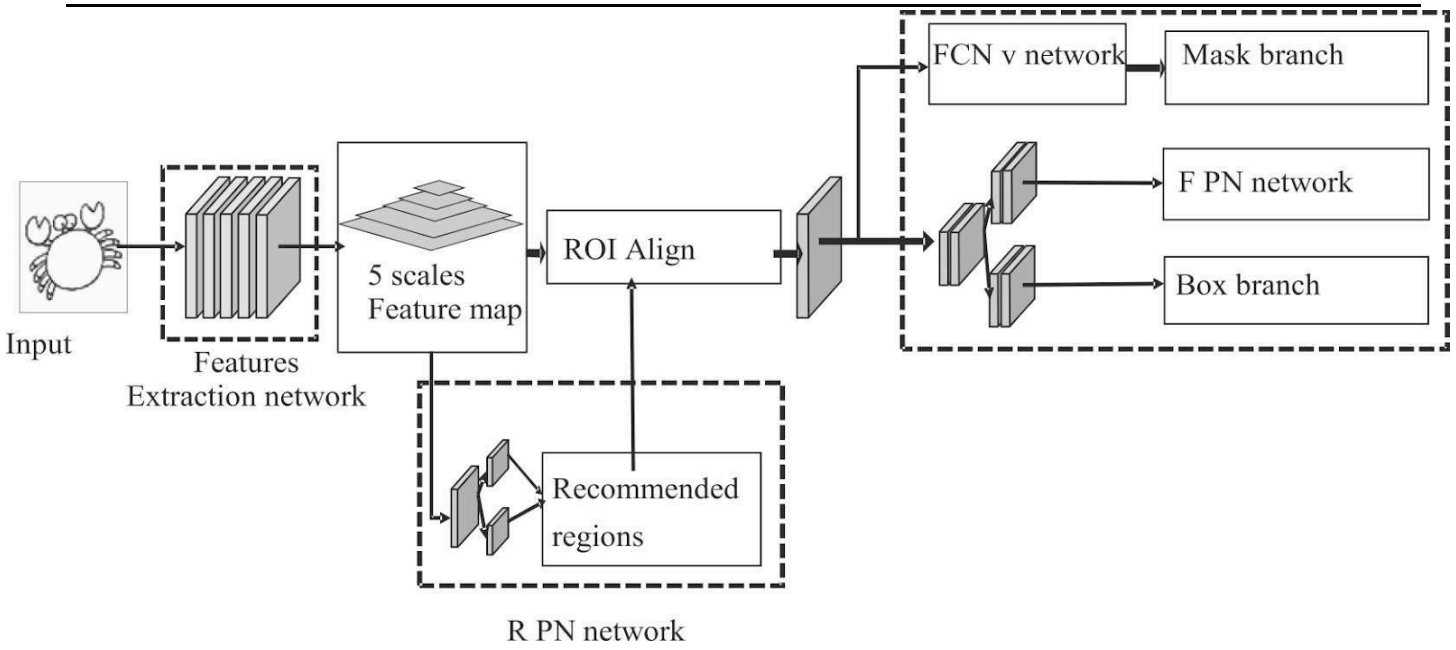


Figure 4. Mask R-CNN Architecture

- **You Only Look Once (YOLO)**, on the other hand, is widely used in fruit detection for its real-time speed and accuracy. It processes entire images in a single pass, making it efficient for detecting fruits in various environments. YOLO has been applied to tasks like mango and apple detection, achieving high precision and F1 scores. Its speed and precision make it ideal for practical, real-time applications relating to fruit detection.

We have combined our findings into this table:

Methods	Parameters	mAP50-95	Inference speed	Training Time
Fast R-CNN (ResNet-50 FPN, 1x)	41 millions	19.7%	0.029 seconds/image	4 hours
Faster R-CNN (ResNet-101 FPN, 3x)	61 millions	21.9%	0.051 seconds/image	21 hours
Mask R-CNN (ResNet-101 FPN, 3x)	63 millions	39.4%	0.056 seconds/image	1.06 days
YOLO (YOLOv11n)	2.6 millions	39.5%	0.0015 seconds/image	around 3 - 5 hours

Table 1: Comparison between R-CNN variants and YOLOv11n model performance on MS-COCO dataset

We find that YOLO models is a good balance between high precision, number of parameters and speed, boasting a much lower training time compared to R-CNN and its variants. These advantages combine with its ability to deploy on mobile devices has made it the best choice for us to look into further. Following this, we will examine various versions of YOLO and utilize them to train our custom dataset.

3. PROJECT MANAGEMENT PLAN

3.1 Research model & papers

From the start of our project, we have worked together to research relevant papers and existing models to quickly implement the project. This strategy has given us more time to expand and improve the project, with the aim of creating a well-rounded and complete graduation thesis.

Task name	Owner	Start date	End date	Status
Research the knowledge of real-time object detection Find relevant papers, documents (minimum 4 papers each member)	All members	Week1	Week2	Done
Research frameworks, library methods to support model	Minh ,Huy	Week2	Week3	Done
Search for a object detection dataset to train the model (YOLO)	All members	Week3	Week4	Done
Conducting Research on YOLO Model: Implementation, Training, and Comparative Analysis	All members	Week4	Week5	Done

Table 2. Research model & papers plan

3.2 Data preparation & training

After conducting research, training, aggregating results, comparing different YOLOV11 models as our primary model. Moving forward, we will continue to further research and expand upon this model. Simultaneously, we will also expand the dataset to enhance the model's accuracy, stability, and generalization capabilities.

Task name	Owner	Start date	End date	Status
Summary of results after training YOLO(5 ,7,8,9,10,11) Models Finalize the Selection of the Optimal Model for YOLOv11	All member	25/10	1/11	Done
DATA Mining Convert segment to yolo bounding boxes Split train val test data , remove null data , merge data	Minh , Huy, Hiền	25/10	1/11	Done
Label data	All member	2/10	28/10	Done
find early stopping function for model	Minh	5/10	7/10	Done
Create tool label data	Minh	25/9	1/10	Done

Table 3. Data preparation & training plan

3.3 Model and Dataset Refinement

After reaching this stage, our team had nearly completed the project. However, to further explore the model's capabilities, we decided to enhance both the model and the dataset. Our goal was to achieve superior performance compared to the original model.

Task name	Owner	Start date	End date	Status
Fine-tune YOLO model	All member	25/10	1/11	Done

Search for new datasets that match the model	Minh , Huy	25/10	29/10	Done
Data preprocessing	Minh , Huy, Hiền	25/10	29/10	Done

Table 4. Model and Dataset Refinement plan

Task name	Owner	Start date	End date	Status
file the project to push to the cloud	Minh	10/10	11/15	Done
Test the model on the web environment to ensure the performance and reliability of the model.	Minh	10/10	11/15	Done
Test the model on the ios app environment to ensure the performance and reliability of the model.	Minh	10/10	11/15	Done

Table 5. Deployment plan

3.5 Write Project Report

This part is like a diary of our project process. We record every process of making and implementing the project so that it is convenient to track and manage later.

Task name	Owner	Start date	End date	Status

Document the entire deblurring model research and development process from start to finish.	Trường , lợi	05/10	11/15	Done
Take notes of all meetings with supervisors and project reviews.	Minh, Huy	05/10	11/15	Done
Document the methods and techniques used in each phase of the project.	Trường , Lợi	05/10	11/15	Done

Table 6. Project report writing plan

3.6 Document

This section is to summarize the content in the process that we have done for the audience to see.

Task name	Owner	Start date	End date	Status
Abstract	Lợi	0/11	11/15	Done
Introduction	Huy	0/11	11/15	Done
Related Work	Trường	0/11	11/15	Done
Project Management Plan	Minh	0/11	11/15	Done
Materials and methods	Trường, Huy, Minh, Hiền	0/11	11/15	Done

Result	Hiền, Trường	0/11	11/15	Done
Discussion	Lợi	0/11	11/15	Done
Conclusion	Lợi	0/11	11/15	Done

Table 7. Work Schedule

4. MATERIALS AND METHODS

In this project, we mainly explore the use of YOLO models in fruit detection and classification. The YOLO (You Only Look Once) object detection and classification model has undergone significant evolution since its inception in 2015. Here's a condensed summary of YOLO models architecture and approach that we used:

4.1 YOLO: The Initial Real-time Object Detection Algorithm, The Starting Point Of All YOLO Models Afterwards

The first version of YOLO revolutionized object detection by framing it as a regression problem, diverging from the conventional classification methodology. It employed a single convolutional neural network (CNN) that segments the image into a grid, generating multiple predictions for each grid cell. Low-confidence predictions were filtered out, and overlapping bounding boxes were eliminated to produce the final results.

4.2 YOLOv5: Further Refinements for Precision and Efficiency

YOLOv5 made additional enhancements to increase both precision and efficiency. It utilized a Scaled-YOLOv4 backbone and integrated new methodologies such as CIOU loss and CSPDarknet53-PANet-SPP to improve precision. The updated YOLOv5 algorithm exhibited a 0.7% increase in mean average precision (mAP) over YOLOv4, while also decreasing its weight file size by 53.7 MB, making it a more effective tool for real-time object detection.

