

UNITEC INSTITUTE OF TECHNOLOGY

MASTER THESIS

---

# Apple Detection Using Filters Under Varying Lighting Conditions

---

*Author: Kiran Sonala*

*Supervisor: Dr Lei Song*

*Co-supervisor: Sachin Sen*

*A thesis submitted in partial fulfillment of the requirements for the  
degree of Master of Applied Technologies  
in the*

School of Computing, Electrical & Applied Technology

21.02.2025

## Abstract

The advancement of intelligent farming and agricultural automation has led to the development of robotic methods for efficient apple harvesting in recent years. However, because of complicated backgrounds and fluctuating illumination, it is still challenging to identify apples in genuine orchard settings. Issues like shifting lighting intensities, shadows, and image noise complicate detection, impacting the reliability of these automated systems in real-world applications. Identifying apples under different lighting conditions, such as sunlight, shadow, darkness, and blur, presents a significant obstacle in automating apple harvesting. Traditional object detection algorithms struggle with these unpredictable lighting variations, leading to decreased accuracy. There is a need for a system that can dynamically adapt to these environmental challenges and improve apple detection reliability in varying orchard conditions.

This research investigates the integration of Support Vector Machine classification and image preprocessing techniques as a preprocessing step for YOLOv9-based object detection. By classifying images based on their lighting conditions and applying appropriate filters, the research aims to enhance image clarity before detection. The study further explores various filters, such as contrast adjustment, gamma correction, and histogram equalization, to determine their effectiveness in improving detection under specific lighting scenarios. The proposed solution involves a multi-phase approach where an SVM classifier identifies the lighting condition of each image, and the corresponding filter is applied to optimize image quality. YOLOv9, an efficient object detection model, is then used to detect apples in the pre-processed images. This adaptive system enables real-time preprocessing tailored to lighting variations, ensuring that each image is processed optimally for improved feature extraction and detection accuracy.

The experimental results demonstrate that the integrated SVM-Filter-YOLOv9 pipeline significantly enhances detection rates, particularly in low-light and high-contrast environments. By employing adaptive gamma correction, contrast stretching, and colour enhancement filters, the system achieved an overall improvement of approximately 23% in apple detection accuracy compared to non-filtered images. Notably, individual filter performance was especially effective for dark conditions, showing a 47% improvement, and for sunlight conditions, with a 37% increase. This study provides a scalable and reliable framework for automated fruit detection systems, positioning it as a valuable tool for future robotic harvesting technologies.

## Acknowledgements

I want to express my deepest gratitude to everyone who supported and guided me throughout the development of this thesis.

I sincerely thank my supervisor, Dr. Lei Song, for your expertise in conceptualising the research and proposing methods and technologies related to object detection and machine learning. Your constant reviews, insightful suggestions, and guidance have been invaluable in shaping the direction and focus of this work.

I am also profoundly grateful to my co-supervisor, Sachin Sen, for your essential guidance in research writing and for helping me structure and present my ideas effectively. Your mentorship has dramatically enriched the quality and relevance of this thesis.

I am deeply thankful to my family for their unwavering love, care, and sacrifices, which have been a source of strength and motivation throughout this journey.

Finally, I want to express my gratitude to my Mom and Dad. I miss you every day.

Kiran Sonala

## List of Figures

FIGURE 1 A MULTI-ARM HARVESTING ROBOT .....	5
FIGURE 2 SINGLE SHORT DETECTOR ARCHITECTURE .....	17
FIGURE 3 FASTER RCNN ARCHITECTURE.....	20
FIGURE 4 IMAGE LIGHTING CONDITION DETECTION USING SVM .....	34
FIGURE 5 DATA POINTS FROM IMAGE FEATURE VECTORS .....	37
FIGURE 6 IMAGE PRE-PROCESSING FILTER APPLICATION ON THE ORIGINAL IMAGE. ....	40
FIGURE 7 IMAGE BEFORE AND AFTER THE SUNLIGHT FILTER .....	44
FIGURE 8 IMAGE BEFORE AND AFTER THE SUNLIGHT FILTER .....	44
FIGURE 9 IMAGE BEFORE AND AFTER SHADOW FILTER .....	45
FIGURE 10 IMAGE BEFORE AND AFTER SHADOW FILTER .....	45
FIGURE 11 IMAGE BEFORE FILTER AND AFTER DARK FILTER.....	46
FIGURE 12 IMAGE BEFORE FILTER AND AFTER DARK FILTER .....	47
FIGURE 13 IMAGE BEFORE AND AFTER BLUR FILTER .....	48
FIGURE 14 IMAGE BEFORE AND AFTER BLUR FILTER .....	48
FIGURE 15 APPLE DETECTION USING YOLOV9 .....	49
FIGURE 16 YOLOV9 ARCHITECTURE WITH BACKBONE, NECK AND HEAD .....	51
FIGURE 17 IMAGES TESTED WITH ONLY YOLOV9 FOR APPLE DETECTION .....	55
FIGURE 18 IMAGES TESTED WITH FILTERS AND YOLOV9 FOR APPLE DETECTION .....	56
FIGURE 19 IMAGES PASSED TO TRAINED SVM, FILTER SELECTION AND APPLE DETECTION USING YOLOV9 .....	57
FIGURE 20 SAMPLE IMAGES FROM THE DATASET.....	60
FIGURE 21 FOLDER STRUCTURE.....	61
FIGURE 22 SAMPLE IMAGES FROM THE BLUR FOLDER.....	61
FIGURE 23 SAMPLE IMAGES FROM THE DARK FOLDER .....	61
FIGURE 24 SAMPLE IMAGES FROM THE SHADOW FOLDER.....	62
FIGURE 25 SAMPLE IMAGES FROM THE SUNLIGHT FOLDER .....	62
FIGURE 26 CONVERTING INPUT IMAGE INTO A ONE-DIMENSIONAL FEATURE ARRAY.....	63
FIGURE 27 APPLE DETECTION RESULTS BEFORE AND AFTER DARK FILTER .....	72
FIGURE 28 APPLE DETECTION RESULTS BEFORE AND AFTER DARK FILTER .....	72
FIGURE 29 APPLE DETECTION RESULTS BEFORE AND AFTER DARK FILTER .....	73
FIGURE 30 APPLE DETECTION RESULTS BEFORE AND AFTER SHADOW FILTER.....	73
FIGURE 31 APPLE DETECTION RESULTS BEFORE AND AFTER SHADOW FILTER.....	74
FIGURE 32 APPLE DETECTION RESULTS BEFORE AND AFTER BLUR FILTER .....	74
FIGURE 33 APPLE DETECTION RESULTS BEFORE AND AFTER BLUR FILTER .....	74
FIGURE 34 APPLE DETECTION RESULTS BEFORE AND AFTER SUNLIGHT FILTER.....	75
FIGURE 35 APPLE DETECTION RESULTS BEFORE AND AFTER SUNLIGHT FILTER.....	75

## List of Tables

TABLE 1: ML TECHNIQUES FOR FRUIT DETECTION UNDER LIGHTING VARIATIONS .....	13
TABLE 2 SOFTWARE COMPONENTS USED IN IMPLEMENTATION.....	58
TABLE 3 CONFUSION MATRIX FOR FOUR LIGHTING CONDITIONS IN APPLE DETECTION USING RBF KERNEL-SVM. .....	66
TABLE 4 CONFUSION MATRIX FOR FOUR LIGHTING CONDITIONS IN APPLE DETECTION USING LINEAR KERNEL- SVM.....	67
TABLE 5 CONFUSION MATRIX FOR FOUR LIGHTING CONDITIONS IN APPLE DETECTION USING RBF KERNEL-SVM. .....	69
TABLE 6 CONFUSION MATRIX FOR FOUR LIGHTING CONDITIONS IN APPLE DETECTION USING LINEAR KERNEL- SVM.....	70
TABLE 7 PERFORMANCE OF FILTERS ON APPLE DETECTION - DETECTION ACCURACY AND NUMBER OF APPLES DETECTED IMPROVEMENT.....	76
TABLE 8 INTEGRATED RESULTS – SVM, FILTERS AND YOLOV9.....	78
TABLE 9 INTEGRATED RESULTS WITH TEST DATA – SVM, FILTERS AND YOLOV9 – APPLE DETECTION ACCURACY AND NUMBER OF APPLES DETECTED IMPROVEMENTS .....	79

## List of Abbreviations

YOLO	You Only Look Once
CNN	Convolutional Neural Network
SVM	Support Vector Machine
RGBD	Red, Green, Blue, Depth
RBF	Radial Basis Function
DCNN	Deep Convolutional Neural Network
AI	Artificial Intelligence
ML	Machine Learning
HDR	High Dynamic Range
SSD	Single Shot MultiBox Detector
RoI	Region of Interest
PGI	Programmable Gradient Information
RPN	Region Proposal Network
BiFPN	Bidirectional Feature Pyramid Network
KNN	K-Nearest Neighbors
ANN	Artificial Neural Network
CA	Coordinate Attention
GELAN	Generalized Efficient Layer Aggregation Network
Fast R-CNN	Fast Region-Based Convolutional Neural Network
Faster R-CNN	Faster Region-Based Convolutional Neural Network
CLAHE	Contrast Limited Adaptive Histogram Equalization
DK_YOLOv5	Dark Knowledge YOLO version 5
LE-YOLO	Lightweight Enhanced YOLO

MARL	Multi-Agent Reinforcement Learning
CHT	Circular Hough Transform
NLM	Non-Local Means
HSV	Hue, Saturation, Value
YIQ	Luminance and Chrominance
BGR	Blue, Green, Red

## Table of Contents

<b>Abstract</b> .....	ii
<b>Acknowledgements</b> .....	iii
<b>List of Figures</b> .....	iv
<b>List of Tables</b> .....	v
<b>List of Abbreviations</b> .....	vi
<b>Chapter 1 Introduction</b> .....	3
<b>1.1 Background</b> .....	4
<b>1.2 Problem Statement</b> .....	6
<b>1.3 Research Objectives</b> .....	6
<b>1.4 Research Question</b> .....	7
<b>1.5 Contributions</b> .....	7
<b>1.6 Thesis Structure</b> .....	8
<b>Chapter 2 Literature Review</b> .....	9
<b>2.1 Fruit Detection</b> .....	9
<b>2.1.1 Fruit detection process</b> .....	14
<b>2.2 Apple Detection Techniques</b> .....	15
<b>2.2.1 Single-Shot MultiBox Detector</b> .....	16
<b>2.2.2 Faster R-CNN</b> .....	19
<b>2.2.3 You Only Look Once (YOLO)</b> .....	23
<b>2.3 Lighting conditions</b> .....	28
<b>2.4 Gaps in the current research</b> .....	29
<b>2.4.1 Identification of Lighting Conditions</b> .....	29
<b>2.4.2 Advantages of Using SVM Over Other Machine Learning Algorithms</b> .....	29
<b>2.4.3 Applying the Right Image Preprocessing Filter Based on Lighting Condition</b> .....	30
<b>2.4.4 YOLOv9 Object Detection: Pre and Post-Filter Results</b> .....	31
<b>Chapter 3 Methodology</b> .....	32
<b>3.1 Image Lighting or Noise Condition Detection</b> .....	34
<b>3.2 Filter Selection Based on Image Lighting or Noise Conditions</b> .....	39
<b>3.3 Apple Detection</b> .....	49
<b>Chapter 4 Experiment Setup and Testing</b> .....	54
<b>4.1 Experiment Setup</b> .....	54
<b>4.2 System Environment Setup</b> .....	57

4.2.1	Software Environment .....	57
4.2.2	Hardware Specifications .....	58
4.3	Data Collection and Processing .....	59
4.3.1	Dataset Description .....	59
4.3.2	Dataset Categorization (Manual sorting into sunlight, shadow, dark and blur) ....	60
4.3.3	Dataset Split and Feature Extraction (Train, Validation, Test) .....	62
4.4	SVM configuration and parameters .....	63
4.4.1	Kernel Selection .....	63
4.4.2	Hyperparameter Tuning .....	64
4.4.3	Regularization .....	64
Chapter 5 Results and Discussion .....		65
5.1	SVM Validation Classification Results ( Lighting Condition Detection ) .....	65
5.2	SVM Testing Classification Results (Lighting Condition Detection) .....	68
5.3	Performance Analysis of Filters .....	71
5.3.1	Apple Detection results with and without filter .....	71
5.3.2	Filter Performance Results .....	75
5.4	SVM classification and YOLOv9 apple detection results with Filters .....	78
5.5	Findings .....	80
Chapter 6 Conclusion and Future Research .....		84
6.1	Future Works .....	85
Appendix .....		87
Bibliography .....		92

## Chapter 1 Introduction

Human existence and the global economy depend heavily on agriculture. However, it faces enormous obstacles today due to population expansion and climate change. It is now more important than ever to increase agricultural production's sustainability and efficiency. Automation technologies have become a vital remedy and a key component of agricultural modernization. China produced more than 49 per cent of the world's apples, ranking first. An autonomous and unmanned harvest has been a pressing solution to the fruit industry's bottleneck to save labour expenses (Snyder & Ni, 2017). Considering this, the number of robotic harvesters has increased over the past few decades and is considered a promising technology. Harvesting robots in orchards has garnered much interest recently due to advancements in computer and sensor technologies, and significant strides have been made in fruit detection and localization (Gongal et al., 2015). In previous years, harvesting robots for apples (Kootstra et al., 2021), tomatoes (Rong et al., 2022), oranges (Zhou et al., 2022), and pineapples (Kurbah et al., 2022) were developed.

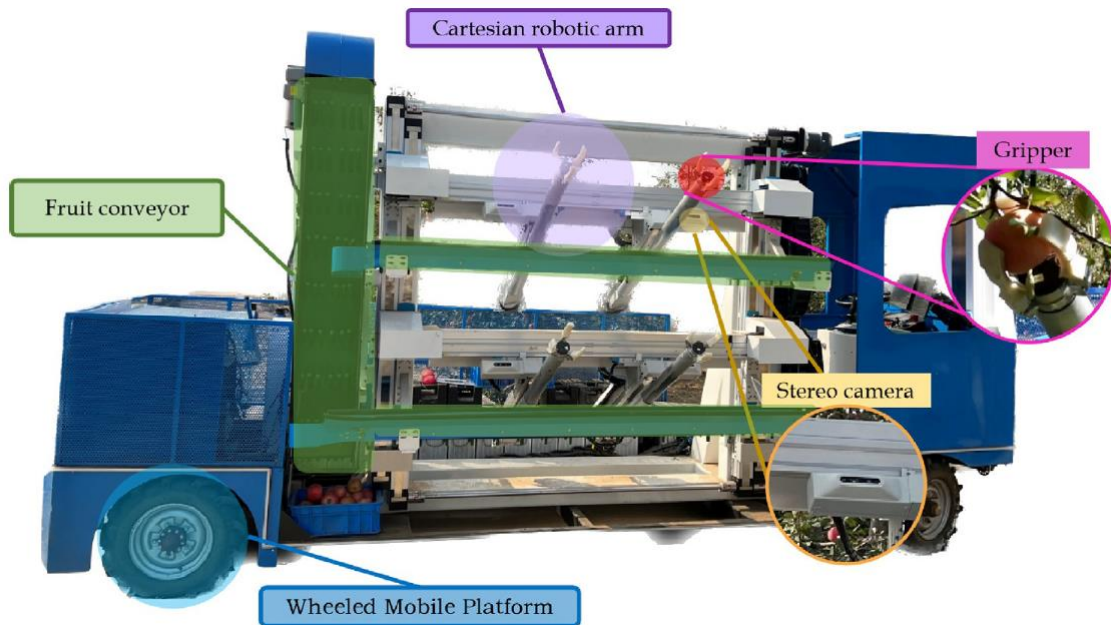
The picking robots have produced apples through dwarfing and dense planting, which orchard owners have increasingly embraced because of their exceptional advantages in facilitating mechanised processes. Widespread harvesting during the picking season and intense ripening are two of the most notable aspects of apple cultivation. The main issue with robots' practical use is their overall operating efficiency. Apple detection in natural orchards is still difficult because of several visual and environmental elements, even with advancements in computer vision technologies and robotic harvesting systems. Unpredictable weather variations modify lighting circumstances, such as abrupt cloud cover or changes in sunshine intensity, which can decrease image clarity and detection accuracy. This has a substantial impact on apple detection. Furthermore, the detecting process is made more difficult by the different lighting circumstances introduced by the different times of day, morning, noon, evening, or night, such as intense glare, deep shadows, or poor visibility. Overlapping leaves, branches, and even other apples can cause occlusions, which block parts of the target fruit from the camera's view. Furthermore, factors like varying apple sizes, colour changes caused by ripening stages, debris, or other background items can all create noise in detection. These challenges require robust preprocessing techniques and adaptable algorithms that can handle a variety of illumination, occlusion, and environmental conditions to ensure efficient and accurate apple harvesting.

## 1.1 Background

Artificial Intelligence (AI) is increasingly valuable in agriculture, especially for plant disease identification and fruit detection within orchards. Machine Learning (ML) techniques, like Convolutional Neural Networks (CNN), enable complex image analysis, recognizing spatial hierarchies that enhance fruit detection accuracy even under varied orchard conditions (Javaid et al., 2023). Ensemble methods, which combine multiple models, have been shown to improve classification reliability by addressing environmental challenges such as inconsistent lighting, occlusions, and fruit size or colour variations. Data augmentation techniques, including image rotation and scaling, further enhance model robustness, allowing them to generalise effectively across different settings (Kaur et al., 2021). Together, these advancements in AI offer practical solutions for precision agriculture, where early disease detection and accurate fruit identification optimise harvest efficiency, minimise crop loss, and support sustainable farming practices. As these AI systems are integrated with other data sources, such as environmental sensors, they hold immense potential to transform agriculture, making it resilient to diverse conditions and adaptable to future challenges.

The study (Li et al., 2023) developed a fruit recognition and localisation algorithm based on multitasking deep learning, which was introduced to enhance the accuracy of apple positioning. Additionally, a method that combines multiple perspectives is proposed to acquire a comprehensive global fruit map. Fruit detection and localisation are addressed by:

- A fruit detection and localisation algorithm that aims to increase hidden fruits' localisation accuracy and recognition rate by utilizing several stereo cameras and a multi-task deep convolutional neural network.
- A task planning method based on a multi-agent reinforcement learning model that optimizes the time needed to pick various fruits with several arms to reduce the average cycle time and boost the overall operating efficiency of the robot.
- A highly integrated robotic system with four arms for picking apples in orchards. Field harvesting tests under various apple tree growth conditions revealed that the robot's average cycle time ranged from 5.8% to 6.7% and its harvest success rate from 71.28% to 80.45%.



*Figure 1 A multi-arm harvesting robot*

The multi-arm robotic apple harvesting system combines cutting-edge vision, task planning, and control technology, as depicted in Figure 1. The perception phase uses RGBD cameras and a deep convolutional neural network (DCNN) to precisely identify and locate fruit in the orchard. This method adjusts to problems like branch occlusions and changing sunlight by dividing fruit pieces and including border boxes. Accurate Apple positioning is possible via frustum-based localisation, incorporating localised data into a global frame.

The system uses a multi-agent reinforcement learning (MARL) framework to control the cooperative movements of the four robotic arms during the task planning stage. This paradigm maximises job allocation and sequencing by considering fruit distribution, overlapping work areas, and arm movements. Fruit picking is made efficient and flourishing by the system's ability to support sequential and parallel operations while reducing idle time and potential conflicts.

Each robotic arm, which has specially designed grippers to enable accurate harvesting, operates in unison to handle control and execution. Approach, extension, grab, retraction, and placement are the steps in a specific cycle that the arms go through. The robot can move fluidly between tree rows on the wheeled mobile platform, guaranteeing continuous operation. The multi-arm

robotic system maintains fruit quality and adapts to changing orchard circumstances by combining these operations to produce high apple harvesting success rates.

Apple detection in outdoor environments is a multifaceted challenge impacted by numerous factors, including the complex structure of plants, natural variations in fruit shape, size, and colour, and unpredictable environmental conditions. The dense foliage and interwoven branches often result in occlusion, concealing fruit from detection systems and complicating accurate classification. Additionally, the diversity in fruit appearance due to natural growth variations, seasonal influences, and varietal differences necessitates algorithms that can adapt to various visual characteristics. These challenges are compounded by fluctuating lighting conditions, with intense sunlight, shadows, and varying weather patterns, such as rain, fog, or snow, which introduce image noise and obscure essential details required for accurate detection. Addressing these challenges demands a robust, adaptable detection system that integrates advanced computer vision algorithms with environmental resilience, capable of high accuracy under variable conditions and resilient to equipment wear, positioning the field of apple detection at the intersection of technological innovation and agricultural efficiency.

## **1.2 Problem Statement**

Detecting apples in natural orchard environments presents significant challenges due to varying lighting conditions during the day at different times, which are further influenced by changing weather patterns. These lighting conditions can range from direct sunlight, shadowed areas, and low-light scenarios to motion blur, each affecting image quality and detection accuracy. Overexposure to sunlight can wash out critical features, while shadows and low light obscure details, making it difficult for traditional computer vision methods to detect apples reliably. To address these challenges, robust preprocessing techniques are necessary to enhance image clarity before detection, and accurately identifying lighting conditions or noise in the captured images becomes crucial for effective apple detection. This will improve individual apple detection accuracy, increase the number of apple detections, and reduce false detections.

## **1.3 Research Objectives**

The primary objective of this research is to develop a robust apple detection system capable of operating effectively under diverse lighting conditions and noise levels. This involves integrating image enhancement filters tailored to specific lighting variations with a YOLOv9 object detection model. The study aims to improve detection accuracy by preprocessing images based on their lighting conditions or noise characteristics, as identified by a Support Vector

Machine (SVM) classifier. This approach seeks to make the detection model more adaptable, reliable, and practical for real-world applications.

#### 1.4 Research Question

- How do different lighting conditions affect apple detection in orchard images?
- How can the lighting conditions or noise of the captured image be identified?
- What filters need to be applied to the image to get the enhanced image for object detection?
- Which image filters are most effective in enhancing apple detection under varying lighting conditions or noise levels in the image by using YOLOv9?

#### 1.5 Contributions

This research offers several significant contributions to agricultural automation and computer vision, particularly in improving apple detection under challenging lighting conditions in natural orchard environments. First, developing an automated lighting condition detection system based on a Support Vector Machine (SVM) classifier represents a novel approach to adaptive image preprocessing for agricultural tasks. The SVM model, trained to recognize specific lighting conditions such as sunlight, shadow, blur, and darkness, allows the system to dynamically assess and categorize images based on their lighting environment with high accuracy. This classification process is essential for optimizing the preprocessing pipeline, ensuring that each image undergoes the most appropriate filtering technique to mitigate the adverse effects of varying lighting on object detection. The system lays a foundation for more sophisticated, condition-aware preprocessing in agricultural imaging tasks by reliably identifying and adapting to lighting scenarios.

This study introduces an integrated filtering mechanism that significantly enhances image quality before object detection. By applying custom filters tailored to specific lighting conditions, the system refines image characteristics such as brightness, contrast, and sharpness, thereby improving the visibility of essential features and structures within each image. This preprocessing step is critical, enabling the YOLOv9 object detection model to perform more effectively by receiving images optimized for accurate feature extraction. For instance, contrast adjustments can reduce glare and restore detail in overexposed images due to direct sunlight. In contrast, in low-light conditions, brightness adjustments make hidden features more accessible to the detection model. This adaptive filtering strategy optimises image clarity and

ensures that the YOLOv9 model operates at its full potential, resulting in more reliable and consistent apple detection across various environmental conditions.

Integrating SVM-based classification, adaptive filtering, and YOLOv9 detection into a unified detection pipeline represents an innovative approach to addressing the limitations of traditional object detection models in uncontrolled outdoor settings. Conventional models often struggle with the unpredictability of natural lighting, leading to reduced accuracy in real-world agricultural applications. By leveraging a combined approach, this system achieves a robust detection pipeline that dynamically adapts to environmental variations, thus enhancing detection rates and reducing false negatives. The comprehensive methodology presented in this study addresses existing challenges in fruit detection. It offers a scalable framework that could be applied to other types of agricultural automation, making it a valuable contribution to precision agriculture technologies. This integrated pipeline is a practical and effective solution for real-time applications, facilitating the development of autonomous systems capable of reliable object detection under diverse and often unpredictable conditions in outdoor agricultural environments.

