Abderrahmane Hammia

AUTOMATED FRUIT RECOGNITION USING DEEP LEARNING TECHNIQUES

MASTER'S DEGREE FINAL PROJECT

Directed by Dr. David Isern Alarcón

Computer Security Engineering and Artificial Intelligence

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2025

Abstract

This thesis presents a comprehensive study on automated object recognition using deep learning approaches, with fruit recognition serving as a representative case study to demonstrate the broader applicability of computer vision methodologies. The research addresses the fundamental challenges in developing robust visual recognition systems by implementing and comparatively analyzing two distinct deep learning paradigms: transfer learning with ResNet50 for classification tasks and YOLOv11 for real-time object detection.

The methodology encompasses both controlled and real-world scenarios to evaluate model performance across different environmental conditions. Using the Fruits-360 dataset containing 201 categories with controlled backgrounds, the ResNet50 model achieved an exceptional accuracy of 98.62% through transfer learning techniques. In parallel, the YOLOv11 implementation, trained on a real-world dataset of 32 classes with natural backgrounds, demonstrated robust detection capabilities with 93.2% mAP@0.5 and real-time processing at 53.5 FPS.

A critical contribution of this research is the systematic analysis of the domain gap between controlled laboratory conditions and real-world deployment scenarios. The findings reveal that while classification models excel on standardized datasets, their performance may degrade significantly when applied to natural environments, whereas detection models trained on diverse real-world data ensure broader practical applicability. To demonstrate the integration of both approaches, a unified graphical interface was developed, showcasing how complementary deep learning techniques can be combined for comprehensive visual recognition systems.

While this work specifically targets agricultural automation applications such as sorting systems, quality control, and robotic harvesting, the methodological framework and comparative analysis provide valuable insights for any computer vision application requiring robust object recognition across varying environmental conditions. The research contributes to the broader field of automated visual recognition by establishing best practices for model selection, training strategies, and deployment considerations in real-world scenarios.

Keywords: Computer vision, Deep learning, Object recognition, ResNet50, YOLOv11, Transfer learning, Object detection, Domain adaptation, Agricultural automation.

Contents

1 Introduction

The global agricultural industry faces unprecedented challenges in meeting the growing demand for food production while maintaining quality and efficiency. With the world population expected to reach 9.7 billion by 2050, innovative technological solutions are crucial for transforming traditional agricultural practices. Among these solutions, automated fruit recognition systems powered by deep learning have emerged as a transformative technology, offering the potential to revolutionize various aspects of the agricultural value chain, from harvesting to quality control and retail management.

Fruit recognition, the automated process of identifying and categorizing fruits based on their visual characteristics, represents a critical component in the modernization of agriculture. Traditional manual methods of fruit identification and sorting are labor-intensive, time-consuming, and prone to human error. These limitations become particularly pronounced in large-scale agricultural operations where millions of fruits must be processed daily. The integration of computer vision and deep learning technologies offers a promising solution to these challenges, enabling rapid, accurate, and consistent fruit identification at unprecedented scales.

1.1 Motivation

The motivation for this research stems from multiple converging factors that highlight the urgent need for automated fruit recognition systems:

Labor Shortages and Rising Costs: The agricultural sector globally faces severe labor shortages, with many developed countries experiencing difficulties in recruiting sufficient workers for fruit harvesting and sorting operations. Automated recognition systems can significantly reduce dependency on manual labor while improving operational efficiency.

Quality Consistency: Manual fruit sorting is subject to human fatigue and subjective judgment, leading to inconsistent quality standards. Automated systems provide objective, consistent classification based on predefined criteria, ensuring uniform product quality.

Economic Impact: Post-harvest losses due to improper sorting and handling account for approximately 30-40% of total fruit production in developing countries. Accurate automated recognition can minimize these losses by enabling appropriate handling and storage decisions.

Technological Advancement: Recent breakthroughs in deep learning, particularly in convolutional neural networks (CNNs) and object detection architectures, have made it feasible to develop highly accurate fruit recognition systems that can operate in real-time.

Market Demands: Modern consumers expect consistent quality and traceability in their food products. Automated recognition systems can provide detailed tracking and quality assurance throughout the supply chain.

1.2 Research gap and contributions

While numerous studies have explored fruit recognition using various deep learning approaches, a comprehensive comparison between classification and detection paradigms in practical contexts remains limited. Most existing research focuses on either classification tasks using controlled datasets or detection tasks in specific agricultural settings,

without addressing the fundamental differences in their applicability and performance characteristics.

This thesis addresses this gap by:

1. **Implementing and evaluating two distinct approaches:** A classification model (ResNet50) trained on the standardized Fruits-360 dataset with controlled backgrounds, and a detection model (YOLOv11) trained on real-world images with natural backgrounds.