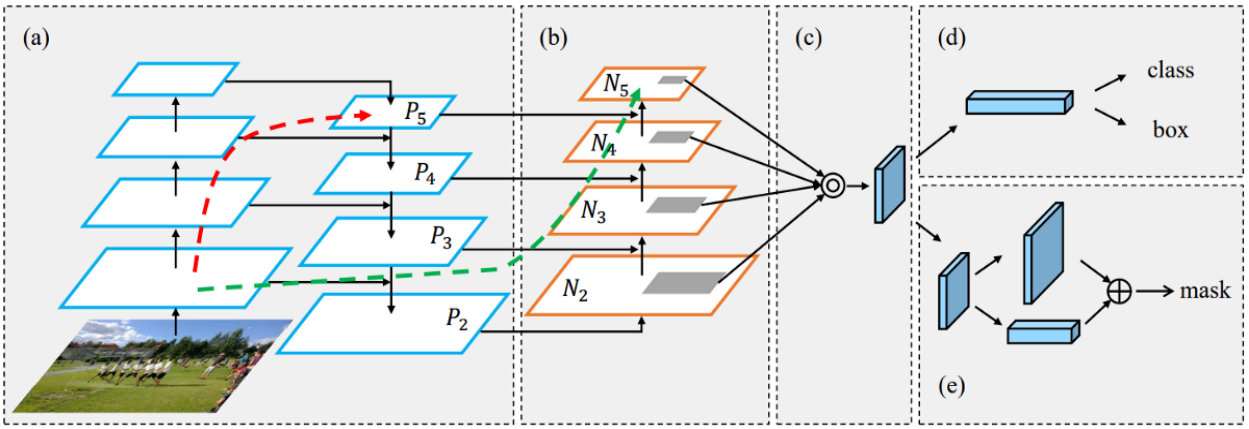


Figure 5. PANet Architecture
(a) FPN backbone. (b) Bottom-up path augmentation. (c) Adaptive feature pooling. (d) Box branch. (e) Fully-connected fusion

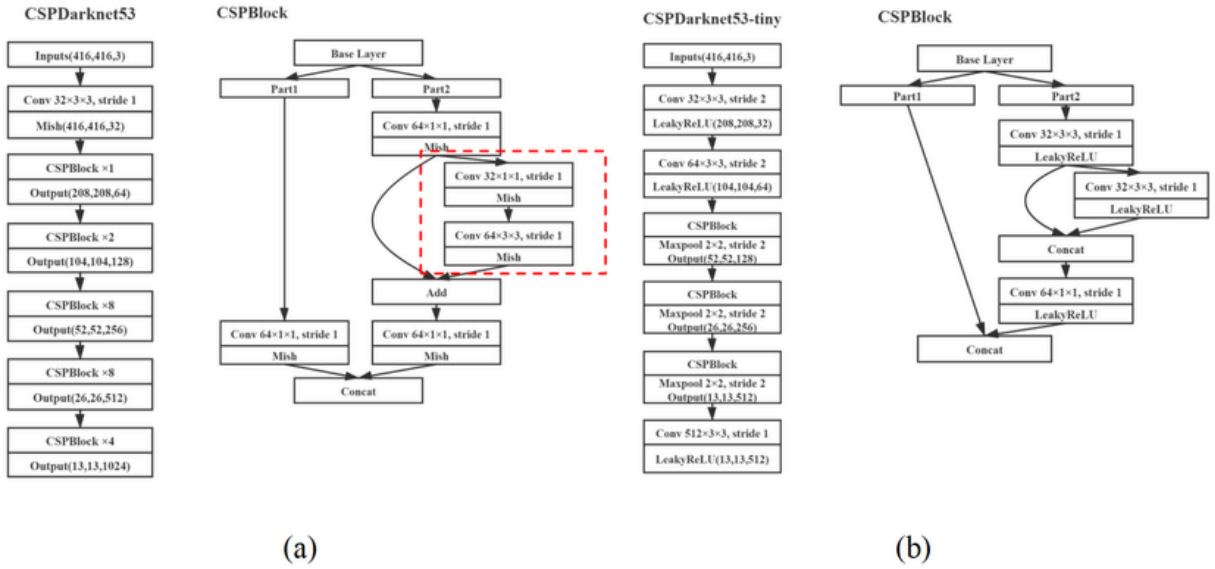


Figure 6. The structure of CSPDarknet53 (a) and CSPDarknet53-tiny (b)

4.3 YOLOv7: A Robust Tool for Real-time Object Detection

YOLOv7 introduces several key innovations that enhance its performance and efficiency. It uses a CSPDarknet-Z backbone, incorporates a Leaky ReLU activation function, and introduces TIoU loss, all of which improve its accuracy and capability in object detection. These advancements also include object-centric segmentation, further refining the model's precision. Compared to its predecessor, YOLOv6, YOLOv7 achieves a 1.0% increase in average precision (AP) while significantly reducing the model's weight file size by 70.5 MB. These improvements made YOLOv7 a highly effective and robust tool for real-time object detection tasks.

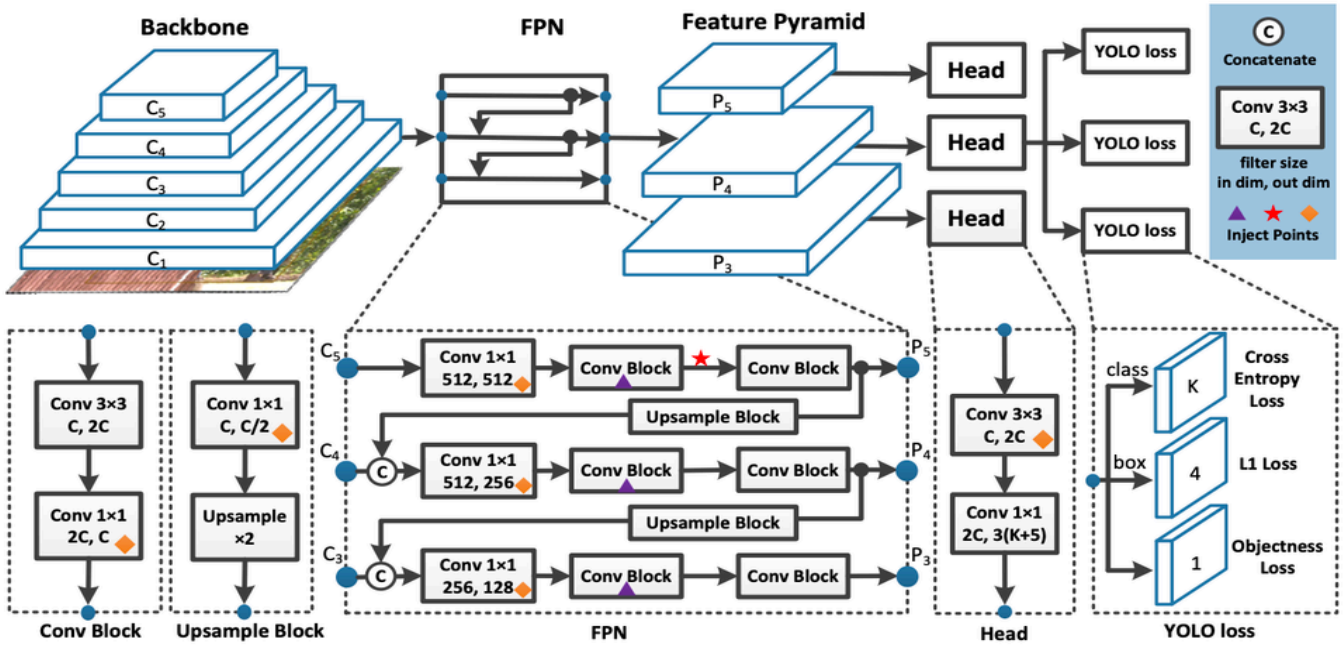


Figure 7. Model View of YOLOv7 Architecture

4.4 YOLOv8: Multi-Scale Detection and Enhanced Backbone

YOLOv8 introduced the CSPDarknet-AA backbone, an upgraded version of the CSPDarknet series, recognized for its effectiveness in object detection tasks. A key feature of YOLOv8 is its capability for multi-scale object detection, allowing the model to identify objects of various sizes within an image. It also employed the ELU activation function, which accelerates learning in deep neural networks by addressing the vanishing gradient issue, thus facilitating faster convergence. The model adopted GIoU loss, enhancing object localization precision by considering the shape and size of bounding boxes. YOLOv8 achieved a 1.2% improvement in average precision (AP) compared to YOLOv7, while also minimizing its weight file size by 80.6 MB, increasing efficiency for deployment in resource-limited settings.