## **1.6 Thesis Structure**

The thesis is structured as follows: Chapter 1, Introduction, lays the foundation by outlining the problem, setting clear research objectives, posing key research questions, and summarizing the study's contributions. Chapter 2, Literature Review, explores existing methods for fruit detection and various preprocessing techniques used to handle different lighting conditions in orchard environments. Moving forward, Chapter 3, Methodology, elaborates on the integrated detection pipeline, detailing the roles of SVM classification, filter selection, and the YOLOv9 detection model. Chapter 4, Experimental Setup, describes the data collection processes, image preprocessing, and model training, ensuring a transparent approach to experimentation. Chapter 5, Results and Discussion, presents the detection outcomes, evaluates the performance of various filters, and discusses the effectiveness of the proposed solution under different lighting scenarios. Finally, Chapter 6, Conclusion and Future Work, summarizes the study's essential findings and contributions while suggesting potential avenues for future research and improvement.

## Chapter 2 Literature Review

The literature review provides an overview of significant methods and research advancements related to fruit detection, focusing on apple detection and the challenges posed by various lighting conditions. It first looks at general fruit detection methods for identifying and detecting fruits in agricultural settings. After that, the emphasis switches to apple detection techniques, emphasizing feature extraction and the performance of different methods. The review also examines the effects of different lighting conditions on detection accuracy. It highlights the research gap, particularly in better preprocessing techniques to handle a variety of lighting circumstances in apple detection.

### 2.1 Fruit Detection

Over the coming years, intelligent machines and agricultural robots are expected to increase significantly and spread throughout developed nations. Harvesting speciality crops like apples, oranges, cherries, and pears requires much work, and their profitability is dwindling due to growing expenses and a shortage of trained workers. In Washington State alone, more than 15 billion apples must be handpicked by seasonal labour, with an estimated harvesting cost of \$1150–\$1700 per acre per year (Gallardo et al., 2010). The labour-intensive harvesting process incurs a high cost in fruit production. To reduce labour costs, an autonomous and unmanned harvest has been an urgent need to address the bottleneck of the fruit industry. The techniques have been investigated using different types of sensors, combinations, and image processing techniques. Detection and localisation of fruit are the most fundamental information for machine vision-based harvesters (Tang et al., 2020).

Machine vision systems have become integral to developing robotic fruit harvesting, enabling automated systems to identify and locate fruits in complex environments accurately. These systems utilize advanced image processing techniques to differentiate between fruit and foliage, ensuring precise picking while minimizing damage to the fruit and the plant. (Schertz & Brown, 1968) They first proposed the concept of automated harvesting as an alternative to mechanical harvesting. (Pejsa & Orrock, 1984; Sistler, 1987) Highlighted that citrus fruit and apples hold the most significant potential for fruit detection and robotic harvesting.

K-means clustering is a popular unsupervised classification technique. The input data set is split into clusters based on how far off they are from one another by nature using a learning method called K-means. It shortens the distance that separates each object from the cluster centre overall. Iteratively moving objects across the clusters continues until the sum no longer

minimizes (Shapiro & Stockman, 2001). In order to discover fruits, K-means clustering converts images into feature vectors, which frequently represent attributes like colour, texture, or shape. These feature vectors are clustered, representing a potential fruit group or type. After selecting K random points, the technique matches each data point to the nearest centroid using the Euclidean distance. The centroids are iteratively updated by calculating the mean of the data points within each cluster until convergence. The final clusters in fruit detection aid the segmentation and identification of the various fruit types or parts of the fruits in the image. Preprocessing techniques such as edge detection and colour space conversion can increase the accuracy of clustering.

K-means clustering is simple, scalable, and converges quickly; it is ideal for big datasets in fruit detection. It enables the algorithm to cluster based on various fruit features, such as colour and shape, by providing variable feature selection. Its unsupervised nature is one of its key advantages in situations where labelled datasets are few. Among its many shortcomings, K-means is sensitive to initialization, struggles with non-convex geometries, and performs poorly in varying illumination conditions. It can converge to local minima, suffer from outliers, and require preset cluster sizes, which results in less-than-ideal fruit classification.

The study combines low- and high-level visual clues from multi-modal pictures with K-means clustering (Wachs et al., 2010) to separate apples from complex backgrounds. This method improves the system's accuracy by improving apple detection under challenging situations like shadows and occlusions. The combination of thermal and colour imaging dramatically enhances the clustering process and allows for apple localization in orchard conditions. Apples from both images were separated using K-means clustering based on the 'a' and 'b' channels from the L/a/b colour space (Hunter, 1948). Morphological procedures and CHT were applied next to reduce noise and improve classification accuracy. The accuracy of identifying fruit in colour images was 50.6%, 38.8%, and 53.2% when data from thermal and colour shots was combined.

Bayesian classification (Cheeseman et al., 1988) is a probabilistic classifier based on the Bayes theorem, which grounds statistical interpretation in prior knowledge and probability distributions. The Bayesian classifier rule is based on maximizing posterior probability, minimizing probability error, or minimizing Bayes risk, depending on the prior probability (Duda et al., 2012). An object is classified using Bayesian classification into a group that maximizes the posterior probability. Using prior and class-conditional probabilities, the

algorithm determines the likelihood that a given data point (such as a fruit feature vector) belongs to each class. The process involves determining the likelihood, the probability of the data given a class, the posterior probability, and the prior probability of the initial hypothesis of class frequencies using Bayes' rule. Assigning the class with the highest posterior probability is the decision rule to minimize classification errors or Bayes risk. When previous knowledge is available, this approach performs exceptionally well when the data fits a probabilistic model, such as a Gaussian distribution.

Bayesian classification offers a probabilistic approach that effectively handles ambiguity in fruit recognition. It works best when antecedent knowledge is provided, such as expected class frequencies, making better decisions possible even when insufficient or sparse data exists. The method's adaptability to incorporate a variety of factors, including colour, texture, and shape, allows it to be customised for various fruit identification jobs. Moreover, fruit detection scenarios often entail complicated backdrops and noisy data, which the probabilistic technique it employs helps to handle. However, when data distributions are poorly known or unavailable, it can be challenging to accurately compute prior probabilities and likelihoods, a prerequisite for using Bayesian classifiers. The premise that data matches a probabilistic model may not hold for various fruit detection tasks, particularly in diverse environments or with varying lighting. Moreover, Bayesian classifiers can be computationally costly when dealing with large datasets, especially when estimating probabilities for multiple classes. They are, therefore, less appropriate for real-time detection applications.

Based on colour information, (C. Slaughter & C. Harrell, 1989a) classified oranges using a Bayesian classifier. After calculating the RGB (red, green, and blue) mean and covariance of the fruit and background pixels, prior probability was assumed. Thirteen photographs with varying occlusion degrees, backgrounds (bright, with only leaves, or with a variety of natural backgrounds), and lighting conditions were utilised to evaluate the images. Regarding the classification of orange pixels, the authors reported an accuracy of 75.0%; erroneous positives and false negatives were not disclosed in the study. The primary causes of the inaccuracy in this investigation were the occlusion of fruit, bright backdrop lights, and variable lighting conditions. A Bayesian classifier was also used by (Chinchuluun & Lee WonSuk, 2006) to segment citrus photos based on the 'S' and 'I' components of the Hue, Saturation, Value (HSV) color space and the luminance (Y) and chrominance (IQ) (YIQ) colour space, respectively. Using pictures of fruit collected in the field, the algorithm was assessed. The regression

coefficient between manually and machine-counted fruit was 0.89, although the authors did not divulge the system's absolute accuracy.

An artificial neural network (Wu & Feng, 2018) is a supervised learning system that improves with each training cycle by assiduously absorbing information from its surroundings. In order to compute output depending on input, neurons in a neural net communicate information across different regions of the network through connections between them. Several layers can exist in a neural network, depending on how complicated the system is (Zou Jinming and Han, 2009). The architecture consists of an input layer where the network receives inputs about the fruit's colour, texture, and shape. In order to detect non-linear relationships in the data, these features pass through one or more hidden layers, where neurons carry out weighted summing and activation functions. The output layer, which categorises the input as a specific fruit variety, is the last. Large, annotated photo datasets can be used to train artificial neural networks in fruit recognition, teaching them to distinguish between different fruits using these visual cues. Convolutional layers are a sophisticated technique frequently added to improve the network's capacity to identify fruits in pictures with different lighting conditions, angles, or occlusions. Because of their great flexibility, ANN-based architectures can increase detection accuracy when working with complicated datasets.

ANNs can train complicated, non-linear connections from data; they are an incredibly adaptable tool for fruit recognition. An array of characteristics, including colour, texture, and form, may be incorporated by ANNs, making them perfect for fruit identification under various circumstances, including shifting angles or lighting. Deeper layers' ability to absorb more abstract fruit feature representations due to their layered structure improves fruit identification accuracy through hierarchical feature extraction. Because ANNs perform exceptionally well in image processing tasks, mainly when linked with convolutional layers (as in CNNs), they can also handle extensive collections of fruit photographs with complicated backdrops.

ANN training necessitates a large amount of labelled data and processing power, which can be problematic for applications that need to be deployed quickly or when there is not a lot of data available for fruit identification. Furthermore, ANNs are prone to overfitting, leading to poorer generalisation in real-world situations, especially when dealing with sparse or noisy input. Moreover, optimising an ANN's design can be challenging and time-consuming, including the number of layers and neurons. Lastly, real-time fruit detection may be hampered by the high

computational load of ANN models, mainly when these devices are employed on low-processing systems.

Neural networks were utilised by (Plebe & Grasso, 2001a) to recognise oranges based on colour information. The backpropagation method (Rumelhart & McClelland, 1987) was used to train the network. In this system, errors were propagated backwards from the forward signal by neurons using samples of orange and background pixels from an 800-image batch acquired under various lighting conditions. According to the authors, 87.0% of the oranges in the photos were accurately identified, with 5.0% of false negatives and 15.0% of false positives. The author's analysis revealed that uneven lighting and fruit occlusion were the primary sources of mistakes.

Methods	Accuracy	Limitation	Crops Applied	References
K-means Clustering	53-80	Variable Lighting Conditions	Apple	(Bulanon et al., 2004)(Wachs et al., 2010)
KNN clustering	85-90	Variable Lighting Conditions; Variable Fruit Size	Apple, Banana, Lemon, Peaches, Strawberry	(Linker et al., 2012)(Kurtulmus et al., 2014) (Seng & Mirisae, n.d.)
Bayesian Classifier	75	Variable Lighting Conditions	Citrus, Apple, Peaches	(Kurtulmus et al., 2014)(C. Slaughter & C. Harrell, 1989b)
Artificial Neural Network	61-84	Variable Lighting Conditions; Occlusion	Apple, Citrus, Peaches	(Plebe & Grasso, 2001b)(Regunathan & Lee, 2005)
Support Vector Machine	92-93	Variable Lighting Conditions, Occlusion, Clustering	Apple, Citrus, Peaches	(Ji et al., 2012)(Lü et al., 2014) (J. Wang et al., 2009)

*Table 1: ML Techniques for Fruit Detection Under Lighting Variations*

The table summarises various classification methods used in fruit detection, highlighting their respective sensors, accuracy rates, limitations, crops applied, and references. The methods include K-means clustering, KNN clustering, Bayesian classifiers, artificial neural networks, and support vector machines. The table indicates that SVM and KNN clustering offers the

highest accuracy, ranging between 85% to 93%, while K-means clustering presents the lowest accuracy at 53-80%. The standard limitation across these methods is the challenge posed by variable lighting conditions, significantly impacting accuracy. Additional challenges such as occlusion, variable fruit sizes, and clustering are noted, particularly in methods like ANN and SVM. The methods have been applied to various crops, including apples, bananas, lemons, peaches, and strawberries, demonstrating their versatility across different fruit types. The references provided offer a solid foundation for further exploration of these classification techniques in fruit detection, with notable studies from researchers like Wachs, Bulanon, Kurtulmus, and others.

### **2.1.1 Fruit detection process**

In settings where fruit detection is essential, for proper monitoring and management practices to take place effectively and accurately, start with the vital stage of acquiring data through various advanced means, such as sensors and cameras that capture images of the specific fruits in question using sophisticated technology devices like high-resolution cameras along with multispectral and hyperspectral sensors that are commonly utilised in modern agricultural robotics to collect detailed visual information from orchard environments. The quality of the images obtained at this stage plays a role in determining the overall success of the entire detection process since they are the fundamental basis upon which all subsequent image processing operations are rooted. Quality and sharp images are crucial to guarantee that the detection system has the information to operate effectively while reducing mistakes resulting from inadequate lighting conditions or blurry motion.

After gathering the data comes a step known as image preprocessing. This process involves refining the images to improve quality and minimise distortions or unwanted elements. It includes methods such as filtering to eliminate noise, tweaking contrast levels and converting images into colour schemes to highlight particular details of interest. For example, histogram adjustment can enhance the picture's contrast and highlight the attributes, like the colour and texture of the fruits. Improving image quality in this way helps make the following detection steps work better and with precision.

The procedure then moves on to feature extraction, which seeks to identify the unique characteristics of the fruits in the images after preprocessing. This stage is crucial because it involves determining and assessing crucial traits like colour, shape, and texture necessary to distinguish the fruits from other objects in the image. These features are then supplied into the

classification algorithms after being extracted using edge detection and colour segmentation techniques. The detection system's accuracy depends on the most pertinent information from the extracted image, which is ensured by effective feature extraction.

The retrieved features are then utilised in the classification process, where the system categorises the items in the image, determining which are fruits and which are not. In order to identify the patterns and traits of the fruits based on the features that have been extracted, this stage frequently entails the use of machine learning methods like Support Vector Machines, neural networks, or decision trees. This step correctly categorises the objects in the picture, ensuring that fruits are recognised and set apart from other scene features. The entire efficacy of the fruit identification system is directly impacted by the classification algorithm's precision, making it crucial.

The last phase of the fruit identification process is fruit detection, in which the system locates the fruits precisely in the image or the actual environment. This stage is crucial in applications like robotic harvesting, where precise identification is required to direct the harvesting mechanisms to the appropriate spots. The system can offer the exact coordinates of the fruits based on the categorisation step findings, allowing for accurate and efficient harvesting. By decreasing the possibility of missing fruits or inflicting damage during picking, this phase not only guarantees that the fruits are accurately identified but also maximises the harvesting process. This comprehensive approach to fruit detection, encompassing data acquisition, image preprocessing, feature extraction, classification, and detection, forms the backbone of modern agricultural automation systems.

## **2.2 Apple Detection Techniques**

Detecting apples in their natural environment poses a significant challenge for conventional object identification algorithms due to occlusion, varying lighting conditions, and complex backgrounds. However, deep learning techniques for object detection have made substantial advancements, enabling the automatic extraction of apple attributes such as quantity, pixel location, size, and more from images (Zhao et al., 2019). One key challenge for robots operating in dynamic environments is accurately detecting objects of interest. Traditional methods often need help with occlusions, lighting variations, and complex object shapes. Deep-learning methods have been extensively applied in visual detection tasks. In the realm of object detection, popular algorithms are generally divided into single-stage approaches, such as the YOLO (You Only Look Once) series and SSD (Single Shot Multibox Detector), and two-stage

approaches like RCNN (Region CNN), Fast RCNN, and Faster RCNN. These algorithms demonstrate outstanding performance in well-lit conditions. The most widely used object detection models are thoroughly reviewed, emphasising their advantages, disadvantages, and suitability in diverse real-world situations.

### 2.2.1 Single-Shot MultiBox Detector

The Single-Shot MultiBox Detector (W. Liu et al., 2016) is a widely used object detection model designed for speed and efficiency, making it an attractive option for real-time applications like robotics. Unlike two-stage detectors like Faster R-CNN, SSD uses a single-stage approach, significantly enhancing processing speed while maintaining competitive detection capabilities. This design enables SSD to process images in dynamic environments quickly and efficiently.

#### Technical Specifications

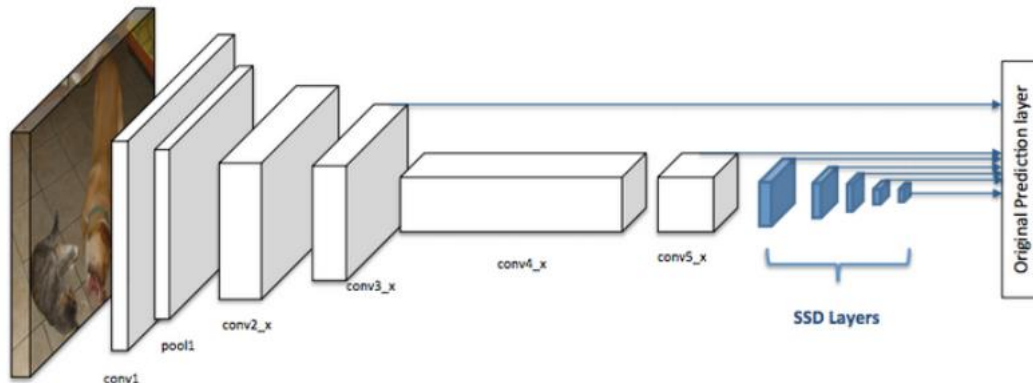
The SSD architecture integrates a pre-trained Convolutional Neural Network (CNN), such as VGG-16 or MobileNet, to extract hierarchical image features. This backbone is complemented by additional convolutional layers specifically optimised for object detection tasks. These layers perform the following operations:

- **Bounding Box Prediction:** Precise localisation of objects through offsets relative to anchor boxes.
- **Class Probability Prediction:** Simultaneous categorisation of detected objects.

One of SSD's standout features is its use of multi-scale feature maps, enabling the detection of objects of varying sizes within a single pass. At each feature map level, SSD applies a collection of default anchor boxes with diverse aspect ratios and scales, ensuring robust detection across a range of object dimensions. The single-stage design eliminates the need for region proposal networks, resulting in faster inference times than two-stage detectors. However, this speed advantage can lead to a trade-off in accuracy, particularly for detecting small or occluded objects, where SSD's reliance on coarser feature maps in deeper layers may limit performance.

**Real-Time Processing:** By eliminating the need for region proposal networks, SSD adopts a single-stage detection pipeline, enabling faster inference speeds compared to two-stage detectors like Faster R-CNN. This advantage makes SSD highly suitable for resource-

constrained environments, such as embedded systems and robotics, where processing speed is critical. However, this speed comes at the cost of lower accuracy for detecting small or highly occluded objects. Figure 2 depicts the SSD architecture.



*Figure 2 Single Shot Detector Architecture*

## Applications and Usage

Research (Liang et al., 2018) demonstrated the potential of SSD for apple detection in orchards, emphasising its ability to operate in real-world scenarios.

SSD proved effective for:

- Detect apples under challenging conditions, such as partial occlusion and complex orchard environments.
- Adapt to objects of varying sizes, thanks to its multi-scale detection capabilities. Furthermore, SSD has been successfully implemented on embedded platforms such as Raspberry Pi and Nvidia Jetson, achieving
- Real-time performance with frame rates of 30 FPS (Xuan et al., 2020).
- Detection accuracy of 83.64% under challenging conditions like overlapping apples and complex backgrounds. (Valdez, 2020) compared SSD with YOLOv3 for defect detection in apples post-harvest. While SSD struggled with small lesions, YOLOv3 showed superior accuracy in identifying apple defects.

## **Advantages and Limitations**

SSD presents several advantages, making it a compelling choice for real-time object detection applications. Its single-stage design enables rapid image processing, making it ideal for scenarios requiring real-time responses. The architecture's scalability allows for task-specific modifications by adjusting the output channels of its detection layers. At the same time, its lightweight and efficient design makes it particularly well-suited for deployment on embedded systems and low-power devices. However, SSD also has notable limitations. The lack of finer feature maps in later layers diminishes its ability to accurately detect small objects, which can be a challenge in applications requiring high precision. SSD's grid-based methodology also struggles in highly cluttered scenes, reducing accuracy in complex backgrounds. Finally, while SSD is optimised for speed, it often falls short of more complex models like Faster R-CNN in precision, underscoring a trade-off between detection speed and accuracy.

## **Recent Developments and Future Directions**

(Ding et al., 2021) introduced an improved SSD model integrating ResNet50 and Receptive Field Blocks (RFB), achieving:

88.7% accuracy in detecting blocked apples.

Improvements of 1.5% over RFB-Net and 3.4% over standard SSD.

(Xuan et al., 2020) developed an SSD-based real-time detection system for low-power devices, achieving robust performance under constrained conditions.

A notable advancement involves the integration of Faster R-CNN with Single SSD to address the limitations of detecting small objects. This combined approach leverages multi-scale feature maps and additional default boxes, achieving 78.68% mean average precision and real-time performance with 89 frames per second on the COCO dataset. These enhancements significantly improve speed and accuracy compared to traditional SSD models (Kumar & Srivastava, 2020).

(L. Wang et al., 2024) They introduced an enhanced SSD model for detecting objects in remote-sensing images under complex backgrounds. By integrating an improved Inception network and a modified Feature Pyramid Network, the model significantly strengthened the feature extraction for small objects and improved multi-scale feature fusion. This approach

demonstrated superior performance, with a mean average precision of 90.2% and improved detection accuracy for small and medium objects.

Further optimisations are needed to detect apples that are significantly obscured. Developing advanced pre-processing and noise-handling algorithms can improve SSD's performance in complex orchard settings. The SSD is a powerful and efficient model for object detection in real-time scenarios. Its ability to balance speed and detection capabilities makes it suitable for agricultural robotics and embedded systems. Despite limitations with small objects and complex scenes, recent advancements and targeted improvements continue to enhance its performance, ensuring its relevance in real-world applications.

### **2.2.2 Faster R-CNN**

The Faster Region-based Convolutional Neural Network (Girshick, 2015) is a popular object detection model using a two-stage high-accuracy detection method. First, a Region Proposal Network (RPN) (Ren et al., 2015) creates possible bounding boxes for objects after a pre-trained Convolutional Neural Network extracts features from the input image. In the second stage, these proposals are refined and classified using another CNN. This architecture enables Faster R-CNN to achieve superior accuracy, particularly in challenging scenarios such as occlusions or complex object poses, making it highly suitable for agricultural applications like orchard management to achieve superior accuracy (Lokanath et al., 2017) compared to single-stage detectors.

It excels at handling occlusions, and challenging object poses, making it well-suited for scenarios like orchards where leaves might partially cover apples or apples hanging at different angles. Faster R-CNN's effectiveness in visually guided robotic arms highlights its potential for high-precision robots (Fu et al., 2020). However, the two-stage architecture comes at the cost of processing speed, making Faster R-CNN less ideal for real-time applications. Additionally, training Faster R-CNN effectively often requires large datasets with well-annotated objects.

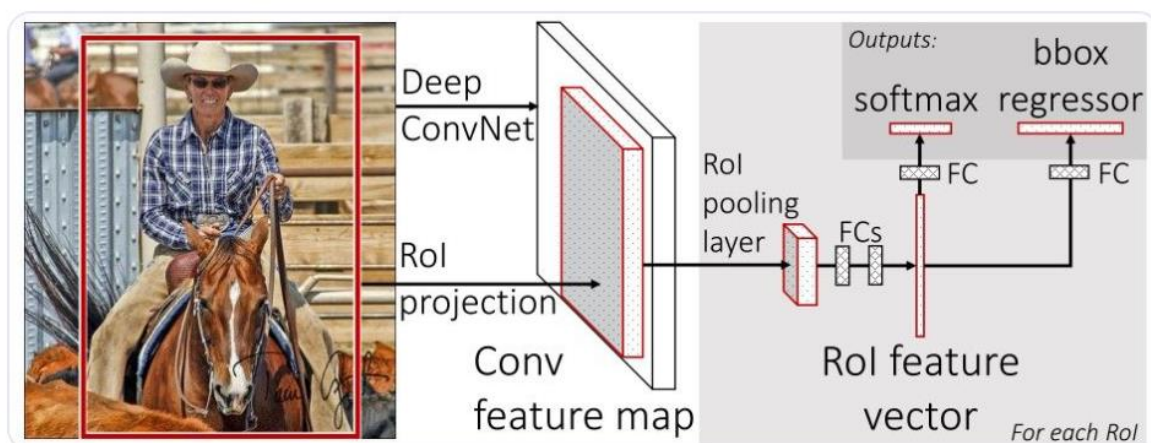


Figure 3 Faster RCNN Architecture

### Technical Specifications

The core idea of Faster R-CNN is to generate region proposals (Regions of Interest, or RoIs) efficiently using a Region Proposal Network and feed them into a convolutional network for classification and localisation. Figure 3 depicts different components involved in Faster RCNN architecture.

#### Region Proposal Network:

- The RPN generates region proposals directly from the feature map, eliminating the need for traditional, computationally intensive proposal generation methods such as selective search.
- By utilising anchors of varying sizes and aspect ratios, the RPN efficiently predicts object locations and scales, making it adaptable to diverse object shapes and sizes within a single image.
- The RPN also ranks the proposals based on their likelihood of containing objects, ensuring the most relevant regions are passed.

#### RoI Pooling:

- RoI Pooling standardises the variable-sized region proposals into fixed-size feature maps, enabling compatibility with the fully connected layers that follow.
- This step projects the proposals onto the convolutional feature map and extracts spatially uniform features, which is critical for accurate classification and bounding box refinement.
- RoI Pooling reduces computational complexity and preserves essential spatial information for precise localisation.

Classification and Bounding Box Regression: The fixed-size feature maps are processed through fully connected layers, leading to two separate outputs:

- **Classification:** Determines the object class using a softmax layer, providing probabilities for each class.
- **Bounding Box Regression:** Refines the coordinates of the proposed bounding boxes, improving localisation accuracy through regression techniques.

## **Applications and Usage**

Agricultural Applications:

- Research (Kang & Chen, 2020) highlights Faster R-CNN's effectiveness in detecting apples in orchards, particularly under conditions of partial occlusion and varying lighting. The model outperformed traditional methods in accuracy but struggled under heavy occlusions.
- Faster R-CNN has been combined with advanced techniques like transformer-based self-attention mechanisms (Kong et al., 2024), significantly enhancing its ability to handle complex orchard environments, achieving a mean Average Precision of 0.692.
- Studies (Le et al., 2019) demonstrate Faster R-CNN's application in guiding visually driven robotic arms, showcasing its potential for precision-focused tasks.

Harvest Robots:

- (Tahir et al., 2021) used an improved Mask R-CNN, an extension of Faster R-CNN, integrating ResNet and DenseNet for feature extraction. This model achieved a precision rate of 97.31% and a recall rate of 95.70%, making it suitable for robotic apple harvesting in challenging scenarios.

## **Advantages and Limitations**

Faster R-CNN is a reliable option for object detection in challenging situations because of its many benefits. It ensures excellent accuracy while recognising objects under challenging situations, like complex stances and partial occlusions. Because of its region-based methodology, the model is resilient when dealing with overlapping objects and changing lighting circumstances. Furthermore, the adaptability of Faster R-CNN makes it possible to incorporate sophisticated feature extraction methods, including transformer-based self-attention mechanisms, which might improve its functionality even more. Nevertheless, the

model has drawbacks. It is less appropriate for real-time applications because its two-stage architecture slows processing performance. Furthermore, implementation on low-power devices is complicated by its computational complexity, which necessitates substantial resources for both training and inference. Faster R-CNN also demands large and well-annotated datasets to achieve optimal performance, which can be a limiting factor in specific applications.

### **Recent Developments and Future Directions**

- Kong et al. (2024) introduced the Faster-RFormer, which integrates transformer-based self-attention mechanisms into the Faster R-CNN framework, improving global feature extraction and enhancing performance in occluded and variable lighting conditions.
- (Gong & Zhang, 2023) introduce significant improvements to Faster R-CNN for detecting apple leaf diseases. It integrates Res2Net and Feature Pyramid Network to enhance feature extraction for better handling of small and dense disease objects. RoIAlign replaces RoIPool, improving feature alignment and detection precision. Soft Non-Maximum Suppression (Soft-NMS) reduces missed detections in overlapping or complex conditions, significantly boosting accuracy.
- (Jia et al., 2020) utilised an improved Mask R-CNN model with ResNet and DenseNet for better feature extraction, achieving high precision and recall rates for apple detection in orchards.
- The study (Hou et al., 2023) addresses the challenge of detecting small disease spots in apple leaf images by proposing the FPN-ISResNet Faster R-CNN model, which combines Feature Pyramid Networks, Inception Squeeze-and-Excitation ResNet, and Faster R-CNN. The model demonstrated superior detection accuracy compared to SSD, VGG-Faster R-CNN, and other Faster R-CNN variations, achieving an average precision of 62.71%, with AP50 and AP75 of 93.68% and 70.94%, respectively. This approach effectively enhances accuracy and generalizability for detecting apple leaf diseases.

### **Challenges**

- Despite advancements, Faster R-CNN struggles with heavily occluded objects and maintaining real-time performance in embedded systems.
- Future research could optimise the model for faster inference while retaining its accuracy, particularly in low-power devices like robots in orchard management.

Faster R-CNN is a robust and accurate object detection model that is well-suited for agricultural and robotic applications. Its region proposal-based architecture performs exceptionally in detecting objects in challenging scenarios, including partial occlusion and varying lighting. However, its computational complexity and slower processing speed make it less ideal for real-time use. Recent advancements, such as integrating transformer-based mechanisms, continue to push its capabilities, ensuring its relevance in high-precision tasks like robotic apple harvesting.

### 2.2.3 You Only Look Once (YOLO)

The "You Only Look Once" (YOLO) series has significantly advanced real-time object detection through its single-stage architecture, enabling simultaneous object localisation and classification. Since its inception by (Redmon et al., 2015), YOLO has undergone multiple iterations, each enhancing performance and adaptability across various applications.

#### **Evolution of YOLO Models**

YOLOv1 introduced a single-stage detection pipeline that enabled faster processing by detecting and classifying objects in one step. However, it faced challenges with small object detection and complex scenes. YOLOv2 improved detection through anchor boxes and batch normalisation, enhancing stability and accuracy across different object scales. YOLOv3 further advanced object detection by introducing multi-scale detection using feature pyramids and a deeper backbone, DarkNet-53, which significantly improved performance for small objects.

With YOLOv4, the focus shifted towards real-world application optimisation, incorporating techniques like CSPNet and Mish activation functions to improve performance and speed. YOLOv4 also introduced advanced data augmentation methods such as Mosaic and DropBlock, making it robust to diverse conditions. YOLOv5, though unofficial, became famous for its ease of use in PyTorch and its optimisation for training and deployment. The later iterations, YOLOv6 and YOLOv7, focused on real-time applications, with YOLOv7 becoming one of the fastest detectors, prioritising both speed and computational efficiency.

YOLOv8 represents the latest evolution, combining the advancements made in previous versions with enhanced training techniques, a more refined architecture, and better handling of small objects and multi-object scenes. It is a highly efficient model suitable for various applications, from autonomous vehicles to medical imaging, owing to its ability to balance speed, accuracy, and resource consumption. YOLOv8 adopts a single-stage approach, significantly reducing processing time, crucial for real-time robotics applications.

YOLOv8 leverages a pre-trained CNN backbone (Lou et al., 2023) to extract image features, followed by a path aggregation network (S. Liu et al., 2018) that effectively combines information from different scales. This allows YOLOv8 to capture high-level semantic information and low-level details crucial for accurate object detection. Finally, the model predicts bounding boxes and class probabilities directly from the feature maps in a single pass. This streamlined architecture contributes significantly to YOLOv8's efficiency (Varghese & Sambath, 2024).

YOLOv8 shines in robotics due to its ability to balance speed and accuracy. Unlike some high-accuracy models that can be sluggish, YOLOv8 achieves good detection accuracy (Xiao et al., 2023) while maintaining processing speed – which is crucial for real-time applications in robots. This efficiency stems from its single-stage architecture. It comes in various versions, allowing you to choose the optimal balance between speed and accuracy based on your robot's needs and computational resources. While achieving the absolute highest accuracy might necessitate exploring more complex models, research by (Feng et al., 2021) demonstrates YOLO benchmark performance on various object detection tasks.

### **Technical Specifications**

By predicting bounding boxes and class probabilities in a single network run, YOLO models simplify object detection through a single-stage detection approach. This architecture is perfect for real-time applications because of its efficiency and speed design.

**The backbone for Feature Extraction:** The foundation of feature extraction Hierarchical features are extracted from the input image by the backbone, usually a Convolutional Neural Network (CNN) that has already been trained. From DarkNet-19 in YOLOv1 to the more profound and more effective DarkNet-53 in YOLOv3, and then to CSPDarkNet and other sophisticated networks in YOLOv4 and YOLOv5, YOLO has developed its backbone networks over the years. These backbones balance the requirement for low-level geographical features and high-level semantic information.

**Neck for Feature Aggregation:** The neck aggregates features from different backbone levels, enabling multi-scale detection. Recent iterations have added methods like Path Aggregation Networks (PAN) and Feature Pyramid Networks (FPN) to enhance the detection of objects of different sizes and in difficult situations like crowded backdrops and occlusion.

**Head for Prediction:** The detection head predicts bounding box coordinates, objectness scores, and class probabilities directly from the aggregated feature maps. Advanced anchor box mechanisms help YOLO predict bounding boxes at different scales, catering to small and large objects in the same image.

**Techniques for Performance Optimization:** YOLO models integrate several optimisation techniques to enhance accuracy and efficiency:

**Anchor Boxes:** Used to predict object dimensions at different scales, improving localisation.

**Advanced Activation Functions:** Mish and Leaky ReLU improve gradient flow, aiding faster convergence.

**Data Augmentation:** Techniques like Mosaic (combining multiple images) and CutMix enhance robustness across diverse datasets.

**Loss Functions:** Improvements in bounding box regression loss, such as CIoU and DIoU, have increased accuracy in localisation.

**End-to-End Training:** YOLO's design allows for end-to-end training, enabling joint optimisation of classification and localisation tasks. This simplifies the pipeline and ensures better synchronisation between object detection tasks.

The architecture's progressive enhancements across versions, such as introducing transformer-based attention mechanisms in YOLOv9, have made YOLO models increasingly capable of handling complex detection challenges. These specifications underline YOLO's adaptability, scalability, and efficiency, solidifying its role as a cornerstone in real-time object detection.

## Applications and Usage

The YOLO series has been widely adopted in various fields due to its real-time detection capabilities:

- **Agricultural Robotics:** YOLO models have been employed for fruit detection and harvesting tasks.
  - (Sekharamantry et al., 2023) implemented a deep learning technique using an enhanced YOLOv5 model to recognise apples. This model incorporated an attention mechanism, an improved loss function, and adaptive pooling to distinguish apples of various sizes in complex backgrounds. With an F1-score of 0.98, precision of 0.97, and recall of 0.99, the YOLOv5-based model demonstrated superior accuracy in addressing challenges like background complexity, motion blur, occlusion, and variable lighting conditions.
  - (Tian et al., 2019) used an improved YOLOv3 model coupled with DenseNet to identify apples at different growth stages under challenging conditions like occlusion, overlap, and changing lighting. The YOLOv3-Dense model outperformed the original YOLOv3 and Faster R-CNN models, providing accurate real-time apple detection. The model efficiently handled challenging orchard circumstances and offered improved feature propagation and reuse for dependable detection.
  - (Yang et al., 2024) studied an improved YOLOv5 model, dubbed CA-YOLOv5, to identify apples in challenging natural scenarios such as overlapping, occlusion, and variable illumination. By including Coordinate Attention (CA) modules in the backbone and a Bidirectional Feature Pyramid Network (BiFPN) in the neck, the model achieved a recall of 82.7%, mAP@0.5 of 89.8%, and an F1-score of 87.0%, outperforming the original YOLOv5.

## Advantages and Limitations

The YOLO series offers several advantages, making it a powerful object detection tool. Its single-stage architecture enables rapid processing, making it highly suitable for real-time applications where speed is critical. The unified design simplifies implementation and

deployment, while its versatility allows it to be applied across a wide range of domains requiring object detection. However, the YOLO models also have some limitations. Earlier versions struggled with detecting small objects, though recent iterations have addressed this issue to some extent. Additionally, challenges persist in accurately detecting objects in highly complex or cluttered environments, where overlapping and dense features can hinder performance. Despite these limitations, the YOLO series remains a leading choice for many real-time detection tasks due to its speed, simplicity, and adaptability.

### **Recent Developments and Future Directions**

Recent advancements in YOLO models highlight their adaptability and continued evolution to address complex detection challenges:

- The research (Malik & Mahmud, 2024) addresses precise weed detection using advanced object detection models in challenging field conditions. It evaluated YOLOv9, YOLOv8, YOLO World, and RT-DETR on two datasets, with YOLOv9 achieving the highest performance, attaining mean Average Precision values of 0.94 and 0.85 on datasets A and B, respectively. YOLOv9 also demonstrated superior speed to RT-DETR, highlighting its potential for precision weeding applications.
- (Tian et al., 2019) leveraged DenseNet integration with YOLOv3 to improve feature propagation and reuse, achieving dependable detection even in challenging orchard circumstances.
- (Yang et al., 2024) demonstrated the effectiveness of CA-YOLOv5 by enhancing the detection of apples under natural challenges, showing improvements in recall, mAP, and F1-score by integrating Coordinate Attention (CA) modules and BiFPN.
- (Sekharamantry et al., 2023) optimised YOLOv5 for apple detection by introducing an attention mechanism and adaptive pooling, addressing various issues like occlusion and complex backgrounds. These enhancements resulted in industry-leading accuracy metrics and robustness in real-world agricultural applications.

### 2.3 Lighting conditions

Research on how illumination affects object detection and image quality has been crucial, especially in domains where precise visual perception is crucial for agricultural robotics. Low light, shadows, and glare are illumination variations that can seriously impair object detection algorithms' effectiveness and result in missing or incorrect identifications. According to studies, poor illumination can create noise or hide crucial visual elements, making it difficult for conventional computer vision algorithms to process images efficiently. Researchers have explored various techniques to mitigate these issues, including adaptive histogram equalisation, gamma correction, and machine learning approaches trained to recognise objects under different lighting scenarios. Moreover, deep learning models, particularly convolutional neural networks (CNNs), have been developed with enhanced robustness to lighting variations by being trained on diverse datasets that include images captured under a wide range of lighting conditions. This line of research continues to evolve, with ongoing efforts to improve the resilience of object detection systems to dynamic lighting environments, thereby increasing their reliability and accuracy in real-world applications.

The illumination in natural settings varies with the season and weather. Photos in low light conditions, including overcast days and nights, typically have low contrast and brightness (F. Liu et al., 2022). These poor-quality photos can have an adverse effect on the assignment's success rate since they could make the activity of gathering apples less visually appealing. This can result in poor apple localisation and identification performance. The research (Mukherjee et al., 2021) addresses the problem of object detection models failing in difficult lighting conditions. They used High Dynamic variety (HDR) imaging instead of Standard Dynamic Range (SDR) images, which capture a wider variety of brightness levels. By preserving crucial visual characteristics, this technique helps identify objects more precisely in challenging lighting situations, such as brightly lit or dimly lit environments. By training models on HDR photos, they achieved greater detection accuracy than with traditional SDR-trained models.

Under low light conditions, object detection models have poor detection accuracy. Research done by (J. Wang et al., 2023) aims to overcome this problem. In the researchers' enhanced YOLOv5-based model, DK\_YOLOv5, low-light picture-enhancing approaches and architectural improvements include the R-SPPF module for faster inference and more robust feature representation. The study (Zhang et al., 2024) investigates the impact of adverse lighting conditions on the accuracy of fruit detection and localisation in automated apple

harvesting. It focuses on developing an accurate method for fruit detection and localisation under challenging light conditions using deep learning techniques. By exploring the LE-YOLO model, the study enhances the traditional YOLOv5 network with an image enhancement module and an attention mechanism to improve robustness and accuracy in apple detection and localisation. Each input image is lightly enhanced for better detection performance. The lightweight image enhancement module improved the YOLOv5s detection model by adding an attention mechanism and improving the loss function to enhance the detection results further.

## **2.4 Gaps in the current research**

### **2.4.1 Identification of Lighting Conditions**

Support Vector Machines have proven effective in classifying various lighting conditions in images because they can handle high-dimensional data and make accurate binary or multi-class distinctions. SVMs work by finding the optimal hyperplane that best separates data points in a high-dimensional space. This makes them particularly suitable for tasks like lighting condition classification, where multiple features (such as contrast, brightness, and texture) may define different lighting conditions. SVMs efficiently process the entire image by considering these features, which helps accurately identify lighting scenarios like sunlight, shadow, blur, or darkness. In our case, SVM can analyse the overall histogram and pixel distribution to classify images under specific lighting conditions before further processing.

Moreover, SVM's ability to generalise from smaller datasets while maintaining accuracy makes it advantageous in real-world scenarios where comprehensive labelled datasets for every lighting condition might not be available. This provides a robust framework to detect conditions under which images are captured, whether the lighting is harsh, soft, or contains various noise artefacts such as shadows. The efficiency of SVM in lighting condition identification enables the selection of appropriate image preprocessing filters, ensuring optimised input for subsequent object detection models like YOLOv9.

### **2.4.2 Advantages of Using SVM Over Other Machine Learning Algorithms**

SVM has several advantages over other machine learning algorithms for image classification tasks, such as identifying lighting conditions. Firstly, SVM is known for its robustness, especially when dealing with high-dimensional data, and it tends to avoid overfitting even when working with smaller datasets. This makes it particularly effective when data is limited

or noisy. Unlike neural networks, which require significant tuning of hyperparameters, SVMs are more straightforward and can achieve high performance with less complexity.

Furthermore, SVMs are particularly suited to binary classification problems, making them a strong candidate for distinguishing between lighting conditions. While neural networks and decision trees may provide more flexibility for multiclass classification, they often require more data and computational resources. Using kernel methods to translate input features into higher-dimensional spaces, SVMs may also effectively handle non-linear data. This is advantageous when distinguishing complex lighting conditions that might not be linearly separable.

SVM maintains its computational efficiency when compared to methods such as KNN (K-Nearest Neighbors), which can become computationally costly as the size of the dataset increases. Furthermore, SVMs are more dependable in real-world applications like lighting classification, where irregular lighting and occlusion may produce outliers because they can handle noisy data and outliers better.

### **2.4.3 Applying the Right Image Preprocessing Filter Based on Lighting Condition**

The proper image preprocessing filters can be used to eliminate noise or improve the image for improved object detection once the lighting state of an image has been identified using SVM. For instance, applying filters such as adaptive gamma correction or histogram equalisation might improve the image's contrast and brightness in low light, making things easier to detect. Dehazing filters or contrast-limiting methods may be more suitable in areas with high levels of sunlight to reduce glare and shadows and enhance image quality.

The accuracy of object detection algorithms like YOLOv9 can be increased by preprocessing the image according to its lighting conditions. Before using the detection method, this preprocessing ensures that the image is normalised and that any noise or illumination anomalies are eliminated. The preprocessing filters are essential in bridging the gap between raw picture data and precise object detection, allowing the model to concentrate on pertinent features instead of being distracted by low-quality images.

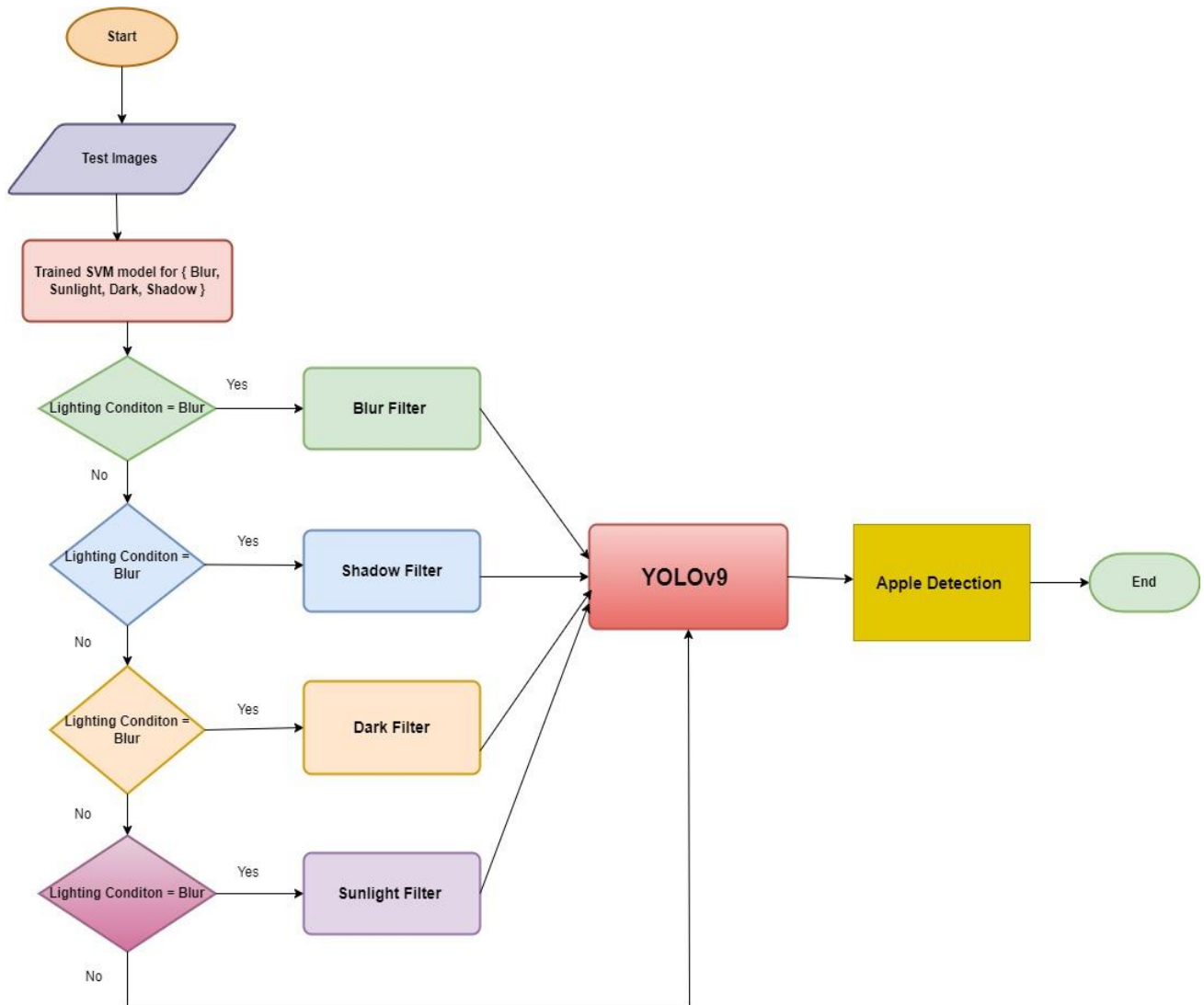
#### **2.4.4 YOLOv9 Object Detection: Pre and Post-Filter Results**

Experimenting with YOLOv9 object detection before and after applying the appropriate filters can provide valuable insights into how lighting conditions impact detection accuracy. YOLOv9 may struggle to detect objects accurately in images without preprocessing, especially in scenarios with harsh sunlight, deep shadows, or low-light conditions. However, the image quality improves once filters are applied based on the classified lighting condition, leading to better feature extraction and higher detection accuracy.

For instance, applying dehazing or contrast reduction filters can make previously undetectable objects clear to the YOLOv9 algorithm in environments with glare. Similarly, noise reduction and contrast enhancement in low-light conditions can make object contours sharper, allowing YOLOv9 to detect objects more reliably. This highlights the importance of integrating preprocessing filters into the object detection pipeline and demonstrates that YOLOv9 can perform efficiently across various lighting conditions when combined with suitable filters.

### Chapter 3 Methodology

The methodology employs an integrated pipeline to enhance apple detection under varying lighting conditions. This pipeline combines Support Vector Machine (SVM) classification, custom image preprocessing filters, and the YOLOv9 object detection model. The process begins with SVM classifying the input image based on distinct lighting conditions such as sunlight, shadow, blur, or darkness, each posing unique challenges to image clarity. For instance, overexposure caused by direct sunlight can obscure details, shadows may lower contrast, and blurring may result from motion or camera shake. Based on the identified lighting condition, an appropriate filter is applied: the sunlight filter adjusts brightness balance using contrast stretching, the shadow filter employs gamma correction to enhance darker regions, the blur filter improves sharpness, and the darkness filter increases visibility through contrast enhancement. The pre-processed images are then input to the YOLOv9 model for accurate apple detection, leveraging improved clarity and contrast. This comprehensive pipeline effectively addresses lighting variations, ensuring robust and precise apple detection.