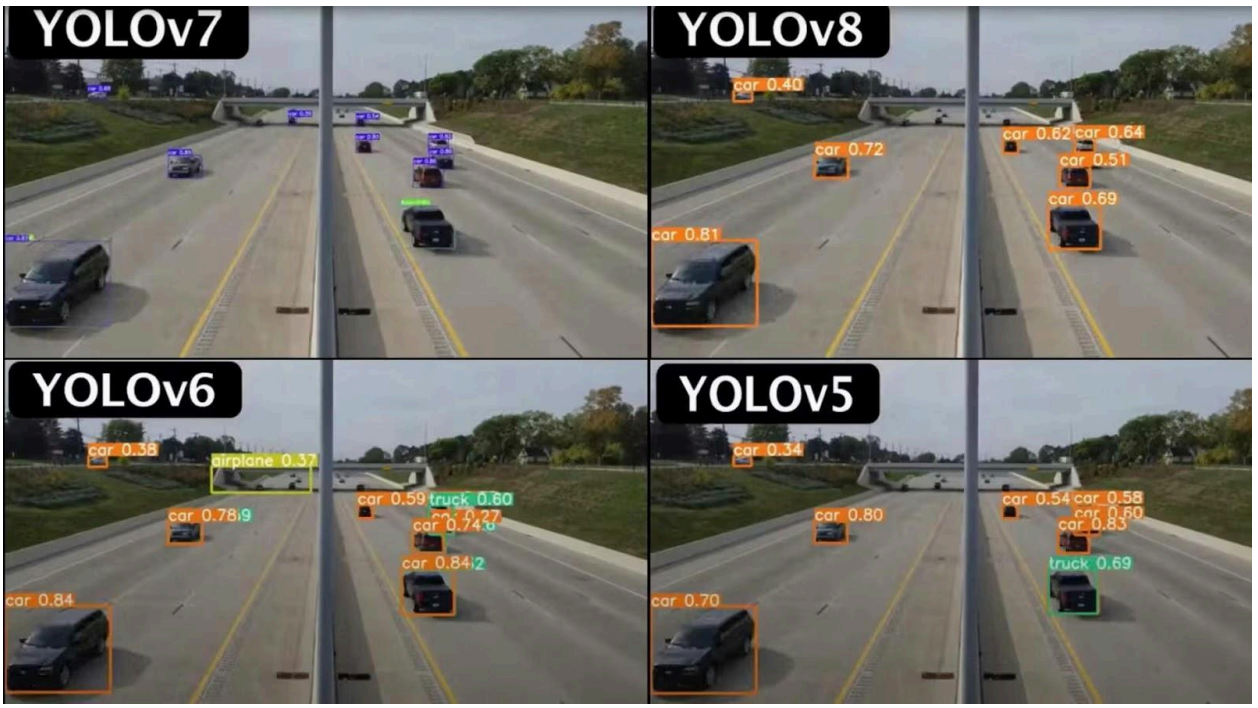


Figure 8. YOLOv8 Comparison with Latest YOLO models

4.5 YOLOv9: Advanced Architecture and Information Retention

Recently released YOLOv9 addressed information loss challenges prevalent in deep neural networks. By integrating Programmable Gradient Information (PGI) with the versatile GELAN architecture, YOLOv9 not only enhances the model's learning capabilities but also ensures the preservation of vital information throughout the detection process, resulting in remarkable accuracy and performance.

Key Features of YOLOv9:

- **Information Bottleneck Principle**: This principle highlights a critical challenge in deep learning: as data moves through successive layers, the potential for information loss escalates. YOLOv9 tackles this issue by employing Programmable Gradient Information (PGI), which helps retain essential data across the network's depth, leading to improved gradient generation and enhanced model convergence.

- **Reversible Functions**: A function is considered reversible if it can be inverted without losing information. YOLOv9 integrates reversible functions within its architecture to prevent information degradation, especially in deeper layers, ensuring the preservation of critical data necessary for object detection tasks.

4.6 YOLOv10: Higher Mean Average Precision

Released on May 23, 2024, YOLOv10 is a cutting-edge real-time object detection model developed by researchers at Tsinghua University. This latest version marks a significant leap forward in object detection technology, achieving lower latency and requiring fewer parameters compared to its predecessors.

In terms of performance, the YOLOv10 research paper highlights that "our YOLOv10-S is 1.8 times faster than RT-DETR-R18 while achieving comparable average precision on COCO, with 2.8 times fewer parameters and FLOPs. Additionally, in comparison to YOLOv9-C, YOLOv10-B demonstrates a 46% reduction in latency and a 25% decrease in parameters while maintaining equivalent performance." **[22]**

YOLOv10 introduces significant advancements in object detection through the implementation of a novel training methodology and various architectural enhancements. Here are some key components of how YOLOv10 works:

NMS-Free Training Strategy with Dual Label Assignments

Traditionally, YOLO models employ a one-to-many assignment strategy during training, assigning multiple positive samples to each ground truth instance. Known as Task-Aware Learning (TAL), this approach generates abundant supervisory signals that enhance optimization and overall model performance. However, it necessitates the use of Non-Maximum Suppression (NMS) during inference to filter out redundant predictions, leading to inefficiencies and increased latency.

To tackle this challenge, YOLOv10 introduces a dual label assignment strategy that integrates both one-to-many and one-to-one matching methods:

- **One-to-One Matching**: In this approach, each ground truth instance is paired with a single prediction, eliminating the need for NMS and enabling end-to-end deployment. While this method simplifies inference, it may result in reduced supervision, potentially impacting accuracy and slowing convergence.
- **One-to-Many Assignment**: This strategy provides richer supervisory signals but still requires NMS during inference.

YOLOv10 creatively combines these strategies by incorporating an additional one-to-one head that aligns with the structure and optimization objectives of the original one-to-many branch. During training, both heads are optimized simultaneously,

leveraging the extensive supervision from the one-to-many assignments. In the inference phase, the model relies solely on the one-to-one head, thereby eliminating the need for NMS and achieving high efficiency without incurring extra inference costs.

Consistent Matching Metric

A key component of the dual label assignment approach is the consistent matching metric utilized to quantitatively assess the level of concordance between predictions and actual ground truth instances. This metric integrates both the classification score and the Intersection over Union (IoU) of the predicted and true bounding boxes, which is defined as follows:

$$m(\alpha, \beta) = s \cdot p^{\alpha} \cdot IoU(\hat{b}, b)^{\beta}$$

where p is the classification score, $\hat{b}$ and b denote the bounding box of prediction and instance, respectively. The variable s represents the spatial prior indicating whether the predicted anchor point falls within the actual instance **[23, 24, 25, 26]**. α and β are two crucial hyperparameters that adjust the relative importance of the semantic prediction task compared to the location regression task. The one-to-many and one-to-one metrics are denoted as $m_{o2m} = m(\alpha_{o2m}, \beta_{o2m})$ and $m_{o2o} = m(\alpha_{o2o}, \beta_{o2o})$, respectively. These metrics are essential for label assignments and supervision for the two heads.

This standardized metric, applied uniformly to both the one-to-many and one-to-one branches, ensures cohesive supervision across the dual heads. By synchronizing the supervision of the one-to-one head with that of the one-to-many head, YOLOv10 improves the quality of predictions during the inference process, resulting in superior performance.

Classification Head, Decoupled Downsampling, Rank-Guided Block Design

The YOLO model architecture comprises several key elements: the stem, downsampling layers, various stages containing essential building blocks, and the classification head. The stem is designed to have a minimal computational cost. In YOLOv10, the emphasis is placed on optimizing the remaining three components to boost overall efficiency:

- **Lightweight Classification Head**

  YOLOv10 redesigns the classification head to reduce computational demands, using two depth wise separable convolutions (3×3) followed by a 1×1 convolution. This change addresses the previous issue where the classification head required significantly more resources than the regression head, thereby minimizing redundancy while maintaining performance.

- **Spatial-Channel Decoupled Downsampling**

Unlike traditional YOLO models that use 3×3 convolutions for downsampling and channel transformation simultaneously, YOLOv10 decouples these processes. It first applies pointwise convolution for channel modification, then depthwise convolution for spatial downsampling, significantly reducing computational costs and preserving information.

- **Rank-Guided Block Design**

YOLOv10 introduces a compact inverted block (CIB) structure, informed by rank analysis to optimize redundancy in deeper layers. This design uses depthwise convolutions for spatial mixing and efficient pointwise convolutions for channel mixing, enhancing overall architecture efficiency within the Efficient Layer Aggregation Network (ELAN).

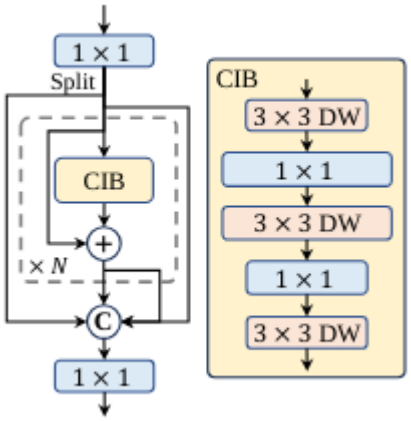


Figure 9. The compact inverted block (CIB)

Kernel Convolution and Self-Attention

YOLOv10 improves accuracy with minimal computational cost through large-kernel convolutions and a partial self-attention (PSA) mechanism.

- **Large-Kernel Convolution**

YOLOv10 employs large-kernel depthwise convolutions selectively in deeper stages, enhancing the receptive field without compromising shallow feature detection or increasing latency significantly.

- **Partial Self-Attention (PSA)**

To mitigate the high costs of traditional self-attention, YOLOv10 introduces PSA, which processes only one segment of features through efficient NPSA blocks, followed by concatenation. This method reduces computational complexity and enhances global representation learning while maintaining low overhead, particularly after Stage 4.