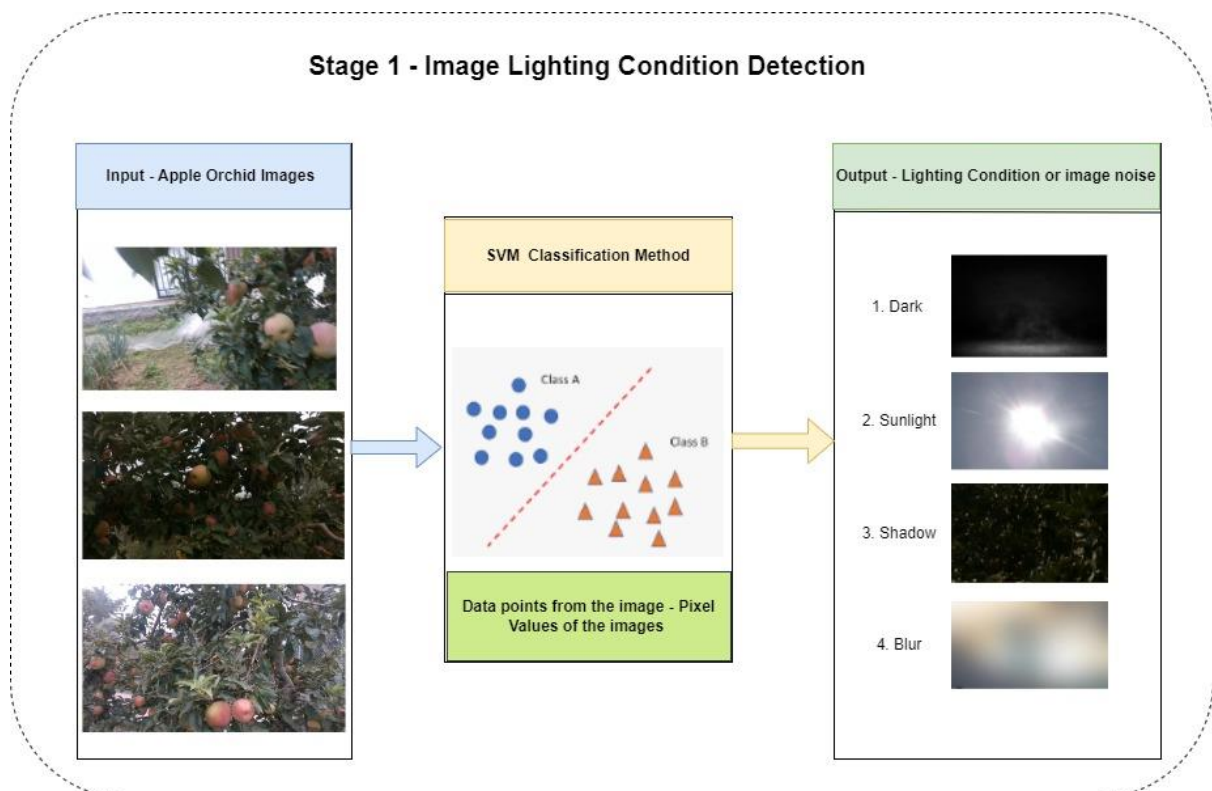
*Figure 3 Flowchart of the overall system - lighting condition detection, filter selection and apple detection process using SVM and YOLOv9.*

Figure 3 illustrates a comprehensive apple detection pipeline that operates under varying lighting conditions by integrating SVM-based lighting condition detection, custom image preprocessing filters, and the YOLOv9 detection model. The process initiates with the input of test images, which a trained SVM model first analyses to classify the prevailing lighting condition into one of four categories: blur, shadow, darkness, or sunlight. After identifying the lighting condition, a corresponding image filter is applied to enhance quality. For instance, a blur filter is used to sharpen edges and clarify apple boundaries; a shadow filter selectively brightens darker areas while preserving well-lit regions; a dark filter boosts brightness and contrast to highlight apples in low-light scenarios; and a sunlight filter adjusts brightness and contrast to minimise glare and enhance apple visibility under overexposure. This tailored

preprocessing ensures optimal image clarity across diverse lighting conditions, enhancing the accuracy of subsequent detection steps. The refined image is then processed by the YOLOv9 model, which performs precise apple detection based on the improved visual input. This integrated approach adapts to real-time changes in environmental conditions and significantly improves detection reliability, demonstrating a robust solution for automated apple detection in dynamic orchard settings.

### 3.1 Image Lighting or Noise Condition Detection

Using a machine learning model, this stage is critical for identifying the prevailing lighting conditions or noise in apple orchard images. The input for this stage is an image captured by the robotic harvesting system. The dataset comprises images taken at different times and locations within orchards. The output of this stage is a label indicating the identified lighting condition or noise in the image generated by an SVM classifier trained to recognise specific classes. The SVM model is trained on diverse lighting conditions and noise levels to enhance its detection accuracy. Figure 4 depicts the process of detecting lighting conditions in the image.



*Figure 4 Image Lighting Condition Detection using SVM*

The input image may have different resolution sizes and be in the JPEG or PNG format. The image is transformed into pixel values and moulded into a one-dimensional vector array to prepare it for Support Vector Machine classification. Each image is resized to  $128 \times 128$  pixels before creating the one-dimensional vector representation. Standardising the size of each image ensures better accuracy during SVM training and significantly reduces processing requirements. This essential step offers several advantages:

- Ensures uniform dimensions across all images, maintaining input consistency for the models.
- Reduces processing time and memory usage, enabling faster training and inference.
- Enhances training efficiency by focusing on crucial image features.
- Aligns input dimensions with pre-trained model requirements, facilitating transfer learning.
- Promotes model robustness by standardising object sizes and reducing overfitting.

The resizing process uses the Bilinear Interpolation method to transform the image to the standard size of  $128 \times 128$  pixels. Bilinear Interpolation generates new pixel values based on surrounding pixels while preserving original image properties. This method scales the coordinates based on the ratio of the original dimensions to the new dimensions. Four neighbouring pixel values are identified from the mapped position's upper, lower, left, and right sides for each new pixel position. Weights are assigned based on horizontal and vertical distances from the mapped position, with closer neighbours receiving higher weights. The weighted average of the four closest pixels is then applied to each pixel in the input image to get the new pixel value. A  $128 \times 128$  pixel resizing is applied to the final image.

Bilinear interpolation formula:

A point  $(x', y')$  in the resized image, the pixel value is determined based on the four nearest pixels in the original image:

$$I'(x', y') = (1 - \Delta x) \cdot (1 - \Delta y) \cdot I(x_1, y_1) + \Delta x \cdot (1 - \Delta y) \cdot I(x_2, y_1) + (1 - \Delta x) \cdot \Delta y \cdot I(x_1, y_2) + \Delta x \cdot \Delta y \cdot I(x_2, y_2)$$

Where:

- $I'(x', y')$  Interpolated pixel value in the resized image.
- $I(x_1, y_1), I(x_2, y_1), I(x_1, y_2), I(x_2, y_2)$  Original pixel values at the four nearest neighbours.
- $\Delta x = x - x_1$  The fractional part of the x-coordinate represents the distance from  $x_1$ .
- $\Delta y = y - y_1$  The fractional part of the y-coordinate, representing the distance from  $y_1$ .

The resized image, now containing 3D vector values, serves as input to the SVM as a single data point. This vector is flattened from 3D to 1D space to meet the SVM models' fixed-size feature vector requirement. The SVM model's resulting input vector is used for training, validation, and testing.

Support Vector Machines (Cortes et al., 1995) are a type of supervised machine learning algorithm frequently used for classification tasks. This study employs SVMs to detect lighting conditions or noise levels in apple orchard images. The primary goal is sorting photos into groups corresponding to various lighting or noise levels. SVMs are selected because they efficiently handle high-dimensional data and address linear and non-linear classification issues. SVMs are chosen above alternative categorisation models for several reasons.

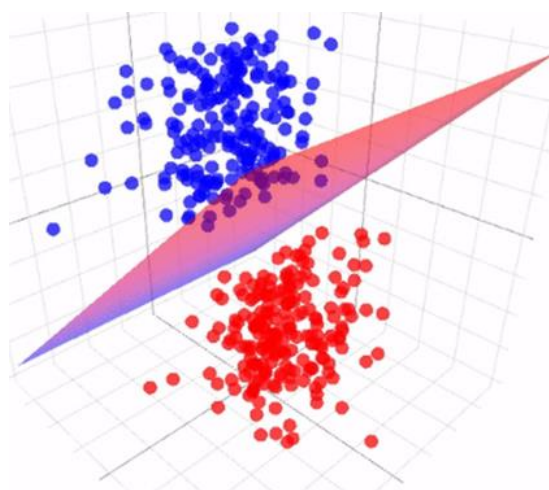
SVM excels in handling high-dimensional data, such as flattened image vectors, where each pixel represents a feature.

- SVMs excel in managing high-dimensional data, such as flattened image vectors where each pixel is treated as a feature.
- They perform well with smaller dataset sizes, which is advantageous when dealing with limited lighting conditions or noise images.
- SVMs handle non-linearly separable data effectively, leveraging the Radial Basis Function (RBF) kernel to distinguish between overlapping image pixel values in different classes.

Each image is represented by features that describe the lighting environment, such as brightness levels, contrast, and pixel intensity distributions. These features are transformed into input vectors for the SVM model, enabling it to identify patterns that distinguish one lighting condition or noise level from another in an N-dimensional space.

The SVM model is trained using a labelled dataset of images converted to 1D vectors, where each image is pre-classified into one of the lighting categories. During training, the SVM

algorithm aims to find the optimal hyperplane that best separates the categories by maximising the margin between the closest points from different classes, known as support vectors. The dimensionality of the hyperplane corresponds to the number of features represented in N-dimensional space. Each data point is a feature vector derived from the image's lighting or noise characteristics, as depicted in Figure 5.



*Figure 5 Data points from image feature vectors*

After extracting the data points, the SVM identifies the hyperplane that separates them. The margin is the distance between the hyperplane and the nearest data points from any class, known as support vectors. For linear SVMs, the hyperplane is represented as:

$$wx + b = 0$$

Where  $w$  is the weight vector,  $x$  is the input vector, and  $b$  is the bias term.

- $wx + b > 0$ , The point is on one side of the hyperplane.
- $wx + b < 0$ , The point is on the other side.

The objective of SVMs is to maximise the margin. However, linear models may not perform well for complex data or multiple classes. Non-linear SVMs offer significant advantages in handling high-dimensional data, making them suitable for detecting nuanced variations. When data is not linearly separable, a linear hyperplane cannot effectively classify it. In such cases, the 'Kernel Trick' maps data into a higher-dimensional space where linear separation is

possible. Instead of transforming the original data points, a kernel function computes the inner product in the higher-dimensional space.

Input vector  $x$  The dataset contains features that represent a single data point.

The transformed vector becomes a given input vector.  $x$ , the feature mapping function  $\phi(x)$ , Transforms the input into a higher-dimensional space, making it linearly separable:

$$\phi(x) = [\phi_1(x), \phi_2(x), \dots, \phi_n(x)]$$

- $\phi(x)$  Is a non-linear transformation of the input vector  $x$ .
- The transformation function  $\phi(x)$  Maps the input space to a higher-dimensional feature space, where linear separation might be possible.

In this transformed space, the SVM finds the optimal hyperplane. The objective function of SVMs is to minimise:

$$\min_{w, b, \xi} \left( \frac{1}{2} |w|^2 + C \sum_{i=1}^n \xi_i \right)$$

Subject to:

$$y_i(w \cdot \phi(x_i) + b) \geq 1 - \xi_i, \quad \text{for } i = 1, \dots, n$$

- $w$  Is the weight vector in the feature space.
- $b$  is the bias term.
- $\xi_i$  Are the slack variables to allow some misclassification, ensuring a soft margin.
- $C$  is the regularisation parameter controlling the trade-off between maximising the margin and minimising the classification error.

To avoid explicitly computing  $\phi(x)$  The kernel trick is used. The kernel function computes the dot product in the transformed feature space:

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

For the RBF kernel, this is given by:

$$K(x_i, x_j) = \exp \left( -\gamma |x_i - x_j|^2 \right)$$

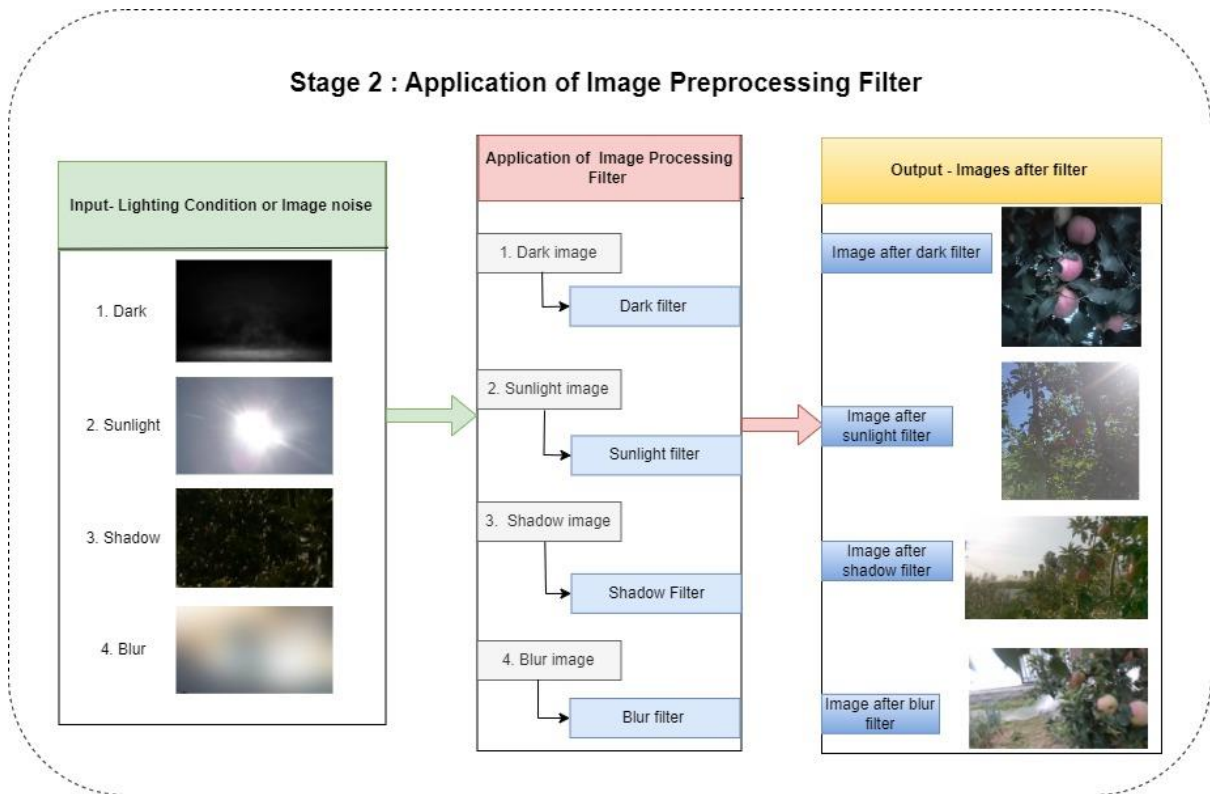
- $\gamma$  is a parameter that controls the influence of each training example, determining the width of the Gaussian.
- $|x_i - x_j|^2$  represents the squared Euclidean distance between two data points.
- The RBF kernel maps the input data to an infinite-dimensional feature space, making it possible to handle complex patterns.

SVMs are remarkably efficient with smaller datasets, making them well-suited for apple harvesting robots, where collecting large, labelled datasets under varied lighting conditions can be challenging. This efficiency ensures that SVMs deliver accurate and reliable classification, even with limited training data.

### **3.2 Filter Selection Based on Image Lighting or Noise Conditions**

The SVM classifies the original input image into one of four lighting conditions: dark, sunlight, shadow, or blur. Based on this classification, an appropriate filter is selected to preprocess the image. This filtering step aims to reduce noise or adjust lighting levels, enhancing image quality and facilitating more effective object detection.

The correct filter selection is critical at this stage, as each filter is specifically designed to address a particular lighting challenge. The output of this stage is a preprocessed version of the original image, prepared to improve detection accuracy in subsequent steps. The system effectively minimises lighting-related distortions by applying these filters before sending images to the detection model, improving overall apple detection performance. This section discusses the design principles of each filter and how they enhance image quality under different lighting conditions. Figure 6 illustrates the application of different filters based on the detected lighting or noise conditions.



*Figure 6 Image Pre-processing filter application on the original image.*

Several image enhancement methods are applied to improve the quality of input images before apple detection. Each method targets specific characteristics, such as contrast, brightness, or noise level, to enhance visibility and clarity.

**Color Space Conversion (BGR to LAB):** Color space conversion from BGR to LAB is an essential preprocessing step that separates luminance (L) from chrominance components (A and B) (Lakhwani et al., 2015). This separation allows brightness adjustments to be made independently of colour information, making it suitable for enhancing contrast without affecting colour balance.

$$I_{LAB} = f(I_{BGR})$$

Where:

- $I_{LAB}$  Is the image in LAB colour space.
- $f(I_{BGR})$  Is the conversion function from BGR to LAB colour space.

Enhances control over brightness, facilitating adjustments that align with human vision.

**Histogram Equalization:** This process involves dispersing the pixel intensity values of a picture over the entire range of potential values to improve contrast (Pizer et al., 1987). Usually, this procedure is used on a grayscale image or a colour space's lightness channel. By equalising the histogram, the method improves the visibility of features, making details more distinguishable throughout the image. It is beneficial for enhancing images with poor contrast caused by uneven lighting or a limited dynamic range.

The new pixel intensity after histogram equalisation is calculated as follows:

$$I_{eq}(x, y) = \left\lfloor \frac{CDF(I(x, y)) - CDF_{min}}{N - CDF_{min}} \times (L - 1) \right\rfloor$$

Where:

- $I_{eq}(x, y)$  The equalised pixel intensity at coordinates  $(x, y)$ .
- $CDF(I(x, y))$  Cumulative distribution function value for pixel intensity at  $(x, y)$ .
- $CDF_{min}$  Minimum CDF value in the image.
- $N$  Total number of pixels in the image.
- $L$  Number of intensity levels (usually 256 for an 8-bit image).

**Deblurring (Gaussian Blur):** This technique reduces high-frequency noise and improves image quality. It involves convolving the original image with a Gaussian kernel, which smooths it by averaging the pixel values within a specified window. This process effectively minimises sharpness caused by noise or artefacts, making critical structures in the image more distinguishable. Gaussian blur is particularly useful for noise reduction while retaining essential image details and preparing the image for further analysis.

The Gaussian function is defined as:

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

Where:

- $G(x, y)$  Gaussian function value at coordinates  $(x, y)$ .
- $\sigma$  The standard deviation of the Gaussian kernel.

This method reduces sharpness caused by noise, enabling more precise identification of edges.

**CLAHE (Contrast Limited Adaptive Histogram Equalization):** CLAHE is designed to enhance contrast in images with uneven lighting. It operates on images in the LAB colour space, specifically focusing on the lightness channel to modify brightness without affecting colour (Reza, 2004). The image is divided into small regions (tiles), and histogram equalisation is applied within each tile. A contrast limiting mechanism prevents over-enhancement of noise by capping the intensity at a specified threshold. This local contrast enhancement makes dark regions more visible while avoiding the artefacts typically introduced by standard histogram equalisation, making CLAHE particularly effective in improving image quality under challenging lighting conditions.

CLAHE enhances the lightness channel as:

$$I_{CLAHE}(x, y) = \text{CLAHE}(I_L(x, y))$$

Where:

- $I_{CLAHE}(x, y)$  Output pixel value after CLAHE.
- $I_L(x, y)$  Lightness channel of the LAB image.

This method enhances local contrast effectively, improving visibility in uneven lighting conditions.

**Gamma Correction:** Gamma correction is a non-linear transformation technique used to adjust the overall brightness of an image. It modifies pixel intensity values based on a specified gamma value, applying a power-law function. This adjustment can brighten or darken the image, making it particularly effective for images with shadowed areas (Rahman et al., 2016). Gamma correction enhances details in bright and dark regions by selecting an appropriate gamma value, resulting in a more visually balanced image.

The gamma-corrected pixel intensity is:

$$I_{\text{out}}(x, y) = I_{\text{in}}(x, y)^\gamma$$

Where:

- $I_{\text{out}}(x, y)$  represents the output pixel intensity after gamma correction.
- $I_{\text{in}}(x, y)$  Is the original pixel intensity.
- $\gamma$  Is the gamma value that controls the transformation.

This method adjusts brightness levels, making shadowed areas more visible.

**Color Channel Denoising:** Color channel denoising is a technique that reduces noise while preserving image quality by denoising each colour channel separately. The input is an image where each colour channel (Blue, Green, Red) is processed individually using Non-Local Means (NLM) denoising (Nam et al., 2016). This approach averages similar patches within the image to reduce noise while retaining important edges and details. The technique is particularly effective for images affected by blur or low-quality captures, enhancing clarity while preserving the original colour characteristics.

The denoised pixel value is:

$$NL(I)(x) = \sum_{y \in \mathcal{N}(x)} w(x, y) \cdot I(y)$$

Where:

- $NL(I)(x)$  The denoised pixel value at position  $x$ .
- $w(x, y)$  Weight is calculated based on the similarity between patches.
- $\mathcal{N}(x)$  Neighbourhood of pixel  $x$ .
- $I(y)$  Intensity of the neighbouring pixel  $y$ .

This method preserves edges and details after reducing the noise.

### Sunlight Filter

The sunlight filter is designed to manage the excessive brightness caused by direct sunlight, enhancing contrast while reducing sharpness to improve the visibility of features without overexposure. The critical techniques employed in this filter include:

- **Color Space Conversion:** Converts the image to the LAB colour space, isolating the lightness channel to facilitate more precise contrast adjustments.
- **Histogram Equalization:** Enhances the lightness channel, improving overall contrast, especially under high sunlight exposure.
- **Deblurring:** Reduces sharpness induced by intense sunlight, providing a more balanced and precise image appearance.

Figures 7 and 8 show the pictures of the images before and after the sunlight filter application on the image.



*Figure 7 Image before and after the sunlight filter*



*Figure 8 Image before and after the sunlight filter*

## Shadow Filter

The shadow filter is designed to enhance visibility in darker regions of the image by selectively increasing brightness, making details in shadowed areas more prominent without significantly affecting the rest of the image. The primary technique employed in this filter is:

- Gamma Correction: Adjusts brightness by applying a gamma value greater than 1, effectively brightening shadowed areas while maintaining the overall image balance.

Figure 9 and Figure 10 show the pictures of the images before and after the dark filter application on the image.



*Figure 9 Image before and after shadow filter*



*Figure 10 Image before and after shadow filter*

## Dark Filter

The dark filter is developed to enhance images captured under low-light conditions by increasing contrast in dimly lit areas while minimising noise amplification. This results in improved visibility of critical features. The techniques employed in this filter include:

- Color Space Conversion (BGR to LAB): Transforms the image to the LAB colour space, focusing on the lightness channel for targeted contrast enhancement.
- CLAHE (Contrast Limited Adaptive Histogram Equalization): Applied to the lightness channel to enhance contrast, specifically in dark regions, effectively preventing excessive noise.

Figures 11 and 12 show the pictures of the images before and after the dark filter application on the image.



*Figure 11 Image before filter and after dark filter*



*Figure 12 Image before filter and after dark filter*

### **Blur Filter**

The blur filter mitigates image noise and blurriness by independently denoising each colour channel, thereby preserving colour integrity and improving image clarity. This approach is particularly beneficial in motion blur or low-quality image captures. The techniques employed include:

- **Color Channel Denoising:** Denoises each colour channel individually while maintaining the colour properties, resulting in a more precise and sharper image.

Figures 13 and 14 show the pictures of the images before and after the blur filter application on the image.



*Figure 13 Image before and after blur filter*

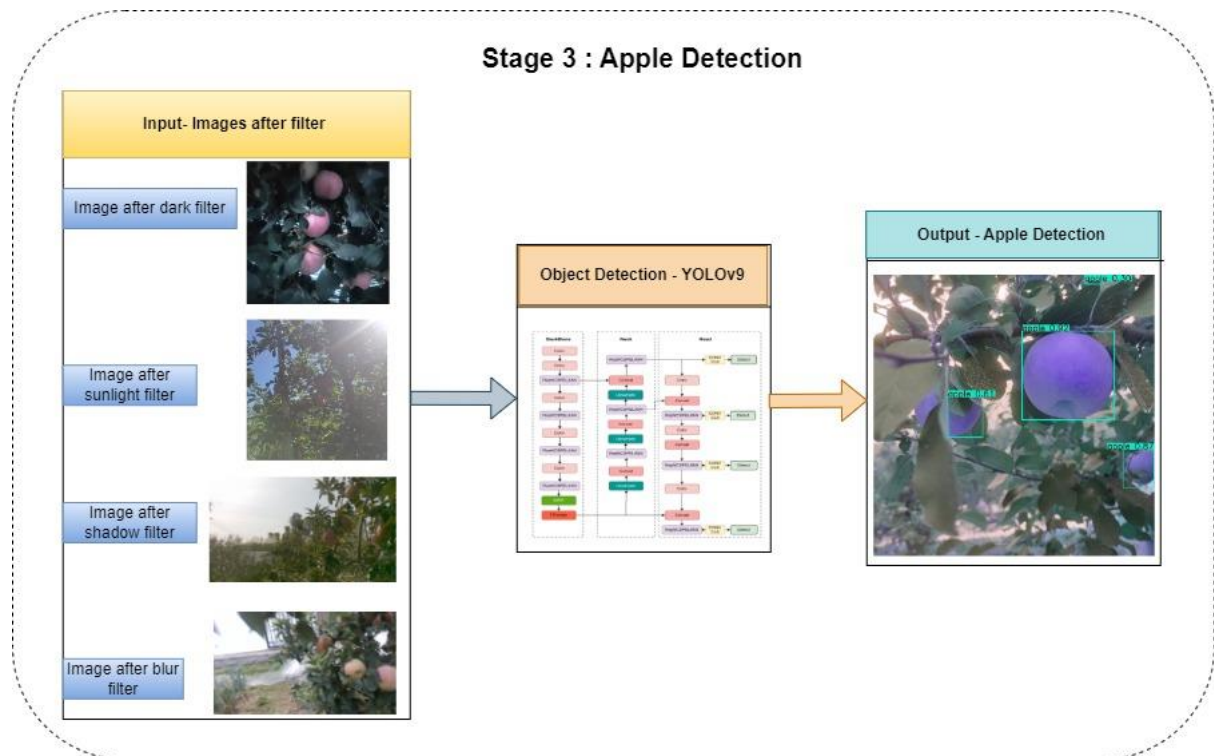


*Figure 14 Image before and after blur filter*

Each filter enhances the visibility of apple features, such as edges and contours, facilitating more accurate identification by the detection model. The processed image is then passed to the next stage, where the YOLOv9 detection model performs apple detection.

### 3.3 Apple Detection

Pre-processed photos with different filters, including dark, sunlight, shadow, and blur filters, are fed into the system's final stage. Using an object detection model, these improved photos are used as input for the object detection model. Apples are recognised and positioned within the pictures. The YOLOv9 framework is used for the detection process. It is well known for its accuracy in identifying things in photos and its efficiency in real-time object recognition. YOLOv9 improves the algorithm's ability to recognise apples in the improved photographs.



*Figure 15 Apple Detection using YOLOv9*

Figure 15 illustrates the apple detection process using YOLOv9, where images enhanced by different filters are input into the YOLOv9 model for object detection. This accurately identifies apples in the output image with the bounding boxes.

YOLOv9 (You Only Look Once, version 9) represents the latest evolution in the YOLO family of models, which are known for their ability to perform real-time object detection with impressive speed and accuracy (C.-Y. Wang et al., n.d.). Previous YOLO models, such as YOLOv3 and YOLOv5, focused on improving detection speed and accuracy in various conditions. YOLOv9 builds on these strengths, incorporating architectural enhancements that significantly boost its efficiency and precision. In tasks like apple detection, where lighting conditions (such as sunlight, shadows, and blur) can significantly affect detection quality, YOLOv9's ability to adapt and improve under such challenging scenarios makes it the ideal choice. One of the critical advancements of YOLOv9 is introducing a novel technique called Programmable Gradient Information (PGI), which aims to tackle information bottleneck issues inherent in deep learning models. This bottleneck typically leads to the loss of essential data during the feedforward process, which can result in inaccurate gradient updates and hinder the performance of deep networks. PGI offers a solution by generating reliable gradients through an auxiliary reversible branch, ensuring the deep features retain key characteristics for better object detection.

YOLOv9 offers a better balance between speed and accuracy than two-stage object detection models like Faster R-CNN. While models like Faster R-CNN are known for precise localisation, they tend to be computationally heavier and slower, making them less suitable for real-time applications. YOLOv9 is optimised to run efficiently on less powerful hardware, providing a seamless performance edge in agricultural applications where real-time results are often necessary. Unlike two-stage models like Faster R-CNN, which generate region proposals before classification, YOLOv9 is a single-shot detector. This means it performs both the detection and classification in one step, drastically improving detection speed. This is especially important in dynamic environments like orchards, where real-time apple detection is crucial for automated harvesting systems. Its architecture is designed to improve object localisation accuracy while maintaining its hallmark speed.

Figure 16 depicts the major components in identifying the objects from the input image. Object detection is achieved through a streamlined process of transforming input images into accurately localised bounding boxes for distinct object classes, such as apples. It has three main components: the backbone, neck, and head.

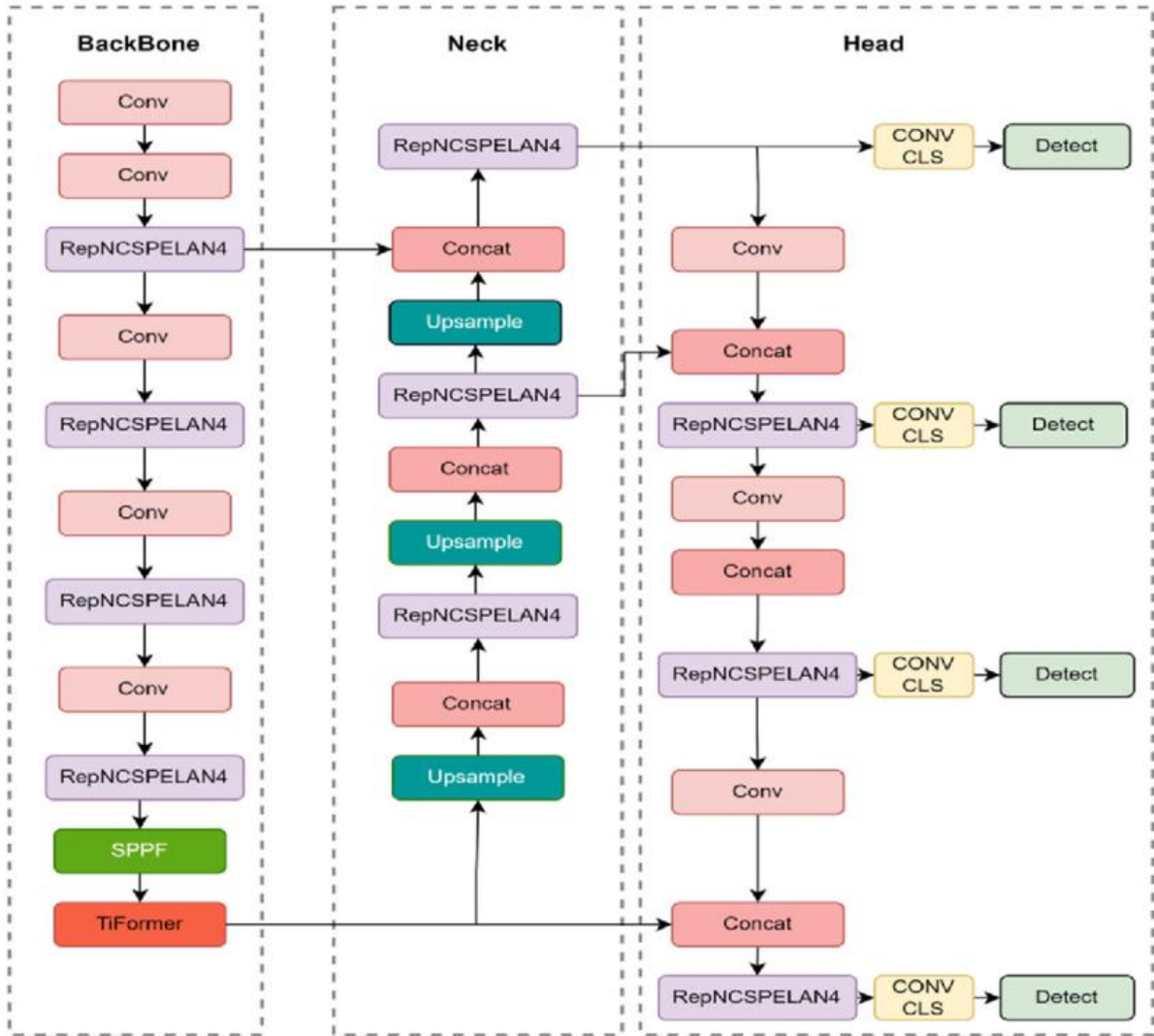


Figure 16 YOLOv9 architecture with backbone, neck and head

The Backbone is the initial part of the YOLOv9 architecture and is responsible for extracting essential feature maps from the input image. This component consists of a deep convolutional neural network, which processes the image through multiple convolutional layers to capture spatial hierarchies and different levels of abstraction. The convolution operation within the backbone is represented as:

$$F_l = \sigma \left( \sum_{i=1}^{C_{in}} W_l^i * I^i + b_l \right)$$

- $F_l$  Feature map output at layer  $l$ .
- $I^i$  Input feature map from channel  $i$ .
- $W_l^i$  Convolutional filter weights at  $l$  layer for channel  $i$ .

- $b_l$  Bias term at layer  $l$ .
- $\sigma$  Non-linear activation function, such as ReLU.
- $*$  Convolution operation.

The neck aggregates features from different scales, enabling multi-scale detection vital for identifying objects of varying sizes. The neck employs upsampling and concatenation techniques to fuse features from different layers, making the model more adaptable to changes in object size. The upsampling operation is represented as:

$$F_{up} = \text{Upsample}(F_l)$$

The concatenation of features from different scales is expressed as:

$$F_{concat} = [F_{low}; F_{mid}; F_{high}]$$

- $F_{up}$  Upsampled feature map from lower resolution to higher resolution.
- $F_{concat}$  Concatenated feature map, combining low, mid, and high-level features.
- $[\cdot ; \cdot]$  Concatenation operator.

The neck ensures that fine and coarse details are preserved for accurate object detection.

The Head component of YOLOv9 predicts bounding boxes, class probabilities, and objectness scores. It uses regression-based techniques to determine the bounding box coordinates and classification scores. The regression for bounding box prediction is represented as:

$$\hat{b} = \arg \min_b |b - \text{True}_b|$$

- $\hat{b}$  Predicted bounding box
- $\text{True}_b$  Ground truth bounding box.
- $|\cdot|$  distance metric, often based on the Intersection-over-Union (IoU) loss.

And objectness score

$$P_{obj} = \sigma(\text{Conv}(F_{head}))$$

- $P_{obj}$  Objectness probability score.
- $\text{Conv}(F_{head})$  Sigmoid activation function.
- Convolution applied to the head's feature map

The head integrates these predictions to output the final detection results:

$$\text{Detection} = \{\hat{b}, P_{obj}, P_{class}\}$$

- $\hat{b}$  Predicted bounding box coordinates.
- $P_{obj}$  Objectness probability.
- $P_{class}$  Predicted class probability for each detected object

YOLOv9 was chosen for its efficiency in real-time detection, adaptability to complex lighting conditions, and superior localisation abilities, adept at identifying apples even when they are partially obscured, making it the best fit for dynamic environments like apple orchards. These features significantly improve over earlier models like YOLOv5 or two-stage detectors like Faster R-CNN.

## Chapter 4 Experiment Setup and Testing

This chapter describes the experimental framework designed to test apple detection under varying lighting conditions in orchard environments. The setup is structured into three key stages, each with specific aims and methodologies that collectively contribute to assessing the effectiveness of filters, classification accuracy, and the final detection performance of YOLOv9. This experimental setup aims to create a robust pipeline for apple detection that dynamically adapts to different lighting and noise conditions.

### 4.1 Experiment Setup

This study's experimental setup involves three stages, each designed to progressively evaluate and enhance the apple detection capabilities of the YOLOv9 model under varying lighting conditions using the filters. These stages aim to test the effectiveness of custom filters and SVM classification integration, ultimately improving detection accuracy across diverse lighting environments. The details of each experimental phase are outlined as follows:

In the first phase, the YOLOv9 model is tested with unprocessed images from four lighting condition folders: blur, sunlight, dark, and shadow. These images were manually sorted into separate folders based on their lighting characteristics. The setup involves configuring the YOLOv9 code to execute apple detection on each category, allowing a baseline comparison of the model's performance under these challenging conditions. This step establishes the model's accuracy without preprocessing or filtering, providing an essential reference for subsequent experiments. This process is depicted in Figure 17.

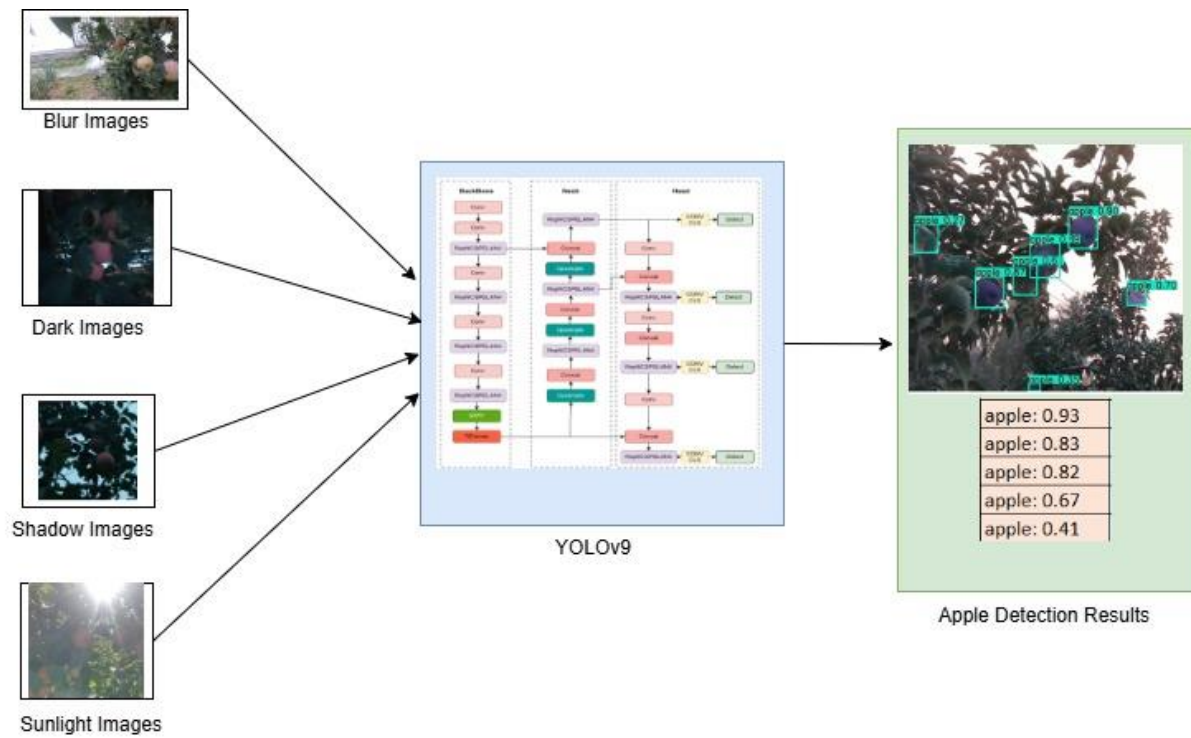
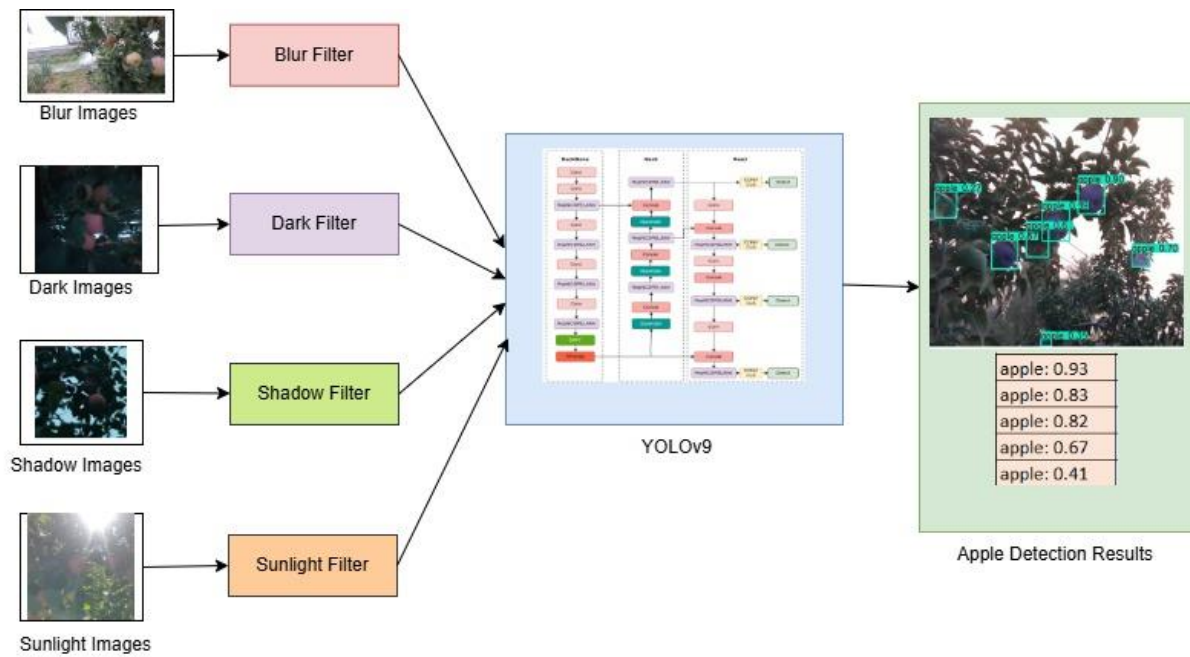


Figure 17 Images tested with only YOLOv9 for apple detection

In the second stage, the manually sorted images are passed through condition-specific filters before YOLOv9 detection. For example, images from the blur folder are processed using a blur filter, while images from the sunlight, dark, and shadow folders are processed with their corresponding filters. The filtered images are then fed into YOLOv9 for apple detection. This phase aims to demonstrate how applying the appropriate filter to each lighting condition can improve detection accuracy. The results from this stage are compared against the unfiltered detection results to analyse the impact of filters on enhancing visibility and feature clarity. The process is explained in Figure 18.



*Figure 18 Images tested with Filters and YOLOv9 for apple detection*

The final stage represents a complete integration of the SVM classifier with YOLOv9 and filters, mimicking an automated real-world detection system. Test images are first passed through the trained SVM classification model, which assigns a label corresponding to one of the four lighting conditions: blur, sunlight, dark, or shadow. The appropriate filter is automatically applied to the image based on the assigned label. For example, if the SVM classifies an image as dark, the dark filter is applied to enhance contrast and visibility. The filtered image is then sent to the YOLOv9 model for apple detection. This integrated approach simulates the end-to-end system design, demonstrating the practical application of SVM-based classification and filter selection in improving YOLOv9's detection accuracy under varying lighting conditions. The final process is depicted in Figure 19.

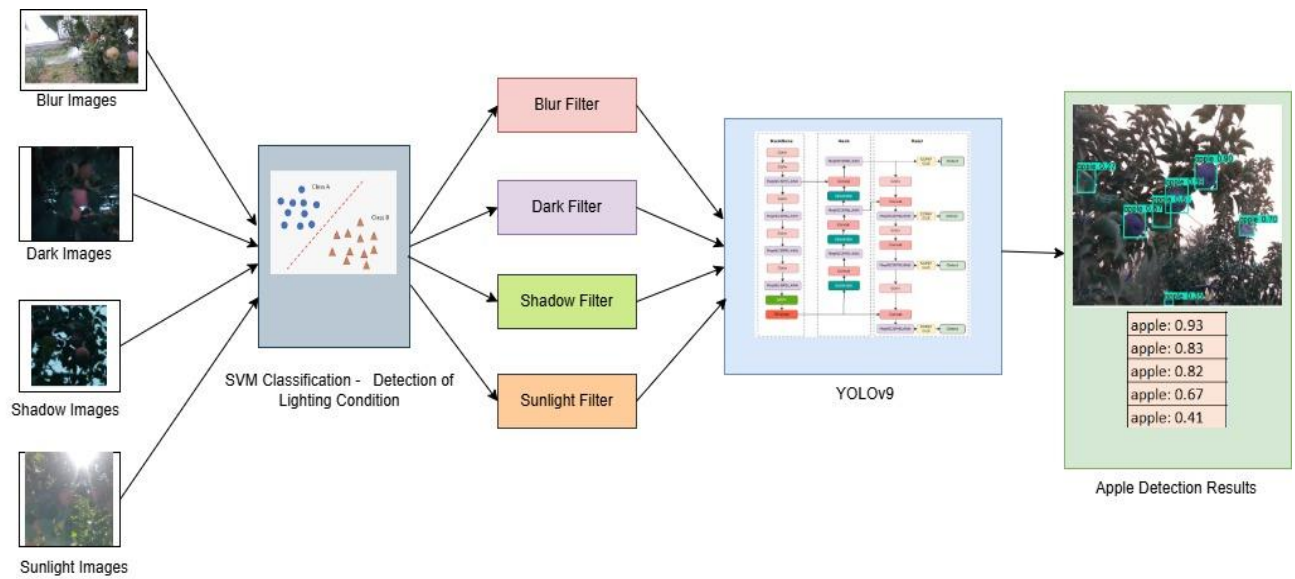


Figure 19 Images passed to trained SVM, Filter Selection and Apple detection using YOLOv9.

## 4.2 System Environment Setup

### 4.2.1 Software Environment

The apple detection model was developed and trained using Python programming language and VS Code. This setup offers several benefits, making it a preferred choice for data science and machine learning projects. Each tool is critical in image preprocessing, model development, training, and evaluation. Table 4-1 lists the software components used for the implementation.

Tool / Library	Purpose
Python	The programming language used for the implementation includes data processing, machine learning, and object detection tasks.
VS code	An Integrated Development Environment (IDE) is used for writing, debugging, and managing the Python code.
NumPy	Numerical computing, including efficient manipulation of arrays and matrices, is

	essential for processing image data for machine learning models.
sci-kit	They are used for implementing the Support Vector Machine (SVM) model, tuning hyperparameters, and computing performance metrics like accuracy and recall.
OpenCV	For image processing tasks such as loading, resizing, and applying filters to images. It also facilitates visualising the results of preprocessing.
PIL (Pillow)	They are used for primary image handling tasks like loading and resizing images to prepare them for feature extraction and model input.
Ultralytics YOLOv9	Implement and train the YOLOv9 object detection model, manage model training and validation, and evaluate detection performance.

*Table 2 Software Components Used in Implementation*

#### 4.2.2 Hardware Specifications

The experimental computations were performed on a desktop workstation equipped with the following hardware specifications:

- CPU: 11th Generation Intel Core i7-1165G7 (8 cores, 2.8 GHz base frequency)
- GPU: Intel® Iris® Xe Graphics (128MB VRAM)
- RAM: 16 GB DDR4
- Storage: 256GB NVMe SSD

While not exceptionally powerful, this hardware setup provides adequate computational resources for training this research's SVM and object detection models. This approach aligns

to create practical and scalable solutions for real-world agricultural applications where resource constraints are expected.