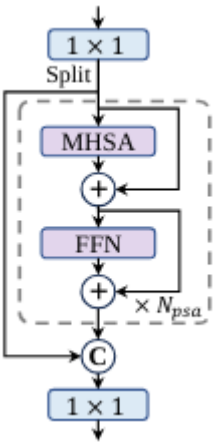


Figure 10. The partial self-attention module (PSA)

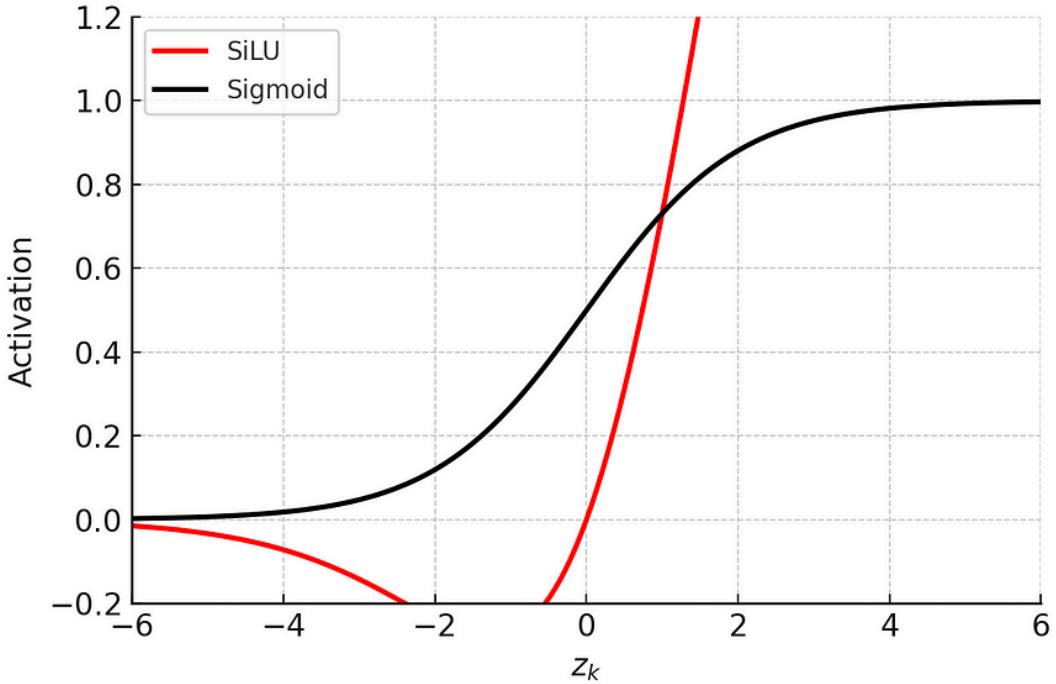
4.7 YOLOv11: Newest YOLO

The design of YOLOv11's architecture aims to enhance both speed and accuracy, drawing from the improvements established in previous YOLO versions, including YOLOv8, YOLOv9, and YOLOv10. Key architectural advancements in YOLOv11 focus on the C3K2 block, the SPFF module, and the C2PSA block, which collectively improve its capacity to handle spatial data while ensuring rapid inference.

4.7.1. Backbone

- **Convolutional Block**

This block is named as Conv Block, which processes dimensions c, h, and w using a 2D convolutional layer, followed by Batch Normalization and a SiLU Activation Function.



SiLU vs Sigmoid Activation Function

Figure 11. SiLU vs Sigmoid Activation Function

- **Bottle Neck**

  The Bottle Neck consists of convolutional blocks with a shortcut parameter for residual inclusion, similar to a ResNet Block. If the shortcut is False, the residual component is ignored.

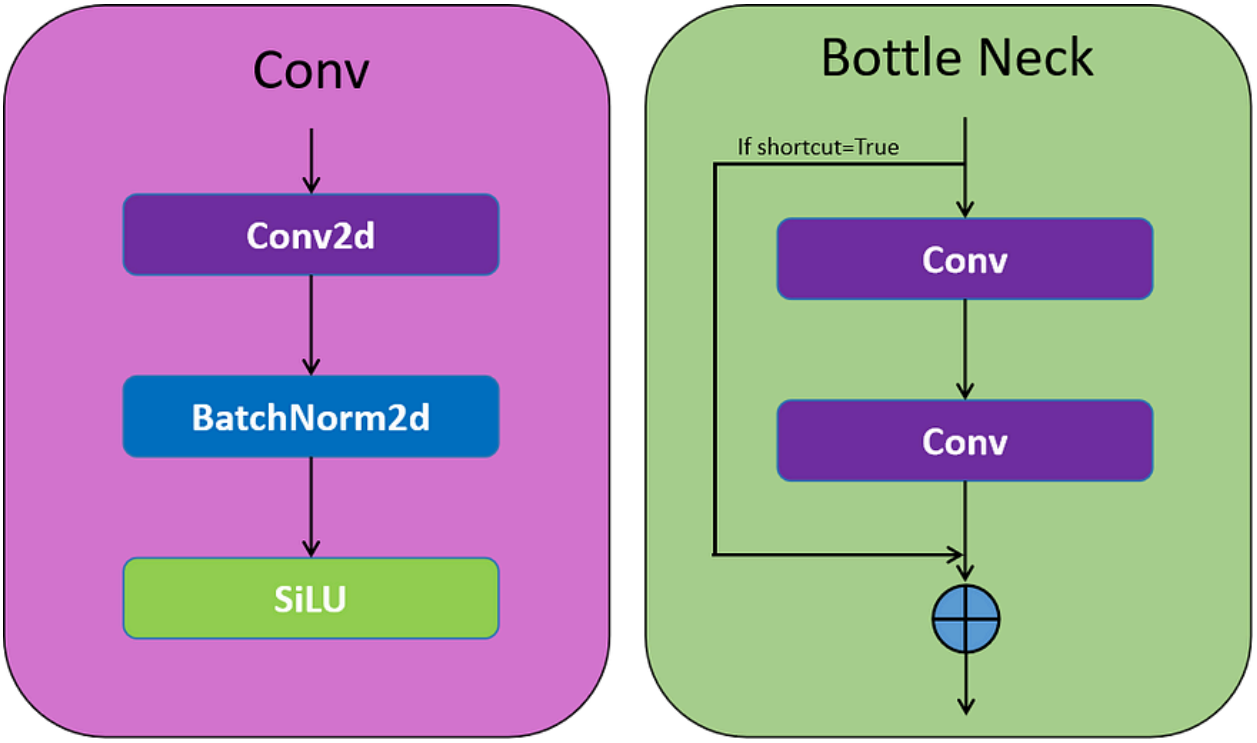


Figure 12. Convolutional Block and Bottle Neck Layer

- **C2F (YOLOv8)**

  The C2F block (Cross Stage Partial Focus) enhances efficiency and feature map preservation. It starts with a Conv Block, splits the output into two halves for 'n' Bottle Neck layers, and concatenates the outputs, followed by a final Conv Block, improving connectivity and reducing redundancy.

- **C3K2**

  YOLOv11 uses C3K2 blocks for feature extraction, employing smaller 3x3 kernels for efficient computation while capturing essential image features. The C3K2 block improves the earlier CSP (Cross Stage Partial) bottleneck by enhancing information flow through smaller kernel convolutions and merging processed feature maps, resulting in better feature representation with fewer parameters than YOLOv8's C2F blocks.

The C3K block processes input through a Conv block and 'n' Bottle Neck layers without splitting, concluding with a final Conv Block. The C3K2 utilizes the C3K block

structure, adding two Conv blocks at the start and end, interspersed with C3K blocks, to balance speed and accuracy while leveraging the CSP design.

4.7.2 Neck: Spatial Pyramid Pooling Fast (SPFF) and Upsampling

YOLOv11 retains the SPFF (Spatial Pyramid Pooling Fast) module, which enhances object detection across various scales, particularly for smaller objects that earlier YOLO versions struggled with. The SPFF module uses multiple max-pooling operations with different kernel sizes to aggregate contextual information, allowing for effective detection of small objects while maintaining real-time processing speeds.

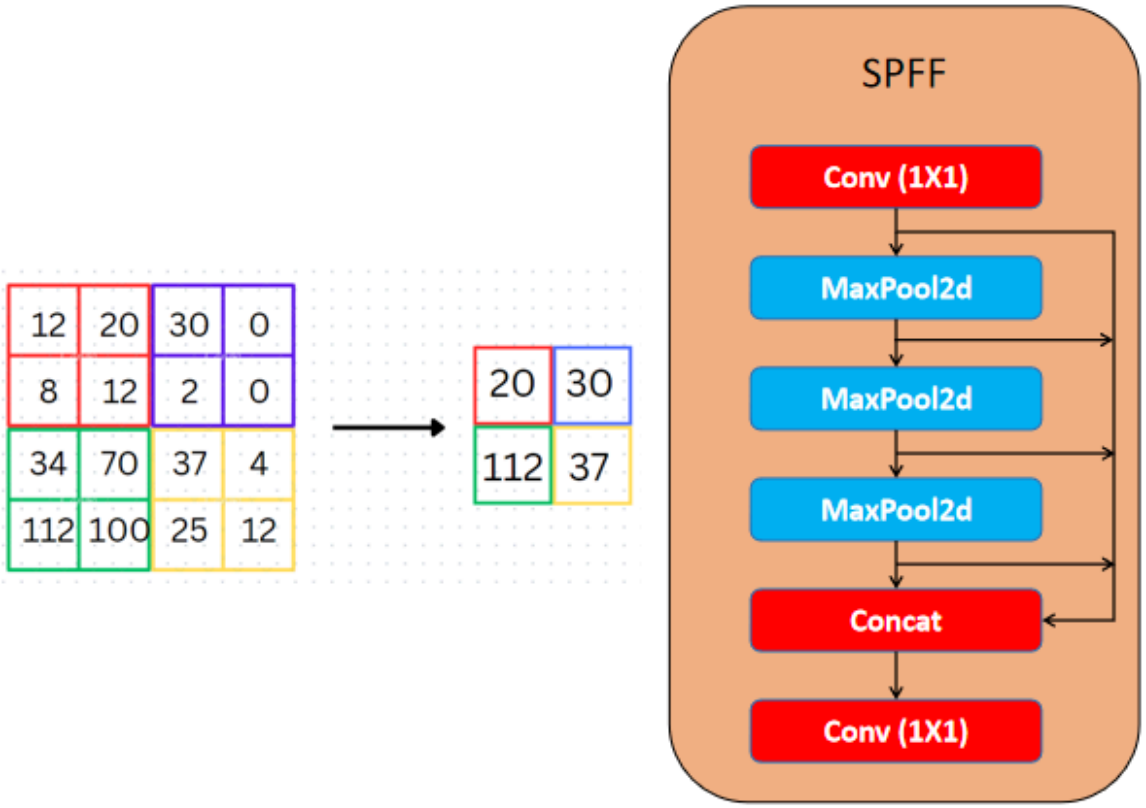


Figure 13. Spatial Pyramid Pooling Fast

4.7.3 Attention Mechanisms: C2PSA Block

YOLOv11 introduces the C2PSA block (Cross Stage Partial with Spatial Attention), which enhances the model's focus on important image areas, particularly smaller or partially obstructed objects.