### 4.3 Data Collection and Processing

The experiment setup begins with collecting and processing a diverse dataset to ensure the robustness of the apple detection model under varying lighting conditions. This section outlines the steps taken for collecting images in real-world apple orchard environments and the subsequent processing techniques to prepare the dataset for practical training and evaluation of the machine learning models.

#### 4.3.1 Dataset Description

Two datasets were used in this study (Li et al., 2023). Two orchards in Beijing, China's Changping and Haidian districts, were the sites of the first dataset collection. A crawler-type orchard mobility robot was used to gather the photographs, which were taken using a Realsense D435i RGBD camera in two conventional orchards in Beijing at various times of the day. The collected dataset includes bounding boxes and 1000 images with over 12k instances. The dataset for this research will be sourced from real-world apple orchards and will comprise approximately 4000 images. These images capture various scenarios, ensuring the model is trained under diverse real-world conditions typical of apple orchards.





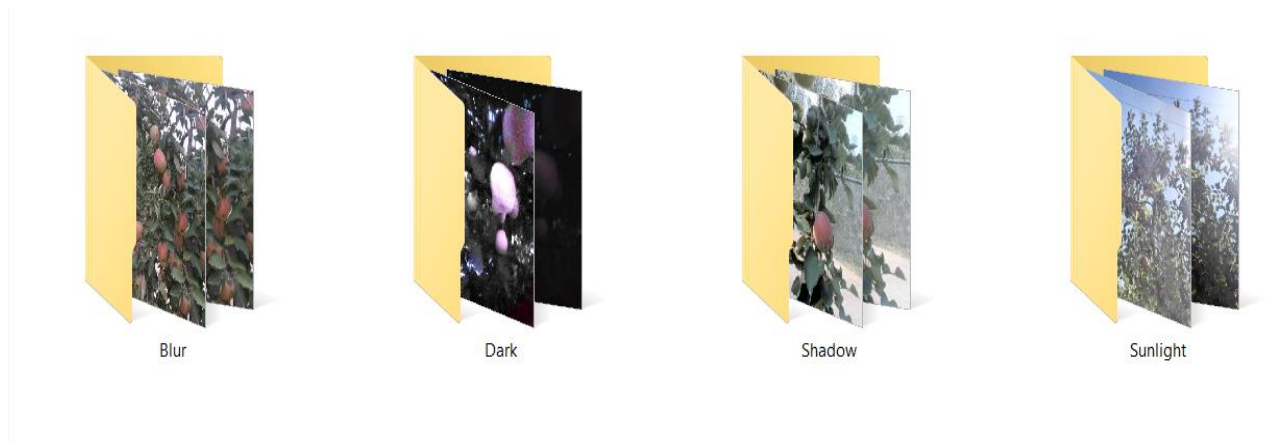
*Figure 20 Sample Images from the dataset*

The dataset used in this study was collected from a real apple orchard, consisting of 4K images captured under varying lighting conditions. The images were categorized into four groups through meticulous observation and manual sorting: blur (representing noise), sunlight, shadow, and darkness (different lighting conditions). This categorisation ensured that the images were organised according to the lighting challenges they presented, enabling the subsequent application of appropriate filters for enhancing the image quality—sample images from the dataset with different lighting conditions depicted in Figure 20.

#### **4.3.2 Dataset Categorization (Manual sorting into sunlight, shadow, dark and blur)**

The dataset used in this study was collected from a real apple orchard, consisting of 4K images captured under varying lighting conditions. The images were categorized into four groups through meticulous observation and manual sorting: blur (representing noise), sunlight, shadow, and darkness (different lighting conditions). This categorisation ensured that the images were organised according to the lighting challenges they presented, enabling the subsequent application of appropriate filters to enhance image quality.

The images are organised in the following folder structure for the training, validation and testing phases. Figure 21 shows the different folders that contain images with varying lighting conditions. Figures 22, 23, 24, and 25 show the sample images from these folders, respectively.



*Figure 21 Folder Structure*



*Figure 22 Sample Images from the Blur Folder*



*Figure 23 Sample Images from the Dark Folder*



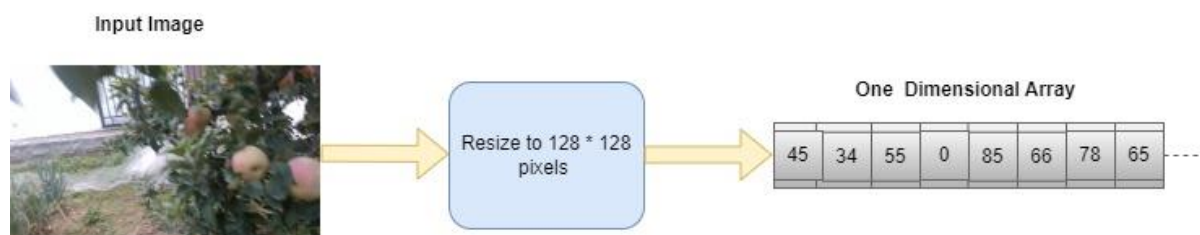
*Figure 24 Sample Images from the Shadow Folder*



*Figure 25 Sample Images from the Sunlight Folder*

### **4.3.3 Dataset Split and Feature Extraction (Train, Validation, Test)**

Following classification, the dataset was divided into 70% training, 20% validation, and 10% testing sets. The Support Vector Machine model was trained using the training set. The photos were flattened and resized before feature extraction to standardise the input data. To act as input vectors for the SVM model, images were first reduced to 128x128 pixels and then flattened into a one-dimensional array. The pixel intensity values are used as input vectors to the training model. This process is explained in Figure 26.



*Figure 26 Converting input image into a one-dimensional feature array*

After training the SVM model, it was validated using the validation set, ensuring its performance was optimised before testing. The testing phase involved applying the trained SVM model to the test set to evaluate its accuracy and robustness in detecting lighting conditions. The trained model, once validated, was saved for future inference.

#### **4.4 SVM configuration and parameters**

The Support Vector Machine (SVM) model's configuration and fine-tuning are covered in this section, with particular attention to how the model is used to classify lighting conditions in apple orchard photographs. The SVM is crucial since it guarantees that each image is appropriately pre-processed and optimises the performance of the YOLOv9 detection model that comes next. We discuss the rationale behind the kernel selection, how to tweak hyperparameters to maximise model performance, and how regularisation maintains the balance between accuracy and generalisation. In order to adapt to various lighting conditions and ensure accurate apple detection in real-world scenarios.

##### **4.4.1 Kernel Selection**

The Radial Basis Function (RBF) kernel is good at tackling non-linear relationship problems often brought on by the lighting variations in orchard photos. The Support Vector Machine model was constructed utilising it. The dataset in question has a range of lighting conditions that result in non-linear class borders, such as shadow, blur, sunlight, and darkness. The RBF kernel may effectively transfer these input qualities into a higher-dimensional space, facilitating the SVM's search for a dividing hyperplane across different classes.

The RBF kernel significantly benefits over a linear kernel since it may capture non-linear relationships without explicitly defining feature changes. When there are intricate, non-linear interactions in the lighting conditions, the RBF kernel performs better in accuracy and generalisation regarding apple detection. On the other hand, the linear kernel performs better in scenarios where classes can be identified linearly, which is rare in agricultural images that

are impacted by a range of environmental factors. Because of the RBF kernel selection, the SVM can handle this complexity, which results in a more accurate classification under different illumination conditions.

#### **4.4.2 Hyperparameter Tuning**

Various parameter combinations were thoroughly analysed using GridSearchCV to optimise the hyperparameter tuning process. The two primary hyperparameters adjusted for this experiment were C and gamma. Parameter C. Wider margins govern the trade-off between lowering classification errors and limiting the decision boundary margin, allowing for lower values of C, which may promote generalisation but may lead to incorrect training point classification. Conversely, higher values of C could lead to overfitting because they would correctly identify each point. From 0.1 to 100, a broad range of values of C were looked at to identify the optimal equilibrium.

The parameter gamma influences the training example's effect's extent and determines the decision border's complexity. Smaller values have a more significant impact and result in more straightforward option boundaries, whilst larger values allow for more complex limits. The gamma values of scale 0.01, 0.1, 1, and 10 were examined using the grid search. The cross-validation evaluation of these parameter combinations made the model's resilience across different data splits possible. The optimal setting,  $C = 1.0$  and  $\text{gamma} = \text{scale}$ , minimised the chance of overfitting while guaranteeing that the SVM effectively identified the varied illumination scenarios typical of orchard pictures. This combination provided the ideal balance between accuracy and generalisation.

#### **4.4.3 Regularization**

Regularisation of parameter C is crucial in balancing classification accuracy with model generalisation. It was found that the optimal value for this dataset was  $C = 1.0$  since it provided an effective trade-off between maintaining a sufficiently high margin and correctly classifying the training set. This balance ensured the model could react adequately to fresh pictures while preventing overfitting. This is particularly crucial since images of orchards often feature diverse lighting conditions, requiring a model that can adapt to different conditions.

It was found that evaluating different values of C (such as 0.1) resulted in underfitting, where the decision boundary of the model was too simplistic to categorise complicated patterns, hence decreasing accuracy. On the other hand, overfitting resulted in  $C > 1.0$  (e.g., 10 or 100), as the

model became unduly focused on fitting the training data including noise which decreased performance on validation and test datasets. In order to retain acceptable classification accuracy throughout a range of lighting conditions during training and testing,  $C = 1.0$  offered a middle ground that prevented the model from being either very rigid or extremely sensitive to noise.

## **Chapter 5 Results and Discussion**

This chapter presents the results of the experiments conducted to detect apples under varying lighting conditions and thoroughly discusses their implications. The goal is to evaluate the effectiveness of the proposed methodology, which integrates lighting condition or noise classification of the input image, targeted filtering, and deep learning-based detection. Section 5.1 focuses on the performance of the SVM in detecting lighting conditions and analysing classification metrics and challenges. Section 5.2 compares the filters applied to enhance images under various lighting scenarios using qualitative and quantitative analyses. Section 5.3 details the performance of the YOLOv9 apple detection model, presenting results for each lighting condition and analysing the influence of preprocessing filters on detection accuracy. Finally, Section 5.4 discusses the key findings, challenges faced, practical implications, and potential future improvements, providing a comprehensive understanding of the experiment outcomes and their relevance to real-world automation applications.

### **5.1 SVM Validation Classification Results ( Lighting Condition Detection )**

Lighting condition categorisation is a critical step in preprocessing pipelines for Apple detection. This part evaluates how well the SVM model identifies lighting situations such as sunlight, shadow, darkness, and blur to optimise every image for the subsequent apple detection. The SVM model was trained to classify the lighting conditions in the apple orchard images into four categories: sunlight, shadow, darkness, and blur. The classification results were evaluated on both the validation and testing datasets.

During the validation phase, the SVM model achieved an accuracy of 78.38%. This accuracy indicates that the model correctly classified 78.38% of the validation images into their respective lighting conditions. The validation accuracy indicates how well the model generalises to data not seen during training. A result of 78% suggests that while the model has learned to differentiate between the lighting conditions reasonably well, there may still be some cases where it struggles to classify challenging or ambiguous conditions correctly.

Perceived Filter Condition / Filter Condition Produced	Perceived Filter Condition			
	Blur	Dark	Shadow	Sunlight
Blur	6	0	0	2
Dark	0	17	0	0
Shadow	1	1	3	0
Sunlight	3	0	1	3

*Table 3 Confusion Matrix for Four Lighting Conditions in Apple Detection using RBF Kernel-SVM.*

Total Predictions: 37

Correct Predictions: 29 (Green Colour in the above table)

Validation Accuracy:  $(29 / 37) * 100 = 78.38\%$

The confusion matrix presented in Table 3 illustrates the performance of the SVM classifier in distinguishing between four distinct lighting conditions: Blur, Dark, Shadow, and Sunlight within apple orchard images. The rows represent the actual lighting conditions, labelled as the "Filter Condition Produced," while the columns denote the predicted classifications, labelled as the "Perceived Filter Condition." Correct classifications, highlighted in green along the diagonal, indicate instances where the classifier accurately matched the perceived condition to the actual condition. For example, 17 photos labelled "Dark" were accurately identified as "Dark," and 6 photos labelled "Blur" were appropriately classified as "Blur." Misclassifications, such as instances where images labelled "Blur" were incorrectly classified under "Sunlight," appear outside the diagonal. The matrix reveals a high accuracy rate, with 29 out of 37 images correctly classified, yielding a validation accuracy of 78.38%. This evaluation method is vital for assessing the classifier's effectiveness in realistic scenarios where

lighting variations impact detection accuracy, underscoring the robustness of the model in differentiating lighting conditions essential for optimal apple detection.

<b>Perceived Filter Condition</b> <b>Filter Condition Produced</b>				
	Blur	Dark	Shadow	Sunlight
Blur	5	0	1	2
Dark	0	17	0	0
Shadow	2	1	2	1
Sunlight	3	0	1	3

*Table 4 Confusion Matrix for Four Lighting Conditions in Apple Detection using Linear Kernel-SVM.*

Total Predictions: 37

Correct Predictions: 27 (Green Colour in the above table)

Validation Accuracy:  $(27 / 37) * 100 = 72.38\%$

The confusion matrix in Table 4 presents the performance of the SVM classifier with a Linear Kernel for classifying four lighting conditions: Blur, Dark, Shadow, and Sunlight in orchard images intended for apple detection. The rows indicate the actual lighting conditions under the “Filter Condition Produced,” while the columns represent the predicted classifications under the “Perceived Filter Condition.” Correct classifications, marked in green along the diagonal,

show instances where the perceived condition accurately matches the actual condition, such as 5 correctly classified “Blur” images and 17 correctly identified “Dark” images. Misclassifications are represented outside the diagonal, highlighting areas where the classifier’s predictions diverged from the actual lighting conditions, such as cases where “Shadow” images were misclassified as “Sunlight.” The classifier achieved 27 correct predictions out of 37, resulting in % validation accuracy of 72.38%. This matrix serves as a crucial diagnostic tool to evaluate the effectiveness of the Linear Kernel-SVM in distinguishing complex lighting scenarios, which is essential for enhancing detection accuracy in real-world orchard settings.

## **5.2 SVM Testing Classification Results (Lighting Condition Detection)**

The SVM model was evaluated on the test dataset consisting of images taken under four lighting conditions: Blur, Dark, Shadow, and Sunlight. The testing process involved loading images from the specified test folders and processing them to generate predictions for each class. The SVM model predicted the lighting condition for each test image, which was then compared to the ground truth labels to determine its accuracy.

## Confusion Matrix

Given the 100% accuracy, the confusion matrix would show zero off-diagonal values, indicating no misclassifications. All images were correctly classified into their respective categories, with all predicted values aligning perfectly with the proper labels. The figure below shows the confusion matrix.

<b>Perceived Filter Condition</b> <b>Filter Condition Produced</b>		Blur	Dark	Shadow	Sunlight
Blur	3	0	0	0	
Dark	0	8	0	0	
Shadow	0	0	2	0	
Sunlight	0	0	0	3	

*Table 5 Confusion Matrix for Four Lighting Conditions in Apple Detection using RBF Kernel-SVM.*

Total Predictions: 16

Correct Predictions: 16 ( Green Colour in the above table )

Validation Accuracy:  $(16 / 16) * 100 = 100\%$

The "Perceived Filter Condition" in this confusion matrix represents the actual classification categories (Blur, Dark, Shadow, and Sunlight) the model aims to predict. In contrast, the "Filter Condition Produced" reflects the predicted categories assigned by the SVM classifier. The diagonal cells highlighted in green indicate correctly classified conditions where the predicted

filter condition matches the actual condition. All predictions are correct, resulting in a perfect classification for each lighting condition.

Out of the 16 total predictions, all 16 are accurate, yielding a validation accuracy of 100%. This high level of accuracy suggests that the RBF Kernel-SVM model is highly effective in distinguishing between the four lighting conditions in this dataset. The results demonstrate the model's robustness in categorising each lighting condition accurately, particularly in challenging orchard environments where lighting can vary widely. This perfect classification underscores the SVM model's capacity to adaptively preprocess images, setting a solid foundation for subsequent apple detection tasks.

<b>Perceived Filter Condition</b> <b>Filter Condition Produced</b>	Blur	Dark	Shadow	Sunlight
Blur	3	0	0	0
Dark	0	8	0	0
Shadow	1	0	0	1
Sunlight	0	0	0	3

*Table 6 Confusion Matrix for Four Lighting Conditions in Apple Detection using Linear Kernel-SVM.*

Total Predictions: 16

Correct Predictions: 14 ( Green Colour in the above table )

Validation Accuracy:  $(14 / 16) * 100 = 87.5\%$

This confusion matrix in Table 6 demonstrates the performance of a linear kernel-SVM model in classifying images under four distinct lighting conditions: Blur, Dark, Shadow, and Sunlight. The rows represent the actual lighting conditions as captured in the images, while the columns represent the predicted lighting conditions as classified by the SVM model. Each cell in the matrix indicates the count of images corresponding to a particular actual vs. predicted condition. The diagonal entries highlighted in green indicate correct predictions, where the perceived lighting condition matches the filter condition produced by the model. In this case, out of 16 predictions, the model correctly classified 14, achieving a validation accuracy of 87.5%. This matrix accurately illustrates the model's strengths in identifying Blur, Dark, and Sunlight conditions while showing some misclassification in the Shadow category.

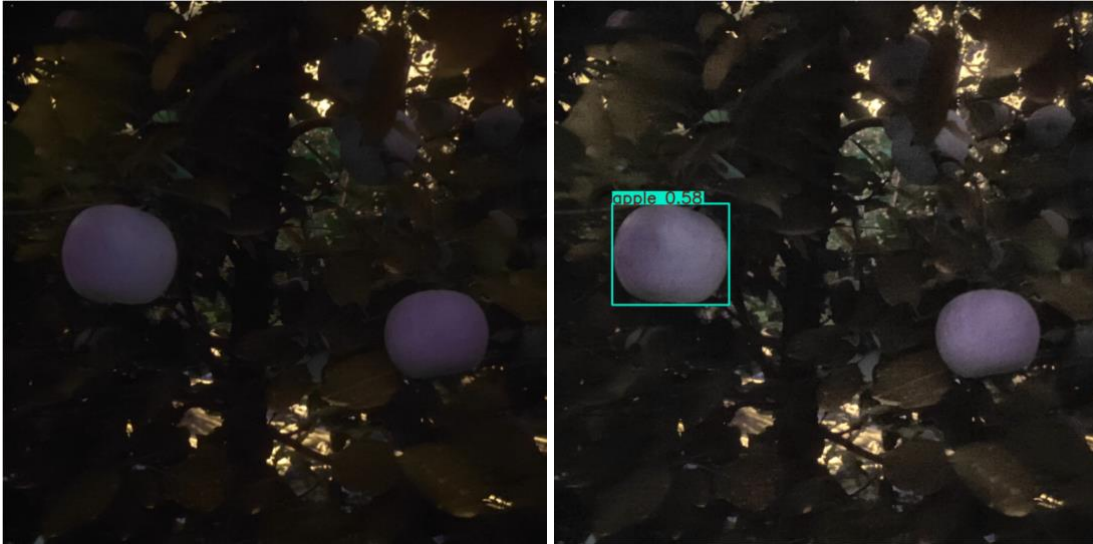
The achieved 100% accuracy suggests that the SVM model with RBF kernel was very effective at distinguishing between the different lighting conditions presented in the dataset. However, this result may indicate that the testing set was either less complex or similar to the training set, potentially leading to overfitting. While this performance is desirable, further validating the model on more diverse datasets is crucial to ensure it generalises well under varying real-world conditions. Overall, the SVM classification model demonstrated excellent performance in detecting lighting conditions in orchard images, as evidenced by perfect accuracy across all categories.

### 5.3 Performance Analysis of Filters

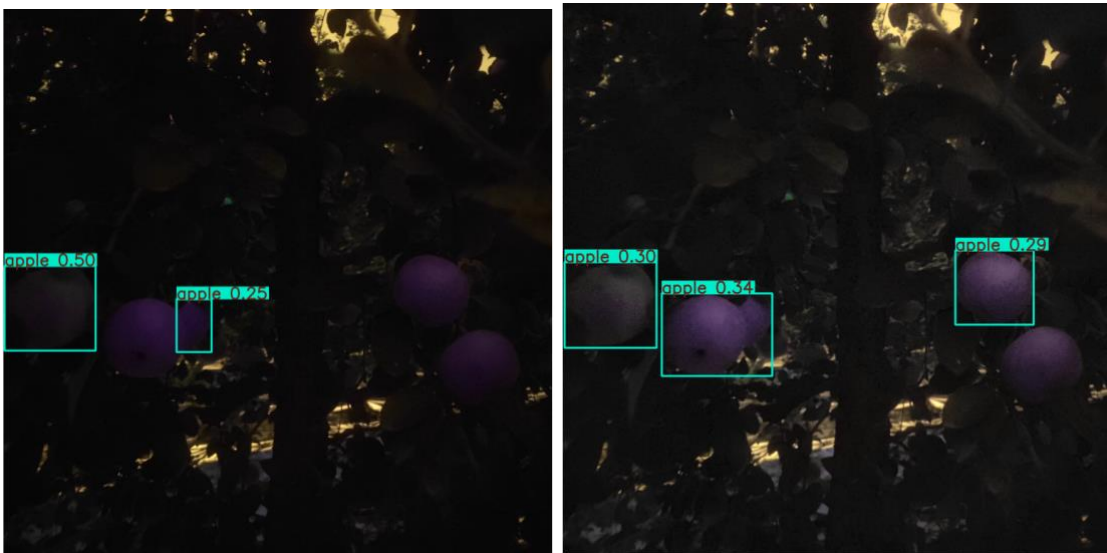
The final phase evaluates apple detection performance under varying lighting conditions using the YOLOv9 model. After applying the appropriate filters based on lighting condition classification, the images were analysed to determine how effectively apples improved detection accuracy and the number of apples detected in challenging scenarios such as blur, darkness, shadow, and sunlight. This section presents the results of the detection process. It assesses the impact of the preprocessing steps on detection accuracy, providing insights into the model's robustness and the effectiveness of the applied filters.

#### 5.3.1 Apple Detection results with and without filter

**Dark:** The images in Figures 27,28 and 29 illustrate the impact of the filter in enhancing visibility under dark conditions with yolov9.



*Figure 27 Apple detection results before and after dark filter*



*Figure 28 Apple detection results before and after dark filter*

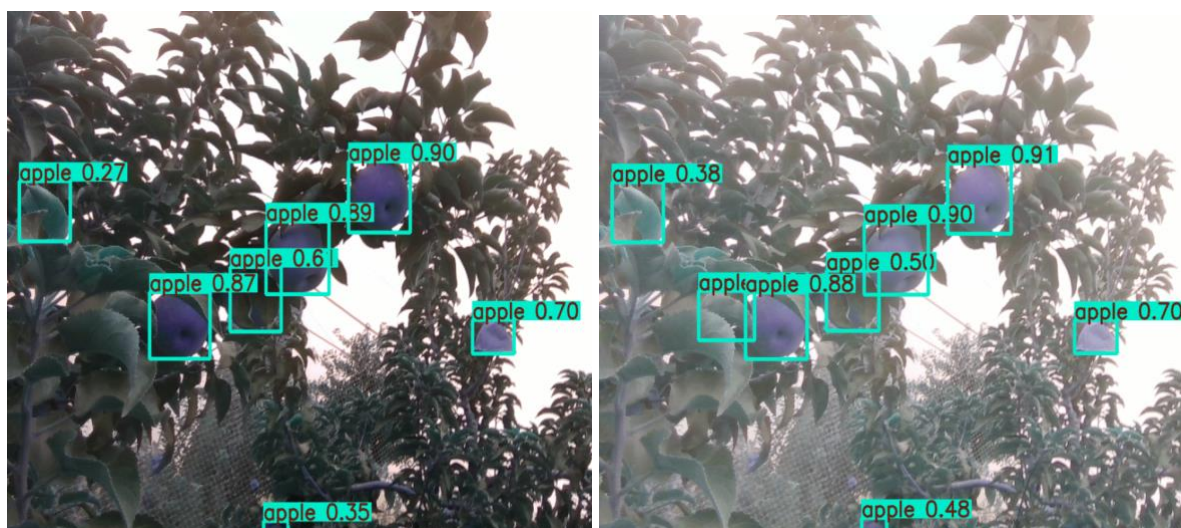


*Figure 29 Apple detection results before and after dark filter*

**Shadow:** The images shown in Figure 30 and Figure 31 demonstrate the improvements in visibility achieved for shadowed areas after applying the shadow filter with yolov9 object detection.