- **Position-Sensitive Attention**

  This component implements position-sensitive attention in conjunction with feed-forward networks, improving feature extraction and processing. It processes input through an Attention layer, concatenates it with the output, and passes it through a Feed Forward Neural Network, followed by two Conv Blocks.

- **C2PSA**

The C2PSA block includes two Partial Spatial Attention (PSA) modules operating on separate branches of the feature map before concatenation. This design emphasizes spatial information while balancing computational efficiency and detection accuracy, allowing YOLOv11 to excel in detailed object detection compared to YOLOv8.

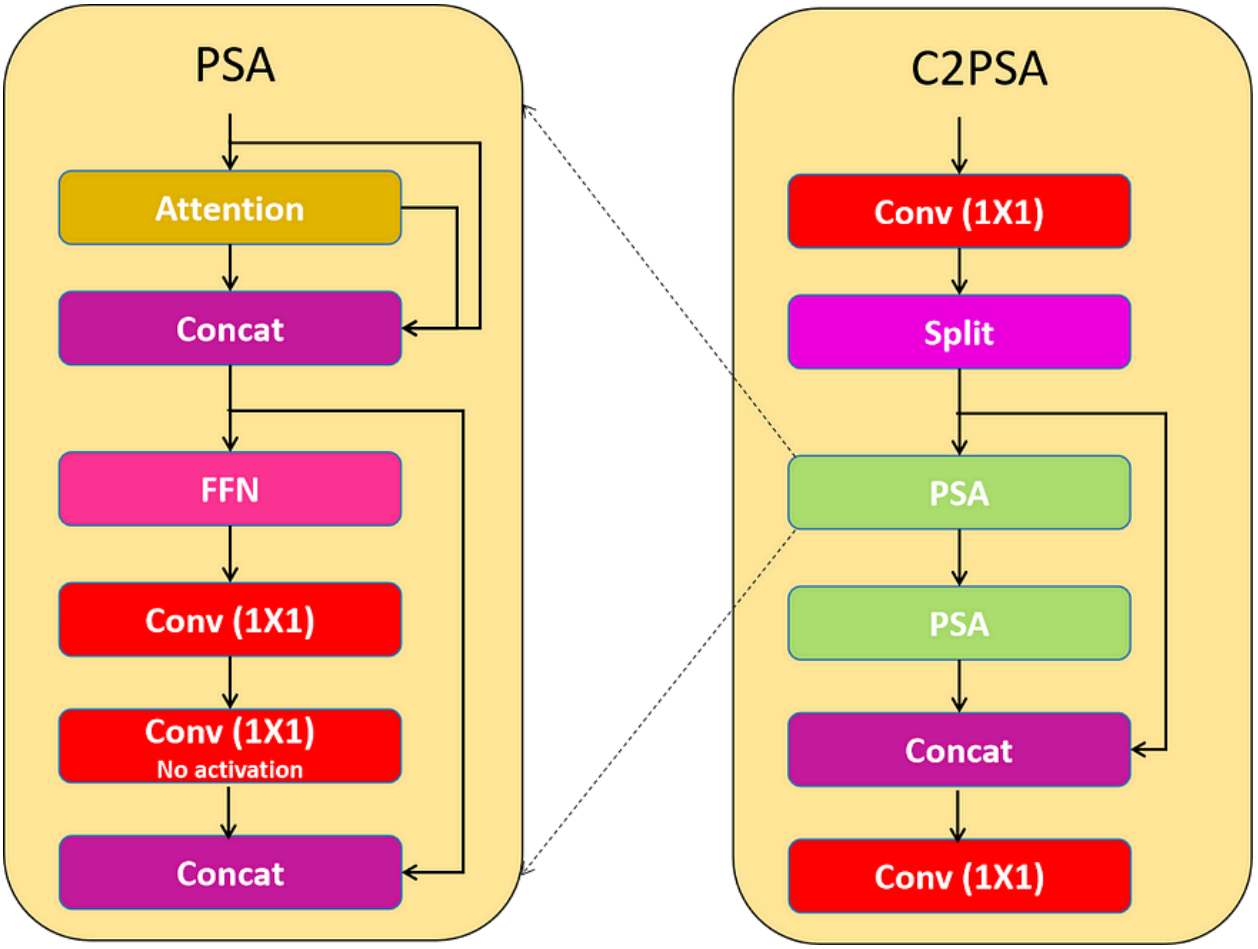


Figure 14. C2-Position Sensitive Attention Block (C2PSA)

4.7.4 Head: Detection and Multi-Scale Predictions

Continuing the tradition of its predecessors, YOLOv11 has a multi-scale prediction head that detects objects of various sizes. It generates detection boxes for three scales (low, medium, high) based on feature maps from the backbone and neck components.

The detection head provides predictions from three feature maps (P3, P4, and P5), ensuring precise detection of smaller objects (P3) and larger objects represented by higher-level features (P5).

4.8 Data Preparation

4.8.1. LaboroTomato:

The instance segmentation dataset is used for problems including object detection, instance segmentation, and semantic segmentation. It is employed in the field of agriculture.

This dataset includes 804 photos with 10610 tagged items in 6 classes: b_green, l_green, l_fully_ripened, and other: b_half_ripened, l_half_ripened, and b_fully_ripened.

Images in the LaboroTomato dataset **[27]** have pixel-level instance segmentation annotations. The instance segmentation job can be automatically converted to semantic segmentation (one mask for each class) due to its nature. We convert segmentation mask to bounding box to use this dataset for training with different YOLO versions.

Figure 15. Laboro tomato segment

Figure 16. Laboro tomato segment to bounding boxes

4.8.2 Tomato blossom end rot dataset :

Collected on roboflow and images downloaded from the internet and labeled by us. The dataset has a total of 491 images and 940 annotations.

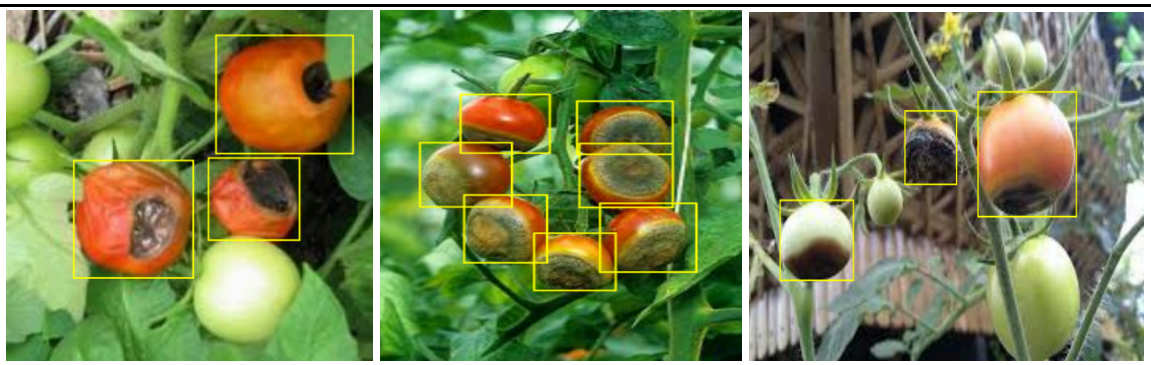


Figure 17. custom strawberry dataset

4.8.3 Strawberry dataset:

Collected on roboflow and google's data. The Dataset has 2756 images include 3 class strawberry ripe , strawberry unripe , strawberry half ripe :

Ripeness is labeled as follow:

- 0%-30% is strawberry unripe
- 30%-90% is strawberry half ripe
- 90%-100% is strawberry ripe

Total annotation

- strawberry Unripe : 2832
- strawberry Ripe : 2031
- strawberry Half ripe : 2448

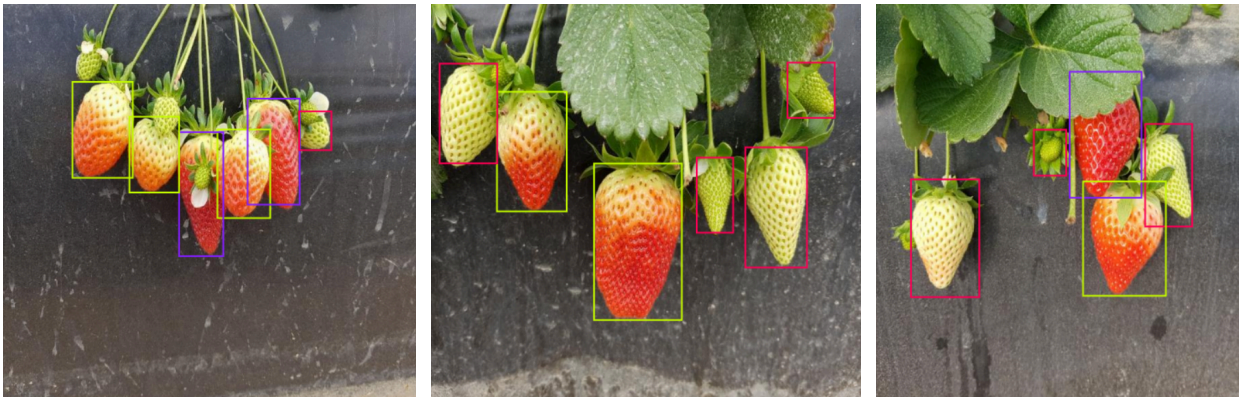


Figure 18. custom strawberry dataset 2

4.8.4 Dataset:

Dataset is made from all the above dataset combined with images from google and roboflow.

- **Laboto tomato** : 358 images includes 3 class l_fully_ripened, l_green, l_half_ripened
- **Strawberry custom** : 2689 images

- **Tomato blossom end rot custom** : 296 images and a few more images included

Total tomato-strawberry object detection (3397 images) :

1. Tomato unripe: 4596
2. Tomato ripe: 1361
3. Tomato half ripe: 1020
4. Straw unripe: 2730
5. Straw half ripe: 2404
6. Straw ripe: 1967
7. Tomato Blossom end rot: 579

4.9 Data Preprocessing

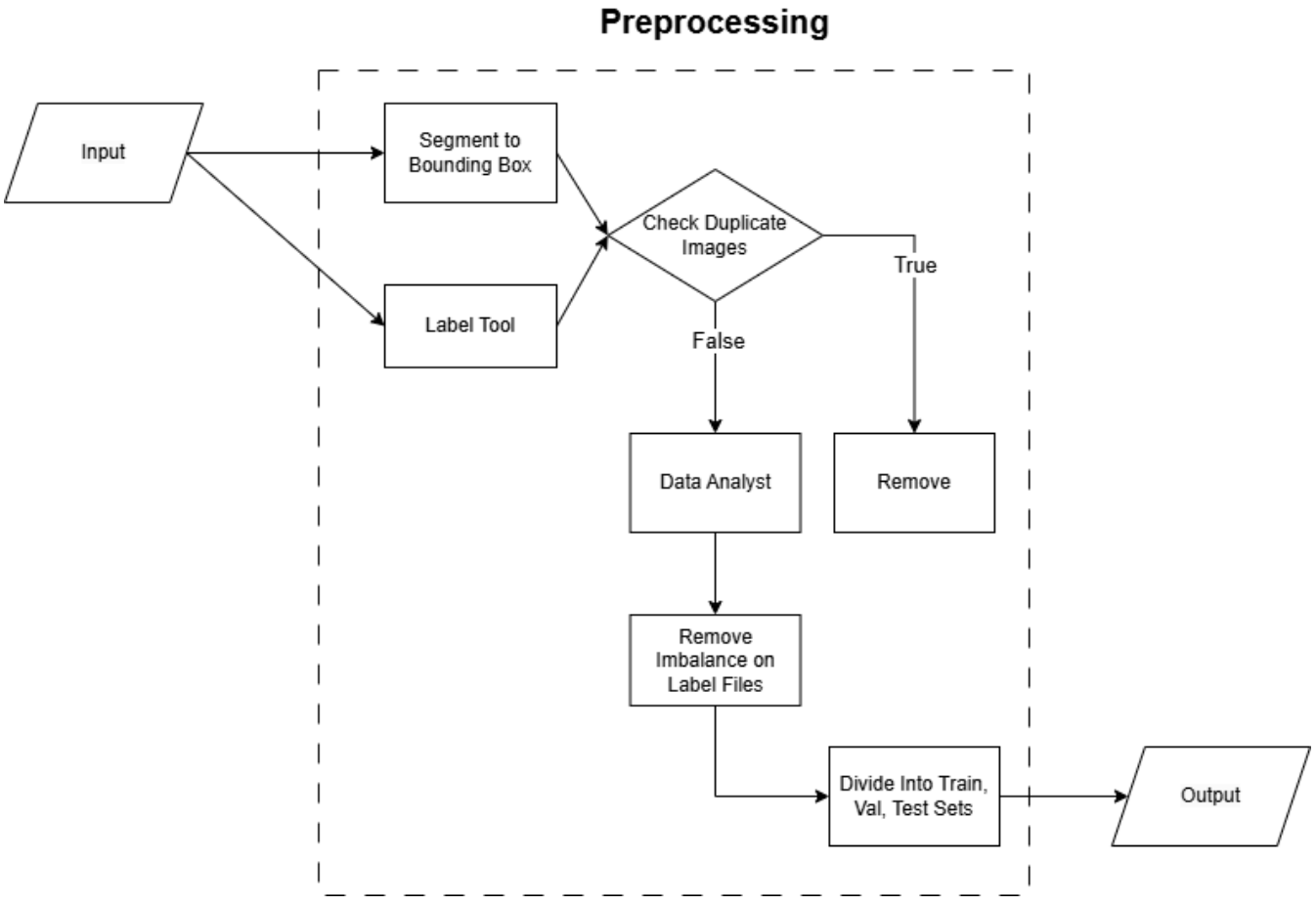


Figure 19. Data Preprocessing Diagram

4.9.1 Transform segmentation labels into bounding box labels

We first transform segmentation labels of Laboro dataset into bounding box format for labels of images in the dataset. The labels of images in the original dataset are in .json format, so we turned them into .txt format. First, we browsed through every object in the json file and skipped through any object with unidentified labels. Then, we determined the coordinates xmin, xmax, ymin, ymax of the bounding boxes to prepare for drawing

them. Afterwards, we calculated the positioning of bounding boxes according to YOLO format. Finally, we created an output line with YOLO format (class_id, x_center, y_center, width, height), removed .jpg and .json extensions from those processed files and saved results in the form of text files (with the extension .txt).

4.9.2 Remove duplicate images

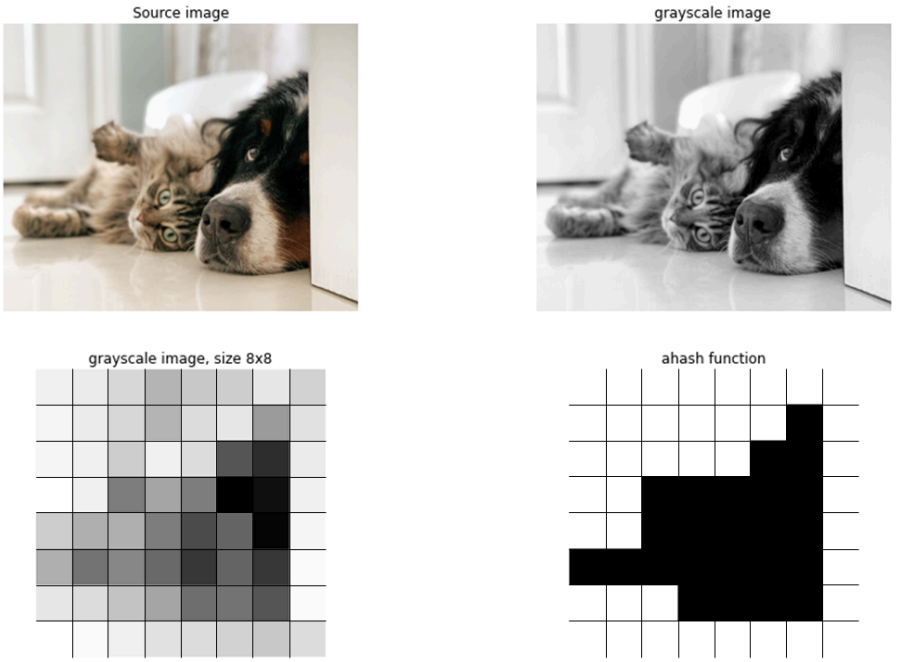


Figure 20. Average Hashing (aHash)

For the second step of data preprocessing, we scanned through the entire dataset to search for duplicate images and removed them. We made a Average Hashing (aHash) for each image for each image to uniquely identify its content. If two images have the same hash, they are duplicates. At the final step, we kept only one copy of the duplicate images and remove the rest into another folder, ensuring their corresponding annotations are also removed. With this processing, we ensure that all images are readable and not corrupted (e.g., broken files or unsupported formats).

4.9.3 Balance classes of the data

Into the third step, we realized that our dataset has the class imbalance problem with one class (i.e. class 1) has 6917 instances while the other two (i.e. class 0 and class 2) have 1870 and 1823 instances respectively. Our solution for the problem at hand is first removing images with only class 1 (zero instance on class 0 and class 2). After this step, 606 class 1 instances have been filtered out and we ended up with these instances 1870, 6311, 1823 for class 0, class 1, class 2 accordingly. Then, we continued to retain any class file with the initial condition that the number of class 1 is smaller than the total number

of class 0 plus class 2. For the rest of the class files, we devised a formula (a way) to sort these files based on the imbalance of every class in it. Our formula is as following:

|(class1 - class0) + (class1 - class 2) + (class0 - class2)|

where class1 is half-ripe class, class0 is green class, and class2 is ripe class. Those class files with the biggest imbalance indexes will be placed in the bottom of the sorted list and will be removed from the dataset. All the above processings will be repeated until one class reaches 1000 instances.

4.9.4 Divide the dataset into train, validation, test sets according to the ratio of 8:1:1

At the final step of modifying the dataset, we first set the split ratios for train, val and test folders as the common ratios of 8:1:1. Then, we used a set to track down all distributed class files to ensure that there is no file going into two separate folders at the same time. We also shuffled class files to create randomness for the final dataset. We copied the remaining label files and respective image files after the third processing step into folders according to the preset ratios. Finally, we counted the number of labels in each folder (train, val, test) and plotted out statistics to recheck the distribution of every label files into train, val, test folders.