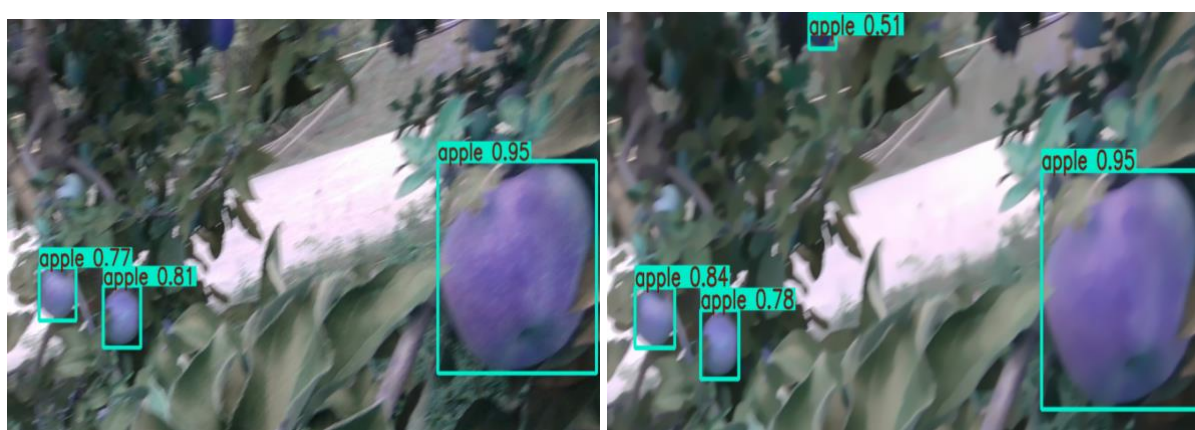


*Figure 30 Apple detection results before and after shadow filter*



*Figure 31 Apple detection results before and after shadow filter*

**Blur:** The images shown in Figure 32 and Figure 33 show the effect of the blur filter in sharpening and enhancing blurred images.

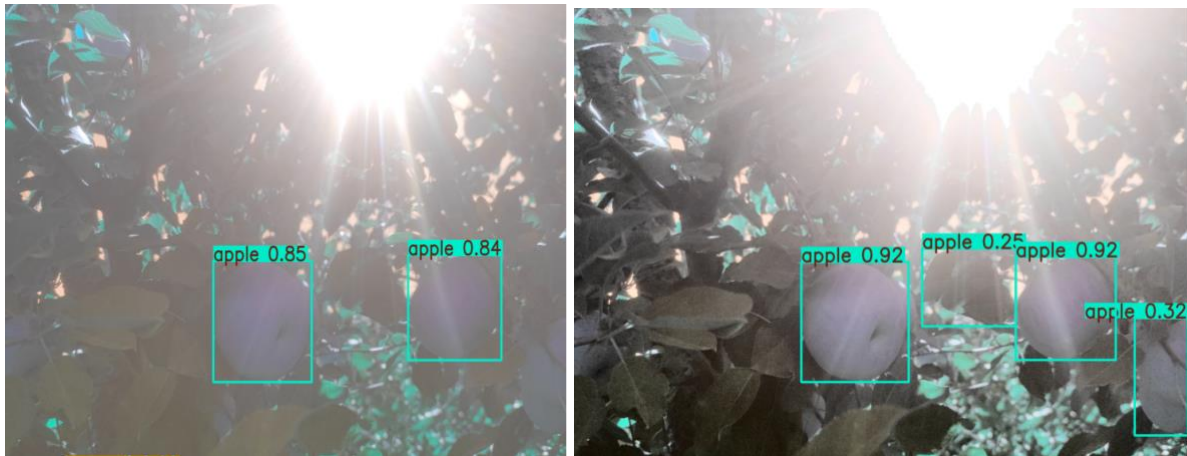


*Figure 32 Apple detection results before and after the blur filter*

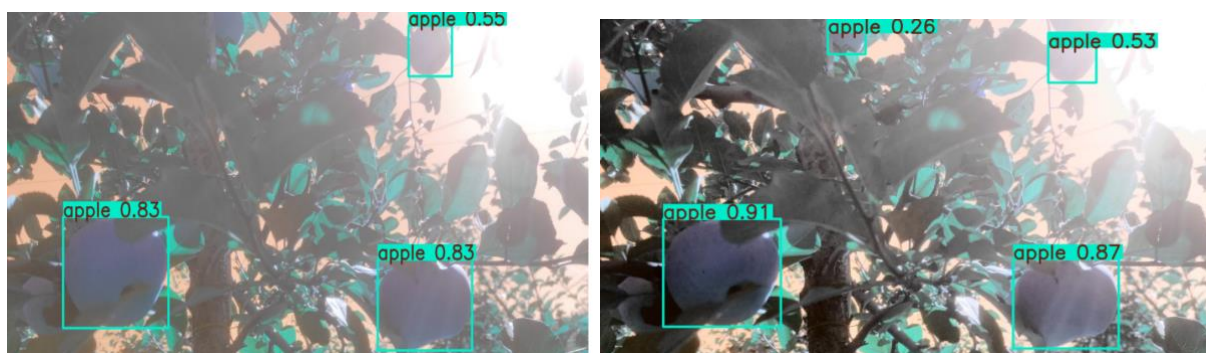


*Figure 33 Apple detection results before and after the blur filter*

**Sunlight:** Figures 34 and 35 depict the reduction of glare and recovery of details in images affected by strong sunlight.



*Figure 34 Apple detection results before and after sunlight filter*



*Figure 35 Apple detection results before and after sunlight filter*

### 5.3.2 Filter Performance Results

To demonstrate how image preprocessing increases detection accuracy, performance metrics, such as accuracy before and after applying the filter, are compared for each filter. The primary conclusions are presented in the tables in this section, each of which includes a supplemental table showing the enhancement procedure's effectiveness. The accuracy figures are an essential indicator that shows the model's ability to recognise apples in challenging circumstances before and after filtering. These tables provide a detailed insight into how each filter impacts the model's overall performance, demonstrating the significance of preprocessing in enhancing detection results.

The findings demonstrate that significant gains in detection accuracy were seen when suitable filters were used, which improved the filtered image's suitability for the YOLOv9 detection model. These comparisons are shown in the consolidated tables, which provide an easy-to-understand overview of the variations in performance for every lighting scenario and filter combination. The accuracy measurements reveal which filters improved the most and under what circumstances they performed best.

Lightning Condition Filter	Apple Detection Accuracy Improvement	Number of Apples Detected Improvement
Shadow	23.82%	16.67%
Blur	16.75%	25%
Sunlight	<b>36.68%</b>	<b>43.75%</b>
Dark	<b>47.10%</b>	<b>33.33%</b>

*Table 7 Performance of Filters on Apple Detection - Detection accuracy and number of apples detected improvement*

The testing results in Table 7 demonstrate that applying filters to the photographs significantly enhanced YOLOv9's apple detection performance. This improvement is evident in the increased detection accuracy across various lighting conditions. According to the results, the dark filter had the highest impact, with a substantial 47.10% increase in detection accuracy. This suggests that enhancing visibility in low-light photographs is crucial for the model to identify objects more accurately. The sunlight filter also contributed significantly, improving detection accuracy by 36.68%, as it effectively reduced glare and restored details previously lost due to overexposure. The shadow filter showed a 23.82% increase in accuracy, demonstrating its effectiveness in dealing with challenging lighting conditions involving partial shadows. Meanwhile, the blur filter showed a modest improvement in detection accuracy of 16.75%, indicating that addressing blur alone may not be as impactful as adjusting lighting-related conditions.

In addition to improving detection accuracy, the filters positively impacted the number of apples detected. The sunlight filter led to the highest increase in the number of apples detected, with an improvement of 43.75%, highlighting its ability to mitigate glare and reveal more details in high-light conditions. The dark filter followed closely, achieving a 33.33% increase in apple count, reinforcing the importance of enhanced visibility in low-light scenarios. The shadow filter showed a 16.67% improvement, while the blur filter, although having a minor effect on accuracy, increased the number of apples detected by 25%. This indicates that each filter uniquely improves the model's detection capabilities under specific conditions.

Applying these filters significantly improved detection accuracy and the number of apples detected. Despite their simplicity, these filters enhanced detection outcomes by targeting specific lighting challenges, underscoring the need for effective preprocessing in orchard settings where illumination variability is a significant obstacle. These results highlight the potential for apple detection algorithms to achieve higher performance when carefully selected filters are applied based on the prevailing lighting conditions.

#### 5.4 SVM classification and YOLOv9 apple detection results with Filters

This section presents the results of the integrated SVM classification, Filters and Yolov9 for objection. The test images will use the trained SVM to detect the image's lighting or noise condition. Then, based on the condition detected, the appropriate filter is applied to the input image for pre-processing to enhance the image. After the image is pre-processed, the image is passed to the yolov9 object detection method to detect the apples with the bounding boxes.

EXPERIMENT RESULTS (Number of apples detected)						
Object Detection Lighting Condition Filter	YOLOv9	YOLOv9 + Dark Filter	YOLOv9 + Blur Filter	YOLOv9 + Sunlight Filter	YOLOv9 + Shadow Filter	SVM Condition Detection + Filter + YOLOv9
Shadow	6				7	7
Dark	6	8				8
Sunlight	37			43		43
Blur	17		24			24
<b>(Apple Count )</b>	<b>66</b>					<b>81</b>

Table 8 Integrated results – SVM, Filters and YOLOv9

Number of apples detected only with YOLOv9 = 66 apples (Blue colour in the table)

Number of apples detected with SVM classification + Filter + YOLOv9 = 81 apples (Purple Colour in the table)

Total improvement in the number of apples detected = 22.73%

The test dataset used in this study included a limited number of images, suggesting that a larger dataset would likely lead to further improvements in apple detections. Adopting this simple yet effective classification and filtering approach enhances apple detection performance. As shown in Table 8, the combined results demonstrate that 81 apples were detected across 16 test images when using SVM classification with appropriate filters and YOLOv9, compared to only 66 apples detected without filter application. This represents an approximate 22.73%

improvement in the number of apples detected. Interestingly, in one of the sunlight images, the number of apples detected with filtering was slightly lower than without, indicating that while filtering improves detection, certain lighting conditions might yield mixed results. However, the accuracy of individual apple detections was noticeably higher across filtered images, suggesting that filters are especially beneficial in challenging lighting conditions. This improvement trend was consistently observed across various lighting scenarios, underscoring the value of targeted preprocessing in enhancing detection reliability.

Lightning Condition Filter	Apple Detection Accuracy Improvement	Number of Apples Detected Improvement
Shadow	<b>29.37%</b>	16.67%
Blur	14.8%	12.22%
Sunlight	<b>33.33%</b>	<b>41.9%</b>
Dark	<b>38.46%</b>	<b>29.10%</b>

*Table 9 Integrated results with test data – SVM, Filters and YOLOv9 – Apple Detection accuracy and number of apples detected improvements.*

The testing results demonstrate that applying filters to the photographs significantly enhanced YOLOv9 apple detection performance. This improvement is evident in the increased detection accuracy across various lighting conditions. According to the results, the dark filter had the highest impact, with a substantial 38.46% increase in detection accuracy. This finding underscores the importance of improving visibility in low-light photographs, allowing the model to detect objects more accurately. The sunlight filter improved considerably, boosting detection accuracy by 33.33%, effectively reducing glare and restoring overexposed details. The shadow filter showed a notable increase in accuracy of 29.37%, proving beneficial in handling partial shadow conditions. Meanwhile, the blur filter showed a more modest improvement of 14.8% in detection accuracy, suggesting that reducing blur alone might have a limited impact compared to other lighting adjustments.

In addition to enhancing detection accuracy, the filters positively influenced the number of apples detected. The sunlight filter achieved the highest increase in apple count, with an improvement of 41.9%, indicating its effectiveness in managing glare and enhancing detail in bright conditions. The dark filter followed with a 29.10% increase, highlighting the significance of improved visibility in low-light scenarios. The shadow filter resulted in a 16.67% increase in apple detection, while the blur filter, though less impactful, still managed to increase the apple count by 12.22%. Each filter uniquely enhances the model's detection capability under specific lighting conditions.

Applying these filters led to significant improvements in detection accuracy and the number of apples detected. Despite their straightforward approach, these filters successfully addressed specific lighting challenges, reinforcing the value of preprocessing in environments with variable illumination. These findings suggest that apple detection algorithms can achieve higher performance by applying carefully selected filters tailored to the prevailing lighting conditions in orchard settings.

## 5.5 Findings

This section presents the key findings of the research, categorised into image classification, the impact of pre-processing filters, and the performance of the integrated detection system.

The Support Vector Machine classifier successfully categorised images into four lighting or noise conditions: blur, sunlight, dark, and shadow, with an accuracy of 100%. This high accuracy is likely due to the distinct pixel value patterns of the dataset, which facilitated straightforward classification. A non-linear Radial Basis Function RBF kernel also improved the classifier's effectiveness. However, this accuracy might partially reflect the dataset's limited diversity.

Detection Accuracy:

- The dark filter achieved the highest improvement with a 47.10% increase in accuracy, enhancing brightness and making apple features more distinguishable.
- The sunlight filter improved accuracy by 36.68%, effectively reducing glare and restoring overexposed details.
- The shadow filter yielded a 23.82% improvement, demonstrating its utility in handling partial shadows.

- The blur filter showed a modest improvement of 16.75%, indicating the need for more advanced algorithms to address blurry images effectively.

#### Number of Apples Detected:

- The sunlight filter led to the biggest improvement in the number of apples detected, increasing by 43.75%.
- The dark filter contributed a 33.33% improvement.
- The blur filter increased detections by 25%, while the shadow filter improved counts by 16.67%.

These results emphasise that filters tailored to specific lighting conditions enhance detection accuracy and the model's ability to detect more apples under challenging conditions. A few more findings are below:

- In some instances, filters improved the number of apples detected but reduced detection accuracy, suggesting that individual detections may have been less precise, but more were identified overall.
- Conversely, some filters reduced the number of apples detected but increased accuracy for individual detections, showcasing their effectiveness in refining specific object features.
- In the best scenarios, accuracy and the number of apples detected increased, demonstrating the potential of well-matched filters to optimise detection performance.

The integrated framework, combining SVM classification, condition-specific filtering, and the YOLOv9 object detection model, showed a 23% overall improvement in apple detection compared to YOLOv9 alone. This structured approach significantly reduced missed detections, even under challenging lighting conditions, highlighting its utility for real-world applications in apple harvesting. The results validate the value of an integrated method combining classification, filtering, and detection to enhance detection outcomes.

#### Key Observations:

- The SVM classifier's ability to accurately classify images based on lighting conditions offers a robust preprocessing step for agricultural robotics, enabling real-time adaptability to varying environmental conditions.
- The applied filters demonstrated considerable efficacy in improving detection results, particularly for dark images, by enhancing visibility and brightness.

- While basic, the filters addressed critical challenges like glare, shadow, and low light but showed limited success in handling blur, indicating the need for advanced noise-reduction algorithms.
- Without pre-processing, object detection would suffer from increased detection failures and inaccuracies, emphasising the importance of adaptive filtering for robust performance.
- This novel approach has not been widely explored in computer vision, particularly in agricultural robotics, making it a significant contribution to the field.

The findings demonstrate that the integration of SVM classification, tailored pre-processing filters, and the YOLOv9 detection model provides a scalable and effective solution for apple detection under diverse lighting and noise conditions. This approach enhances both detection accuracy and object count, significantly reducing failures. Future work should expand the dataset for broader applicability, develop advanced filters to handle complex noise and optimise the system for commercial implementation in agricultural robotics.

### **5.6 Computational Complexity vs. Accuracy Trade-off**

The accuracy of apple detection under various lighting situations is greatly increased by the incorporation of preprocessing filters. Nevertheless, this improvement adds a computational expense that needs to be carefully taken into account for real-time applications. SVM classification for lighting condition detection is one of the several phases in the suggested pipeline. Next, image preprocessing filters are used before the image is sent into the YOLOv9 object detection model. The viability of real-time deployment may be impacted by each of these steps' increased processing times, particularly in contexts with limited resources like embedded devices.

Depending on the number of training samples, the SVM classifier used for lighting condition detection has a complexity ranging from  $O(n^2)$  to  $O(n^3)$  in the worst-case situations. SVM adds latency when handling real-time inputs or processing huge datasets, even though it guarantees accurate classification. Computational overhead is further increased by the need for pixel-wise changes for picture preprocessing filters like gamma correction, histogram equalisation, and deblurring. Applying preprocessing filters might be a limiting factor in real-time detection applications because it typically results in an increase of 10 ms per image in processing time.

Despite the additional computational demand, the benefits in detection accuracy are significant. The filtering process improves apple detection rates, achieving an accuracy increase of 47.1% in dark conditions and 36.68% in high-glare sunlight conditions. However, for real-time deployment, it is crucial to balance accuracy improvements with computational efficiency. One potential optimization strategy is to selectively apply preprocessing filters only when necessary, rather than processing all images uniformly. This could be achieved by setting a confidence threshold, allowing high-quality images to bypass unnecessary filtering. Additionally, GPU-accelerated parallel processing and edge computing techniques can significantly reduce processing time while maintaining detection performance. Future research should explore adaptive filtering techniques that dynamically adjust preprocessing intensity based on input image characteristics, ensuring an optimal balance between accuracy and real-time efficiency.

## Chapter 6 Conclusion and Future Research

This study aimed to create a reliable apple-detecting system that could adjust to different illumination levels in actual orchard settings. We suggested an integrated method that combines SVM classification, custom image filters, and YOLOv9 object identification to address the difficulties of uneven lighting, including direct sunshine, shadows, low light, and blur from movement. The objective was to build a comprehensive pipeline capable of accurately detecting apples by dynamically adjusting preprocessing filters based on the detected lighting conditions.

The experiment was conducted in three stages, each building upon the previous to demonstrate the progressive effectiveness of the system. Initially, images were processed with YOLOv9 without filters to establish a baseline detection rate. This resulted in a detection count of 66 apples, revealing significant limitations when lighting conditions were suboptimal. In the second stage, lighting-based filters were manually applied to images (e.g., a blur filter to blurred images), followed by YOLOv9 detection. This approach improved detection by enhancing image quality, allowing YOLOv9 to detect finer details previously obscured. However, it remained a manual process, highlighting the need for an automated solution. In the final stage, we trained an SVM classifier to automatically categorise images into predefined lighting conditions (Blur, Sunlight, Dark, and Shadow). The classifier's output triggered the selection of an appropriate filter before passing the image to YOLOv9 for apple detection. This integrated pipeline achieved a detection count of 81 apples, reflecting a 22.73% improvement over the baseline, thus underscoring the effectiveness of the proposed approach.

In summary, this thesis demonstrates the feasibility and effectiveness of an integrated SVM-Filter-YOLOv9 pipeline for apple detection under varying lighting conditions. The proposed approach significantly improves detection rates by adaptively preprocessing images based on their lighting conditions. The findings suggest that combining classification and filtering with advanced object detection models like YOLOv9 can enhance detection accuracy in agricultural environments where unpredictable lighting variations are expected. This research contributes a foundational step towards more reliable and automated fruit detection systems, essential for developing efficient robotic harvesting technologies.

This detection pipeline can be further optimised and expanded by addressing the limitations identified and implementing the proposed future work. As a result, it holds the potential to

serve as a robust framework for broader applications within agricultural automation, supporting sustainable farming practices and advancing the field of precision agriculture.

## 6.1 Future Works

While the proposed system demonstrates substantial improvements in apple detection under varying lighting conditions, several limitations suggest opportunities for future enhancements. The SVM classifier, currently limited to detecting four distinct lighting conditions—blur, sunlight, dark, and shadow—reduces adaptability in real-world scenarios. Orchard environments often experience more nuanced lighting variations, such as partial cloud cover, reflections, or artificial light interference. Expanding the classifier's capabilities to address these conditions could significantly improve robustness and versatility. Additionally, static predefined filters tailored to specific lighting conditions limit real-time adaptability. These filters, although practical, lack dynamic adjustment based on image quality. Future research could explore machine learning-based adaptive filtering techniques, such as reinforcement learning, to optimise filter parameters and enhance preprocessing effectiveness in real-time.

Another limitation lies in the specificity of the training dataset, which focuses on a particular orchard environment, capturing specific lighting conditions and apple varieties. This restricts the system's generalizability to diverse agricultural contexts. Broadening the dataset to include varied orchard settings, seasonal changes, and apple species could enhance the detection model's applicability across crops and geographic regions. Furthermore, the system's performance is tied to the YOLOv9 detection model, effectively balancing speed and accuracy. However, alternative models like EfficientDet or RetinaNet may offer superior detection accuracy or computational efficiency. Comparative studies evaluating these models under challenging conditions could inform optimal model selection for specific agricultural applications. Addressing these limitations will enhance the system's adaptability, accuracy, and usability in real-world agricultural environments.

In order to lower computing overhead and preserve detection accuracy, future research should concentrate on improving the effectiveness of preprocessing techniques. Using convolutional neural networks (CNNs) or reinforcement learning (RL) to create an adaptive filtering mechanism that dynamically modifies filter parameters according to image quality is one method. An RL-based system could learn to selectively enhance just the photos that need rectification, balancing processing speed and accuracy instead of applying preprocessing uniformly to all images. The system could be made viable for real-time applications in

autonomous fruit-picking robots by utilising hardware acceleration techniques like CUDA-based parallel processing and edge computing on platforms like NVIDIA Jetson or Google Edge TPU, which could drastically cut down on inference time.

The model's resilience and capacity for generalisation will be further enhanced by enlarging the dataset to encompass a wider range of lighting conditions, orchard settings, and fruit types. Four lighting conditions—sunlight, shadow, dark, and blur—are the main focus of current investigations, however partial occlusions, seasonal fluctuations, artificial lighting, and mixed illumination effects are all present in real-world situations. The model's flexibility may be increased by including domain adaptation strategies that make use of Generative Adversarial Networks (GANs) or synthetic data augmentation. Furthermore, YOLOv9's comparison with other cutting-edge object detection models, such EfficientDet, RetinaNet, and transformer-based detectors, may highlight different designs that are more appropriate for low-power, embedded implementations.