4.9.5 System Configuration and Training Parameters

- Processor (CPU): Intel Core i7-12700K or equivalent
- Graphics Processing Unit (GPU): GPU P100
- Framework: PyTorch 2.0
- Epochs: 100 epochs (based on early stopping criteria)
- Batch Size: 16 (adjusted based on GPU memory).
- Learning Rate: 0.001 with cosine annealing for gradual reduction.
- Confidence Threshold: Set to 0.25 for initial training, with adjustments during evaluation.
- IoU Threshold: 0.5 for determining overlaps during training.
- Patience : After 5 epochs, if the validation loss does not decrease, stop to avoid overfitting.

momentum=0.937, weight_decay=0.0005, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4,, translate=0.1, scale=0.5, fliplr=0.5,mosaic=1.0, erasing=0.4, crop_fraction=1.0,

Training for 100–150 epochs provides sufficient iterations for convergence without overfitting. Early stopping criteria (monitoring validation loss and mAP) are used to terminate training if the model stops improving, preventing unnecessary training cycles.

The batch size determines how many images are processed simultaneously during training. A batch size of 16 strikes a balance between GPU memory limitations and stable

gradient updates. Larger batch sizes may result in out-of-memory (OOM) errors, while smaller sizes may lead to noisier gradient updates, slowing convergence.

The initial learning rate of 0.001 is a common starting point for YOLO models, providing a balance between quick convergence and avoiding large oscillations in the loss function

A confidence threshold of 0.25 ensures that the model only predicts bounding boxes for objects it is reasonably certain about. This reduces false positives during training and focuses learning on meaningful detections.

Intersection over Union (IoU) threshold determines whether a predicted bounding box overlaps sufficiently with the ground truth. A threshold of 0.5 is a standard starting point, balancing strictness and leniency. It ensures that the model learns to prioritize high-quality bounding boxes during training

5. RESULTS

Yolo tiny model comparison results on custom dataset:

Models	**Size**(pixels)	**Map of Val 50-95**	**Params** (M)	**GFLOPs** (B)
YOLOv5nu	640	64.2	2.6	7.1
YOLOv7t	640	60.2	6.2	13.2
YOLOv8n	640	62.4	3.2	8.2
YOLOv9t	640	66.9	2.0	7.6
YOLOv10n	640	56.7	2.3	8.2
YOLOv11n	640	**67.5**	**2.6**	**6.3**

Table 8: Yolo tiny model comparison results on custom dataset

According to the comparison, we see that the results yolo tiny (5,8,11) have fewer parameters but the results and accuracy are the highest in our custom dataset.

The following is an actual comparison when running the model to identify objects

The first will be the YOLOv5nu model

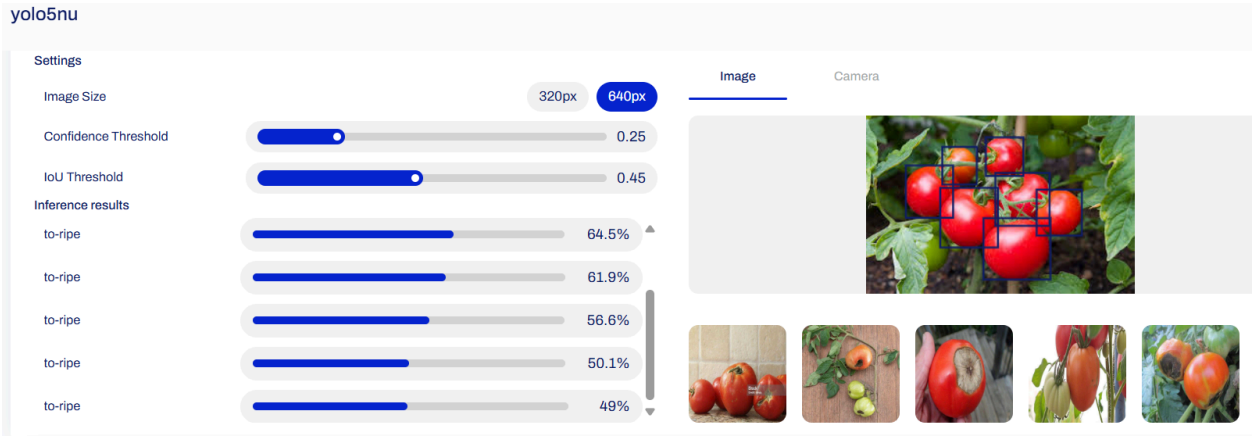


Figure 21. Run detect YOLOv5nu

We can see that the model identifies the object as 100% ripe tomato, and the boxes are very accurately surrounding the tomato.

The second will be model YOLOv8n

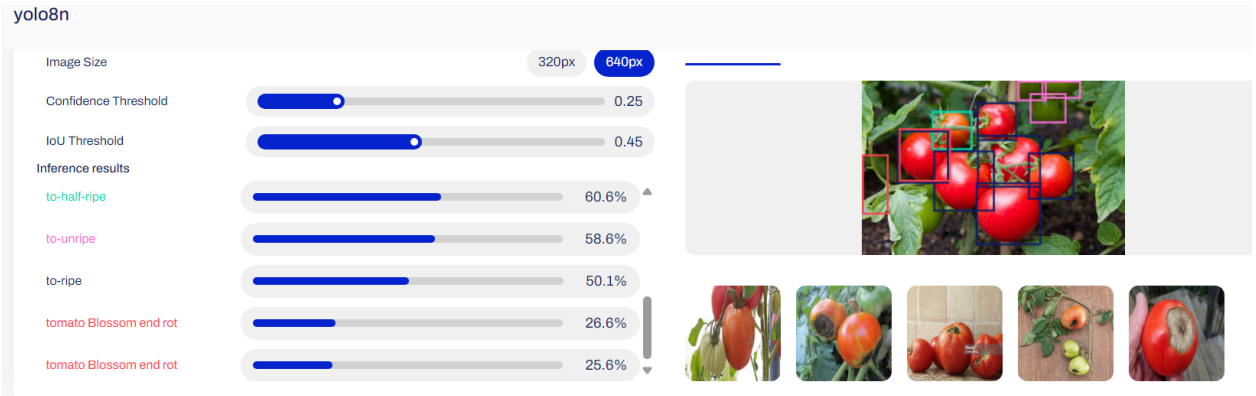


Figure 22. Run detect YOLOv8u

We can see that the model identifies more objects, identifies more distant objects but identifies the wrong object, while there is no tomato blossom end rot, and identifies error boxes and duplicate boxes.

Finally YOLOv11n

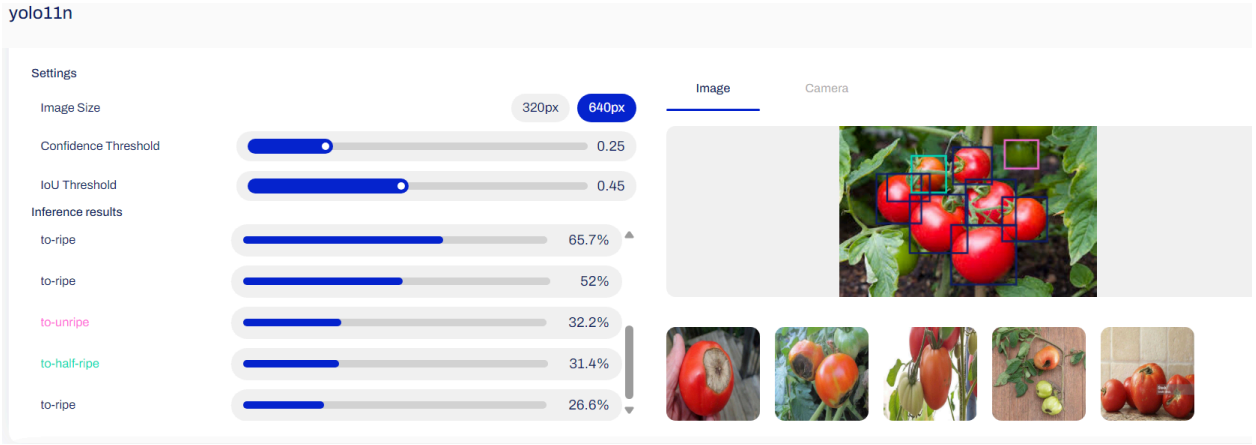


Figure 23. Run detect YOLOv11n

We can see that this is the model that gives the best results compared to the previous two models. The model can identify far and near objects, not misidentify objects, and correctly identify nearly ripe tomatoes, so this is the model that gives the most accurate and most stable practical running results.

Yolo small model comparison results on custom dataset

Models	**Size**(pixels)	**Map of Val 50-95**	**Params** (M)	**GFLOPs** (B)
YOLOv5su	640	67.3	9.1	24
YOLOv8s	640	68.1	11.2	28.6
YOLOv9s	640	66.2	7.2	27.4
YOLOv10s	640	65	7.2	24.8
YOLOv11s	640	67.3	9.4	21.3

Table 9. Yolo small model comparison results on custom dataset

We experimented and saw that the YOLOv11 model has the best overall metrics against other models with high inference speed at 2.4ms and the F1-score at 0.81. So we will use this model for deployment later.

Deployment

To introduce this project to everyone and simplify the operation process, we have an export file train on the website roboflow and ultralytics hub so that end users can understand and experience our project in the simplest and fastest way. Through this, give us feedback and suggestions so that we can fix and develop the project further.

1 Access Ultralytics HUB:

- Log in to your Ultralytics HUB account at Ultralytics HUB.

2. Upload Your Dataset:

- Navigate to the 'Datasets' section.
- Click 'Upload Dataset' and select your dataset files.
- Ensure your dataset is properly formatted for YOLO training.

3. Train Your Model:

- After uploading your dataset, go to the 'Models' section.
- Click 'Create Model' and configure your training parameters.
- Initiate the training process.
- Monitor training progress and metrics in real-time.

4. Deploy Your Model:

- Once training is complete, select your trained model.
- Click 'Deploy' to access deployment options.
- Choose your preferred deployment method:
- Shared Inference API: Ideal for quick testing and small-scale applications.
- Dedicated Inference API: Suitable for production environments requiring higher

performance and scalability.

- For Dedicated Inference API users, click the 'Start Endpoint' button to initiate the endpoint. Once active, HUB will provide a unique URL for your inference tasks.

5. Integrate and Test:

- Use the provided API endpoint to integrate the model into your application.
- Test the deployed model to ensure it meets your performance and accuracy requirements.

6. Export file

- Preview on mobile ultralytics hub app for running real time

6. DISCUSSION

This study confirms the effectiveness of advanced machine learning models, especially YOLOS, in automating fruit ripeness and disease detection to reduce labor costs in agriculture. YOLOS's high precision enables accurate ripeness classification and early disease detection, which are critical for timely interventions and quality maintenance. Integrating YOLOS with CNNs enhanced robustness, adapting well to varied conditions in real-time, field applications.

Our results support a scalable, non-destructive approach for agricultural management, preserving produce quality while optimizing harvest timing and reducing post-harvest losses. Though performance may vary under extreme conditions or with uncommon fruit varieties, further training on diverse datasets could address these limitations. Future directions could involve IoT integration for broader, continuous crop monitoring, enhancing precision agriculture.

Overall, this model combination offers a promising, cost-effective solution for sustainable crop management, improving both efficiency and quality across the supply chain.

7. CONCLUSION

From our study, we developed an automated system for fruit ripeness and disease detection using advanced deep learning techniques, combining robust data preprocessing and the evaluation of multiple models to achieve optimal performance.

- **Model Evaluation:**

  Multiple YOLO models (5, 8, 9, 10, and 11) and alternative models like RMET were tested for accuracy, inference speed, and adaptability. YOLOv11 emerged as the most suitable choice, achieving a mean Average Precision (mAP50) of 0.847, an F1-score of 0.81, and the fastest inference speed at 2.4ms.

- **Data Preprocessing:**

  Transformed segmentation labels into bounding box format for better compatibility. Removed duplicate images based on pixel similarity to enhance data quality.
  Balanced class distributions using a custom formula to ensure uniform representation of green, half-ripe, and ripe fruit classes.
  Split the dataset into training, validation, and testing subsets with an 8:1:1 ratio to optimize model training and evaluation.

- **Real-World Applicability:**

  YOLOv11 demonstrated robustness in handling diverse conditions, including varying fruit appearances and environmental factors, making it suitable for real-time deployment in agricultural settings. The automated detection system reduces reliance on manual labor, minimizes errors, and supports sustainable agricultural practices. While working on this project, we realized that a real-time system is out of scope and unnecessary. We are working on a web-based and app deployment.

CONFLICTS OF INTEREST: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

8. REFERENCES

[1] Sa I., Ge Z., Dayoub F., Upcroft B., Perez T., and McCool C., Deepfruits: a fruit detection system using deep neural networks, *Sensors*. (2016) 16, no. 8, 2-s2.0-84982682350. https://doi.org/10.3390/s16081222

[2] Chen S. W., Shivakumar S. S., Dcunha S., Das J., Okon E., Qu C., Taylor C. J., and Kumar V., Counting apples and oranges with deep learning: a data-driven approach, IEEE Robotics and Automation Letters. (2017) 2, no. 2, 781–788, 2-s2.0-85028676918. https://doi.org/10.1109/LRA.2017.2651944,

[3] Feng Xiao, Haibin Wang, Yueqin Xu, and Ruiqing Zhang. Fruit Detection and Recognition Based on Deep Learning for Automatic Harvesting: An Overview and Review. Agronomy 2023, 13(6), 1625. https://doi.org/10.3390/agronomy13061625

[4] Sarika Bobde, Sarthak Jaiswal, Pradnya Kulkarni, Omkar Patil, Pranav Khode, Rishabh Jha: Fruit Quality Recognition using Deep Learning Algorithm, 21 December 2021. https://ieeexplore.ieee.org/document/9645793

[5] Prasad, K.; Jacob, S.; Siddiqui, M.W. Fruit maturity, harvesting, and quality standards. In Preharvest Modulation of Postharvest Fruit and Vegetable Quality; Elsevier: Amsterdam, The Netherlands, 2018. https://www.sciencedirect.com/science/article/abs/pii/B9780128098073000020

[6] Alistair Mowat, Ray Collins. Consumer behavior and fruit quality: supply chain management in an emerging industry, 1 March 2000. https://www.emerald.com/insight/content/doi/10.1108/13598540010312963/full/html

[7] Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587. https://ieeexplore.ieee.org/document/6909475

[8] Girshick, R. Fast R-CNN. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1440–1448. https://ieeexplore.ieee.org/document/7410526

[9] Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Trans. Pattern Anal. Mach. Intell. 2017, 39, 1137–1149. https://ieeexplore.ieee.org/document/7485869

[10] He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2980–2988.

https://ieeexplore.ieee.org/document/8237584

[11] Bargoti, S.; Underwood, J.P. Image Segmentation for Fruit Detection and Yield Estimation in Apple Orchards. J. Field Robot. 2017, 34, 1039–1060.
https://arxiv.org/abs/1610.08120

[12] Häni, N.; Roy, P.; Isler, V. A Comparative Study of Fruit Detection and Counting Methods for Yield Mapping in Apple Orchards, 2020.
https://arxiv.org/abs/1810.09499

[13] Sa, I.; Ge, Z.; Dayoub, F.; Upcroft, B.; Perez, T.; Mccool, C. DeepFruits: A Fruit Detection System Using Deep Neural Networks. Sensors 2016, 16, 1222.
https://doi.org/10.3390/s16081222

[14] Linjuan Ma; Fuquan Zhang; Lin Xu. Fruit Detection Using Faster R-CNN Based On Deep Network, January 2019.
https://www.researchgate.net/publication/329335219

[15] Longsheng Fu; Yali Feng; Yaqoob Majeed; Xin Zhang; Jing Zhang; Manoj Karkee; Qin Zhang. Kiwifruit detection in field images using Faster R-CNN with ZFNet, 2018.
https://www.sciencedirect.com/science/article/pii/S2405896318311753

[16] Xiaochun Mai; Hong Zhang; Xiao Jia; Max Q.-H. Meng. Faster R-CNN With Classifier Fusion for Automatic Detection of Small Fruits, 30 January 2020.
https://ieeexplore.ieee.org/document/8976303

[17] Liu, X.; Zhao, D.; Jia, W.; Ji, W.; Ruan, C.; Sun, Y. Cucumber Fruits Detection in Greenhouses Based on Instance Segmentation. IEEE Access 2019, 7, 139635–139642.

[18] Weikuan Jia; Yuyu Tian; Rong Luo; Zhonghua Zhang; Jian Lian; Yuanjie Zheng. Detection and segmentation of overlapped fruits based on optimized mask R-CNN application in apple harvesting robot, May 2020.
https://www.sciencedirect.com/science/article/abs/pii/S0168169919326274

[19] HongJun Wang; Qisong Mou; Youjun Yue; Hui Zhao. Research on Detection Technology of Various Fruit Disease Spots Based on Mask R-CNN, 26 October 2020.
https://ieeexplore.ieee.org/document/9233575

[20] P. Ganesh; K. Volle; T.F. Burks; S.S. Mehta. Deep Orange: Mask R-CNN based Orange Detection and Segmentation, 2019.
https://www.sciencedirect.com/science/article/pii/S2405896319324152

[21] Koirala, A.; Walsh, K.B.; Wang, Z.X.; McCarthy, C. Deep learning for real-time fruit detection and orchard fruit load estimation: Benchmarking of 'MangoYOLO'. Precis. Agric. 2019, 20, 1107–1135.
https://www.researchgate.net/publication/331423658

[22] Ao Wang; Hui Chen; Lihao Liu; Kai Chen; Zijia Lin; Jungong Han; Guiguang Ding. YOLOv10: Real-Time End-to-End Object Detection, 30 October 2024.

https://arxiv.org/abs/2405.14458v2

[23] Jocher Glenn, Yolov8, 2023.
https://github.com/ultralytics/ultralytics/tree/main

[24] Chien-Yao Wang; I-Hau Yeh; Hong-Yuan Mark Liao. Yolov9: Learning what you want to learn using programmable gradient information, 2024.
https://arxiv.org/abs/2402.13616

[25] Chuyi Li; Lulu Li; Yifei Geng; Hongliang Jiang; Meng Cheng; Bo Zhang; Zaidan Ke; Xiaoming Xu; Xiangxiang Chu. Yolov6 v3.0: A full-scale reloading, 2023.
https://arxiv.org/abs/2301.05586

[26] Shangliang Xu; Xinxin Wang; Wenyu Lv; Qinyao Chang; Cheng Cui; Kaipeng Deng; Guanzhong Wang; Qingqing Dang; Shengyu Wei; Yuning Du; et al. Pp-yoloe: An evolved version of yolo, 2022.
https://arxiv.org/abs/2203.16250

[27] LaboroTomato: Instance Segmentation Dataset is under CC BY-NC-SA 4.0 license.
https://github.com/laboroai/LaboroTomato#licence