## Appendix

### SVM Test results using RBF kernel in VS code.

```

Loading from folder: C:\project\New_Dataset\images\Blur\test
Processing folder: C:\project\New_Dataset\images\Blur\test
Loading from folder: C:\project\New_Dataset\images\Dark\test
Processing folder: C:\project\New_Dataset\images\Dark\test
Loading from folder: C:\project\New_Dataset\images\Shadow\test
Processing folder: C:\project\New_Dataset\images\Shadow\test
Loading from folder: C:\project\New_Dataset\images\Sunlight\test
Processing folder: C:\project\New_Dataset\images\Sunlight\test
Accuracy: 100.00%

Classification Report:

```

	precision	recall	f1-score	support
Blur	1.00	1.00	1.00	3
Dark	1.00	1.00	1.00	8
Shadow	1.00	1.00	1.00	2
Sunlight	1.00	1.00	1.00	3
accuracy			1.00	16
macro avg	1.00	1.00	1.00	16
weighted avg	1.00	1.00	1.00	16

```

Image 1 (C:\project\New_Dataset\images\Blur\test\apple (265).png): Predicted = Blur, True = Blur
Image 2 (C:\project\New_Dataset\images\Blur\test\apple (365).png): Predicted = Blur, True = Blur
Image 3 (C:\project\New_Dataset\images\Blur\test\apple (8).png): Predicted = Blur, True = Blur
Image 4 (C:\project\New_Dataset\images\Dark\test\Apple (2872).jpg): Predicted = Dark, True = Dark
Image 5 (C:\project\New_Dataset\images\Dark\test\Apple (2892).jpg): Predicted = Dark, True = Dark
Image 6 (C:\project\New_Dataset\images\Dark\test\Apple (4506).jpg): Predicted = Dark, True = Dark
Image 7 (C:\project\New_Dataset\images\Dark\test\Apple (4537).jpg): Predicted = Dark, True = Dark
Image 8 (C:\project\New_Dataset\images\Dark\test\Apple (4539).jpg): Predicted = Dark, True = Dark
Image 9 (C:\project\New_Dataset\images\Dark\test\Apple (4545).jpg): Predicted = Dark, True = Dark
Image 10 (C:\project\New_Dataset\images\Dark\test\Apple (4553).jpg): Predicted = Dark, True = Dark
Image 11 (C:\project\New_Dataset\images\Dark\test\Apple (5669).JPG): Predicted = Dark, True = Dark
Image 12 (C:\project\New_Dataset\images\Shadow\test\apple (222).png): Predicted = Shadow, True = Shadow
Image 13 (C:\project\New_Dataset\images\Shadow\test\Apple (5619).JPG): Predicted = Shadow, True = Shadow
Image 14 (C:\project\New_Dataset\images\Sunlight\test\Apple (3094).jpg): Predicted = Sunlight, True = Sunlight
Image 15 (C:\project\New_Dataset\images\Sunlight\test\Apple (4916).jpg): Predicted = Sunlight, True = Sunlight
Image 16 (C:\project\New_Dataset\images\Sunlight\test\Apple (4921).jpg): Predicted = Sunlight, True = Sunlight
Test Accuracy: 1.0

```

## SVM validation results using RBF kernel in VS code.

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
Ready to run: validate.py
Do you want to continue? (y/n): y
Loading from folder: C:\project\New_Dataset\images\Blur\validate
Processing folder: C:\project\New_Dataset\images\Blur\validate
Loading from folder: C:\project\New_Dataset\images\Dark\validate
Processing folder: C:\project\New_Dataset\images\Dark\validate
Loading from folder: C:\project\New_Dataset\images\Shadow\validate
Processing folder: C:\project\New_Dataset\images\Shadow\validate
Loading from folder: C:\project\New_Dataset\images\Sunlight\validate
Processing folder: C:\project\New_Dataset\images\Sunlight\validate
Image 1 (C:\project\New_Dataset\images\Blur\validate\apple (12).png): Predicted = Blur, True = Blur
Image 2 (C:\project\New_Dataset\images\Blur\validate\apple (240).png): Predicted = Blur, True = Blur
Image 3 (C:\project\New_Dataset\images\Blur\validate\apple (275).png): Predicted = Blur, True = Blur
Image 4 (C:\project\New_Dataset\images\Blur\validate\apple (296).png): Predicted = Blur, True = Blur
Image 5 (C:\project\New_Dataset\images\Blur\validate\apple (362).png): Predicted = Blur, True = Blur
Image 6 (C:\project\New_Dataset\images\Blur\validate\apple (364).png): Predicted = Sunlight, True = Blur
Image 7 (C:\project\New_Dataset\images\Blur\validate\apple (366).png): Predicted = Sunlight, True = Blur
Image 8 (C:\project\New_Dataset\images\Blur\validate\Apple (4340).jpg): Predicted = Blur, True = Blur
Image 9 (C:\project\New_Dataset\images\Dark\validate\Apple (2870).jpg): Predicted = Dark, True = Dark
Image 17 (C:\project\New_Dataset\images\Dark\validate\Apple (4489).jpg): Predicted = Dark, True = Dark
Image 18 (C:\project\New_Dataset\images\Dark\validate\Apple (4505).jpg): Predicted = Dark, True = Dark
Image 19 (C:\project\New_Dataset\images\Dark\validate\Apple (4508).jpg): Predicted = Dark, True = Dark
Image 20 (C:\project\New_Dataset\images\Dark\validate\Apple (4533).jpg): Predicted = Dark, True = Dark
Image 21 (C:\project\New_Dataset\images\Dark\validate\Apple (4567).jpg): Predicted = Dark, True = Dark
Image 22 (C:\project\New_Dataset\images\Dark\validate\Apple (4583).jpg): Predicted = Dark, True = Dark
Image 23 (C:\project\New_Dataset\images\Dark\validate\Apple (4587).jpg): Predicted = Dark, True = Dark
Image 24 (C:\project\New_Dataset\images\Dark\validate\Apple (4591).jpg): Predicted = Dark, True = Dark
Image 25 (C:\project\New_Dataset\images\Dark\validate\Apple (4598).jpg): Predicted = Dark, True = Dark
Image 26 (C:\project\New_Dataset\images\Shadow\validate\apple (3).png): Predicted = Shadow, True = Shadow
Image 27 (C:\project\New_Dataset\images\Shadow\validate\apple (395).png): Predicted = Dark, True = Shadow
Image 28 (C:\project\New_Dataset\images\Shadow\validate\apple (4).png): Predicted = Shadow, True = Shadow
Image 29 (C:\project\New_Dataset\images\Shadow\validate\Apple (4040).jpg): Predicted = Shadow, True = Shadow
Image 30 (C:\project\New_Dataset\images\Shadow\validate\apple (86).png): Predicted = Blur, True = Shadow
Image 31 (C:\project\New_Dataset\images\Sunlight\validate\Apple (3017).jpg): Predicted = Sunlight, True = Sunlight
Image 32 (C:\project\New_Dataset\images\Sunlight\validate\Apple (3085).jpg): Predicted = Sunlight, True = Sunlight
Image 33 (C:\project\New_Dataset\images\Sunlight\validate\Apple (3197).jpg): Predicted = Blur, True = Sunlight
Image 34 (C:\project\New_Dataset\images\Sunlight\validate\Apple (3230).jpg): Predicted = Blur, True = Sunlight
Image 35 (C:\project\New_Dataset\images\Sunlight\validate\Apple (4737).jpg): Predicted = Blur, True = Sunlight
Image 36 (C:\project\New_Dataset\images\Sunlight\validate\Apple (4918).jpg): Predicted = Sunlight, True = Sunlight
Image 37 (C:\project\New_Dataset\images\Sunlight\validate\apple (55).png): Predicted = Shadow, True = Sunlight
Validation Accuracy: 0.7837837837837838

```

### Implementation Details

#### Code to train the SVM model with the label data (Blur,Sublight,Shadow,Dark)

```

def train_svm(features, labels):
    scaler = StandardScaler()
    features = scaler.fit_transform(features)

    #clf = svm.SVC(kernel='linear')
    clf = svm.SVC(kernel='rbf', gamma='scale')

```

```
clf.fit(features, labels)

joblib.dump(scaler, 'scaler.pkl')
joblib.dump(clf, 'svm_model.pkl')
return clf
```

### Code to test the image for the classification after training

```
def test_svm(features, labels, file_paths):
    scaler = joblib.load('scaler.pkl')
    clf = joblib.load('svm_model.pkl')

    features = scaler.transform(features)
    predictions = clf.predict(features)

    correct = 0
    total = len(labels)
    for i in range(total):
        print(f"Image {i + 1} ({file_paths[i]}): Predicted = {predictions[i]}, True = {labels[i]}")
        if predictions[i] == labels[i]:
            correct += 1

    accuracy = correct / total
    return accuracy
```

### validation code

```
def validate_svm(features, labels, file_paths):

    scaler = joblib.load('scaler.pkl')
    clf = joblib.load('svm_model.pkl')

    features = scaler.transform(features)
```

```

predictions = clf.predict(features)

correct = 0
total = len(labels)
for i in range(total):
    print(f"Image {i + 1} ({file_paths[i]}): Predicted = {predictions[i]}, True =
{labels[i]}")
    if predictions[i] == labels[i]:
        correct += 1

accuracy = correct / total
return accuracy

```

### Filters code

```

def improve_blur(img):
    """Apply a basic sharpening filter and slight contrast adjustment."""
    sharpening_kernel = np.array([[0, -1, 0],
                                   [-1, 5, -1],
                                   [0, -1, 0]])
    sharpened = cv2.filter2D(img, -1, sharpening_kernel)
    enhanced_img = adjust_brightness_contrast(sharpened, alpha=1.2, beta=30)
    return enhanced_img

def improve_dark(img):
    """Enhance a dark image using brightness/contrast adjustment and CLAHE."""
    lab = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)
    l, a, b = cv2.split(lab)
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
    cl = clahe.apply(l)
    limg = cv2.merge((cl, a, b))

```

```
enhanced_img = cv2.cvtColor(limg, cv2.COLOR_LAB2BGR)
enhanced_img = adjust_brightness_contrast(enhanced_img, alpha=1.3, beta=40)
return enhanced_img
```

```
def improve_shadow(img):
```

```
    """Enhance visibility in shadows with brightness/contrast adjustment."""
    enhanced_img = adjust_brightness_contrast(img, alpha=1.2, beta=20)
    return enhanced_img
```

```
def improve_sunlight(img):
```

```
    """Reduce glare and enhance details in sunlight images."""
    lab = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)
    l, a, b = cv2.split(lab)
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
    cl = clahe.apply(l)
    limg = cv2.merge((cl, a, b))
    enhanced_img = cv2.cvtColor(limg, cv2.COLOR_LAB2BGR)
    enhanced_img = adjust_brightness_contrast(enhanced_img, alpha=1.1, beta=-20)
    return enhanced_img
```

## Bibliography

- Bulanon, D. M., Kataoka, T., Okamoto, H., & Hata, S. (2004). Development of a real-time machine vision system for the apple harvesting robot. *SICE 2004 Annual Conference, 1*, 595–598 vol. 1.
- C. Slaughter, D., & C. Harrell, R. (1989a). Discriminating Fruit for Robotic Harvest Using Color in Natural Outdoor Scenes. *Transactions of the ASAE*, 32(2), 757–763. <https://doi.org/https://doi.org/10.13031/2013.31066>
- C. Slaughter, D., & C. Harrell, R. (1989b). Discriminating Fruit for Robotic Harvest Using Color in Natural Outdoor Scenes. *Transactions of the ASAE*, 32(2), 757–763. <https://doi.org/https://doi.org/10.13031/2013.31066>
- Cheeseman, P. C., Self, M., Kelly, J., Taylor, W., Freeman, D., & Stutz, J. C. (1988). Bayesian Classification. *AAAI*, 88, 607–611.
- Chinchuluun, R., & Lee WonSuk, L. W. (2006). *Machine vision-based citrus yield mapping system*.
- Cortes, C., Vapnik, V., & Saitta, L. (1995). Support-Vector Networks Editor. In *Machine Learning* (Vol. 20). Kluwer Academic Publishers.
- Ding, X., Li, Q., Wang, X., Chen, L., Son, J., & Song, J.-Y. (2021). Apple Detection Algorithm based on an Improved SSD. *The Journal of the Institute of Internet, Broadcasting and Communication*, 21(3), 81–89.
- Duda, R. O., Hart, P. E., & Stork, D. G. (2012). *Pattern Classification*. John Wiley & Sons.
- Feng, H., Mu, G., Zhong, S., Zhang, P., & Yuan, T. (2021). Benchmark Analysis of YOLO Performance on Edge Intelligence Devices. *2021 Cross Strait Radio Science and Wireless Technology Conference (CSRSWTC)*, 319–321. <https://doi.org/10.1109/CSRSWTC52801.2021.9631594>
- Fu, L., Majeed, Y., Zhang, X., Karkee, M., & Zhang, Q. (2020). Faster R–CNN–based apple detection in dense-foliage fruiting-wall trees using RGB and depth features for robotic harvesting. *Biosystems Engineering*, 197, 245–256.
- Gallardo, R. K., Taylor, M. R., & Hinman, H. (2010). *Cost Estimates of Establishing and Producing Gala Apples in Washington*.
- Girshick, R. (2015). *Fast R-CNN*. <http://arxiv.org/abs/1504.08083>
- Gong, X., & Zhang, S. (2023). A High-Precision Detection Method of Apple Leaf Diseases Using Improved Faster R-CNN. *Agriculture (Switzerland)*, 13(2). <https://doi.org/10.3390/agriculture13020240>

- Gongal, A., Amatya, S., Karkee, M., Zhang, Q., & Lewis, K. (2015). Sensors and systems for fruit detection and localization: A review. In *Computers and Electronics in Agriculture* (Vol. 116, pp. 8–19). Elsevier B.V. <https://doi.org/10.1016/j.compag.2015.05.021>
- Hou, J., Yang, C., He, Y., & Hou, B. (2023). Detecting diseases in apple tree leaves using FPN–ISResNet–Faster RCNN. *European Journal of Remote Sensing*, *56*(1), 2186955.
- Hunter, R. S. (1948). Photoelectric Color-Difference Meter. *Journal of the Optical Society of America (JOSA)*, *38*(7), 661.
- Javid, M., Haleem, A., Khan, I. H., & Suman, R. (2023). Understanding the potential applications of Artificial Intelligence in Agriculture Sector. *Advanced Agrochem*, *2*(1), 15–30. <https://doi.org/https://doi.org/10.1016/j.aac.2022.10.001>
- Ji, W., Zhao, D., Cheng, F., Xu, B., Zhang, Y., & Wang, J. (2012). Automatic recognition vision system guided for apple harvesting robot. *Computers & Electrical Engineering*, *38*(5), 1186–1195. <https://doi.org/https://doi.org/10.1016/j.compeleceng.2011.11.005>
- Jia, W., Tian, Y., Luo, R., Zhang, Z., Lian, J., & Zheng, Y. (2020). Detection and segmentation of overlapped fruits based on optimized mask R-CNN application in apple harvesting robot. *Computers and Electronics in Agriculture*, *172*, 105380. <https://doi.org/https://doi.org/10.1016/j.compag.2020.105380>
- Kang, H., & Chen, C. (2020). Fast implementation of real-time fruit detection in apple orchards using deep learning. *Computers and Electronics in Agriculture*, *168*, 105108. <https://doi.org/https://doi.org/10.1016/j.compag.2019.105108>
- Kaur, P., Khehra, B. S., & Mavi, E. B. S. (2021). Data augmentation for object detection: A review. *2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, 537–543.
- Kong, X., Li, X., Zhu, X., Guo, Z., & Zeng, L. (2024). Detection model based on improved faster-RCNN in apple orchard environment. *Intelligent Systems with Applications*, *21*. <https://doi.org/10.1016/j.iswa.2024.200325>
- Kumar, A., & Srivastava, S. (2020). Object Detection System Based on Convolution Neural Networks Using Single Shot Multi-Box Detector. *Procedia Computer Science*, *171*, 2610–2617. <https://doi.org/10.1016/j.procs.2020.04.283>
- Kurbah, F., Marwein, S., Marngar, T., & Sarkar, B. K. (2022). Design and development of the pineapple harvesting robotic gripper. *Communication and Control for Robotic Systems*, 437–454.
- Kurtulmus, F., Lee, W. S., & Vardar, A. (2014). Immature peach detection in colour images acquired in natural illumination conditions using statistical classifiers and

- neural network. *Precision Agriculture*, 15(1), 57–79.  
<https://doi.org/10.1007/s11119-013-9323-8>
- Lakhwani, K., Murarka, P., & Narendra, M. (2015). Color Space Transformation for Visual Enhancement of noisy color Image. *IET Image Processing*.
- Le, P.-P., Nguyen, V.-T., Guo, S.-M., Tu, C.-T., & Lien, J.-J. J. (2019). Visual-guided robot arm using multi-task faster R-CNN. *2019 International Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, 1–6.
- Li, T., Xie, F., Zhao, Z., Zhao, H., Guo, X., & Feng, Q. (2023). A multi-arm robot system for efficient apple harvesting: Perception, task plan and control. *Computers and Electronics in Agriculture*, 211. <https://doi.org/10.1016/j.compag.2023.107979>
- Liang, Q., Long, J., Zhu, W., Wang, Y., & Sun, W. (2018). Apple recognition based on convolutional neural network framework. *2018 13th World Congress on Intelligent Control and Automation (WCICA)*, 1751–1756.
- Linker, R., Cohen, O., & Naor, A. (2012). Determination of the number of green apples in RGB images recorded in orchards. *Computers and Electronics in Agriculture*, 81, 45–57. <https://doi.org/10.1016/j.compag.2011.11.007>
- Liu, F., Hua, Z., Li, J., & Fan, L. (2022). Dual UNet low-light image enhancement network based on attention mechanism. *Multimedia Tools and Applications*, 82. <https://doi.org/10.1007/s11042-022-14210-2>
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. *Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, 21–37.
- Lokanath, M., Kumar, K. S., & Keerthi, E. S. (2017). Accurate object classification and detection by faster-RCNN. *IOP Conference Series: Materials Science and Engineering*, 263(5), 052028.
- Lü, Q., Cai, J., Liu, B., Deng, L., & Zhang, Y. J. (2014). Identification of fruit and branch in natural scenes for citrus harvesting robot using machine vision and support vector machine. *International Journal of Agricultural and Biological Engineering*, 7, 115–121. <https://doi.org/10.3965/j.ijabe.20140702.014>
- Malik, M. E., & Mahmud, M. S. (2024). Enhanced Weed Detection Using YOLOv9 on Open-Source Datasets for Precise Weed Management. *2024 ASABE Annual International Meeting*, 1.
- Mukherjee, R., Bessa, M., Melo-Pinto, P., & Chalmers, A. (2021). Object Detection under Challenging Lighting Conditions Using High Dynamic Range Imagery. *IEEE Access*, 9, 77771–77783. <https://doi.org/10.1109/ACCESS.2021.3082293>

- Nam, S., Hwang, Y., Matsushita, Y., & Kim, S. J. (2016). A holistic approach to cross-channel image noise modeling and its application to image denoising. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1683–1691.
- Pejsa, J. H., & Orrock, J. E. (1984). *Intelligent robot systems: potential agricultural applications*.
- Pizer, S. M., Amburn, E. P., Austin, J. D., Cromartie, R., Geselowitz, A., Greer, T., ter Haar Romeny, B., Zimmerman, J. B., & Zuiderveld, K. (1987). Adaptive histogram equalization and its variations. *Computer Vision, Graphics, and Image Processing*, 39(3), 355–368.
- Plebe, A., & Grasso, G. (2001a). Localization of spherical fruits for robotic harvesting. *Machine Vision and Applications*, 13(2), 70–79. <https://doi.org/10.1007/PL00013271>
- Plebe, A., & Grasso, G. (2001b). Localization of spherical fruits for robotic harvesting. *Mach. Vis. Appl.*, 13, 70–79. <https://doi.org/10.1007/PL00013271>
- Rahman, S., Rahman, M. M., Abdullah-Al-Wadud, M., Al-Quaderi, G. D., & Shoyaib, M. (2016). An adaptive gamma correction for image enhancement. *EURASIP Journal on Image and Video Processing*, 2016, 1–13.
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). *You Only Look Once: Unified, Real-Time Object Detection*. <http://arxiv.org/abs/1506.02640>
- Regunathan, M., & Lee, S. (2005). Citrus Fruit Identification and Size Determination Using Machine Vision and Ultrasonic Sensors. *2005 ASAE Annual International Meeting*. <https://doi.org/10.13031/2013.19821>
- Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, 28.
- Reza, A. M. (2004). Realization of the contrast limited adaptive histogram equalization (CLAHE) for real-time image enhancement. *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, 38, 35–44.
- Rumelhart, D. E., & McClelland, J. L. (1987). Learning Internal Representations by Error Propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations* (pp. 318–362).
- Schertz, C. E., & Brown, G. K. (1968). Basic Considerations in Mechanizing Citrus Harvest. *Transactions of the ASABE*, 11, 343–346. <https://api.semanticscholar.org/CorpusID:110202030>
- Sekharamanthy, P. K., Melgani, F., & Malacarne, J. (2023). Deep Learning-Based Apple Detection with Attention Module and Improved Loss Function in YOLO. *Remote Sensing*, 15(6). <https://doi.org/10.3390/rs15061516>

- Seng, W. C., & Mirisae, S. H. (n.d.). *A New Method for Fruits Recognition System*.
- Shapiro, L. G., & Stockman, G. C. (2001). *Computer Vision*. Prentice Hall.
- Sistler, F. (1987). Robotics and intelligent machines in agriculture. *IEEE Journal on Robotics and Automation*, 3(1), 3–6.
- Snyder, F., & Ni, L. (2017). Chinese apples and the emerging world food trade order: Food safety, international trade, and regulatory collaboration between China and the European Union. *The Chinese Journal of Comparative Law*, 5(2), 253–307.
- Tahir, H., Khan, M. S., & Tariq, M. O. (2021). Performance analysis and comparison of faster R-CNN, mask R-CNN and ResNet50 for the detection and counting of vehicles. *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, 587–594.
- Tang, Y., Chen, M., Wang, C., Luo, L., Li, J., Lian, G., & Zou, X. (2020). Recognition and localization methods for vision-based fruit picking robots: A review. *Frontiers in Plant Science*, 11, 510.
- Tian, Y., Yang, G., Wang, Z., Wang, H., Li, E., & Liang, Z. (2019). Apple detection during different growth stages in orchards using the improved YOLO-V3 model. *Computers and Electronics in Agriculture*, 157, 417–426. <https://doi.org/https://doi.org/10.1016/j.compag.2019.01.012>
- Valdez, P. (2020). Apple defect detection using deep learning based object detection for better post harvest handling. *ArXiv Preprint ArXiv:2005.06089*.
- Wachs, J. P., Stern, H. I., Burks, T., & Alchanatis, V. (2010). Low and high-level visual feature-based apple detection from multi-modal images. *Precision Agriculture*, 11(6), 717–735. <https://doi.org/10.1007/s11119-010-9198-x>
- Wang, C.-Y., Yeh, I.-H., & Liao, H.-Y. M. (n.d.). *YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information*. <https://github.com/WongKinYiu/yolov9>.
- Wang, J., Yang, P., Liu, Y., Shang, D., Hui, X., Song, J., & Chen, X. (2023). Research on Improved YOLOv5 for Low-Light Environment Object Detection. *Electronics (Switzerland)*, 12(14). <https://doi.org/10.3390/electronics12143089>
- Wang, J., Zhao, D., Ji, W., Tu, J., & Zhang, Y. (2009). Application of support vector machine to apple recognition using in apple harvesting robot. *2009 International Conference on Information and Automation*, 1110–1115. <https://doi.org/10.1109/ICINFA.2009.5205083>
- Wang, L., Shoulin, Y., Alyami, H., Laghari, A. A., Rashid, M., Almotiri, J., Alyamani, H. J., & Alturise, F. (2024). A novel deep learning-based single shot multibox detector model for object detection in optical remote sensing images. *Geoscience Data Journal*, 11(3), 237–251. <https://doi.org/10.1002/gdj3.162>

- Wu, Y., & Feng, J. (2018). Development and application of artificial neural network. *Wireless Personal Communications*, *102*, 1645–1656.
- Xiao, F., Wang, H., Xu, Y., & Zhang, R. (2023). Fruit Detection and Recognition Based on Deep Learning for Automatic Harvesting: An Overview and Review. *Agronomy*, *13*, 1625. <https://doi.org/10.3390/agronomy13061625>
- Xuan, G., Gao, C., Shao, Y., Zhang, M., Wang, Y., Zhong, J., Li, Q., & Peng, H. (2020). Apple detection in natural environment using deep learning algorithms. *IEEE Access*, *8*, 216772–216780.
- Yang, R., He, Y., Hu, Z., Gao, R., & Yang, H. (2024). CA-YOLOv5: A YOLO model for apple detection in the natural environment. *Systems Science and Control Engineering*, *12*(1). <https://doi.org/10.1080/21642583.2023.2278905>
- Zhang, G., Tian, Y., Yin, W., & Zheng, C. (2024). An Apple Detection and Localization Method for Automated Harvesting under Adverse Light Conditions. *Agriculture (Switzerland)*, *14*(3). <https://doi.org/10.3390/agriculture14030485>
- Zhao, Z.-Q., Zheng, P., Xu, S., & Wu, X. (2019). Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, *30*(11), 3212–3232.
- Zou Jinming and Han, Y. and S. S.-S. (2009). Overview of Artificial Neural Networks. In D. J. Livingstone (Ed.), *Artificial Neural Networks: Methods and Applications* (pp. 14–22). Humana Press. [https://doi.org/10.1007/978-1-60327-101-1\\_2](https://doi.org/10.1007/978-1-60327-101-1_2)