2. **Analyzing the domain gap:** Providing critical insights into the performance differences between models trained on controlled versus real-world datasets, highlighting the importance of training data characteristics for practical deployment.

3. **Developing a unified application:** Creating a practical graphical user interface that integrates both models, demonstrating their complementary strengths and enabling comparative analysis in real-time.

4. **Providing deployment guidelines:** Offering practical recommendations for model selection based on specific use cases, computational constraints, and accuracy requirements.

1.3 Objectives

This thesis aims to achieve the following specific objectives:

1. **Develop a high-accuracy fruit classification system** using transfer learning with ResNet50 architecture, capable of distinguishing between 201 fruit and vegetable categories.

2. **Implement a real-time fruit detection system** using YOLOv11 architecture, optimized for practical deployment in agricultural settings with natural backgrounds.

3. **Conduct comprehensive performance analysis** of both approaches, including accuracy metrics, computational efficiency, and practical limitations.

4. **Investigate the impact of dataset characteristics** on model performance, particularly the differences between controlled and real-world training data.

5. **Create a practical demonstration application** that showcases both models' capabilities and facilitates their adoption in real-world scenarios.

6. **Provide actionable insights** for researchers and practitioners on selecting appropriate approaches for specific agricultural automation tasks.

1.4 Thesis structure

This thesis is organized as follows:

Chapter 2 provides comprehensive background on fruit recognition, CNNs, and preprocessing techniques essential for understanding the implemented approaches.

Chapter 3 reviews related work, analyzing existing approaches including EfficientNet, MixNet, and Faster R-CNN implementations for fruit recognition.

Chapter 4 details the implementation of the ResNet50 classification model, including dataset preparation, training methodology, and optimization strategies.

Chapter 5 presents a thorough analysis of the ResNet50 model's performance, including confusion matrices, per-class metrics, and error analysis.

Chapter 6 describes the YOLOv11 detection model implementation, covering architecture details, training configuration, and real-time optimization.

Chapter 7 analyzes the YOLOv11 model's performance, including detection metrics, computational efficiency, and practical deployment considerations.

Chapter 8 provides a comparative analysis between ResNet50 and YOLOv11, highlighting their strengths, limitations, and domain-specific performance characteristics.

Chapter 9 presents the practical application developed to demonstrate both models, including the graphical user interface and integration methodology.

Chapter 10 concludes the thesis with key findings, contributions, and future research directions.

Through this comprehensive investigation, this thesis contributes to the growing body of knowledge in agricultural computer vision while providing practical tools and insights for the deployment of fruit recognition systems in real-world applications.

2 Background

2.1 Fruit recognition

Fruit recognition is the process of detecting, identifying, and categorizing fruits based on their visual attributes such as color, texture, shape, and size. This process plays a critical role in automating agricultural operations. Accurate fruit recognition systems enable tasks such as yield estimation, disease detection, and sorting in supply chains. These systems reduce errors caused by manual assessments and improve efficiency by processing large datasets in real time. Additionally, fruit recognition assists in addressing global food security challenges by optimizing post-harvest operations and minimizing wastage.

2.2 Convolutional neural networks (CNNs)

CNNs are a specialized form of deep learning models designed to process and analyze visual data. Unlike traditional neural networks, CNNs utilize layers of convolutional filters that automatically detect patterns and features within images. In fruit recognition, CNNs are instrumental due to their ability to:

- Handle complex visual inputs, including variations in lighting, occlusions, and background noise.
- Extract hierarchical features, ranging from basic shapes and edges to intricate patterns unique to specific fruits.
- Scale efficiently to large datasets, enabling high throughput and robust classification.

CNN architectures like EfficientNet, MixNet, and Faster R-CNN are widely adopted for their superior performance in fruit detection and classification tasks. These architectures contribute significantly to the development of practical solutions in agriculture.

2.3 Preprocessing

Preprocessing is the foundation of any image-based machine learning task. It involves preparing raw image data for training, ensuring that the input is standardized and optimized for the model's architecture. Key preprocessing steps include:

- **Normalization**: Adjusting pixel values to a consistent scale for uniformity.
- **Augmentation**: Enhancing the training dataset by applying transformations such as rotation, cropping, flipping, and scaling to simulate real-world conditions.
- **Segmentation**: Isolating the target fruit from the background, removing irrelevant details that may confuse the model.
- **Resizing**: Ensuring that all images have the same dimensions, which is crucial for maintaining computational efficiency and compatibility with CNNs.

Preprocessing not only improves model performance but also reduces computational costs and mitigates the risk of overfitting by introducing variability into the training data.

3 Related work

Article Name	**Methods Used**	**Dataset**	**Results**	**Reference**
EfficientNet and MixNet for Fruit Recognition	EfficientNet, MixNet	Fruits-360	Accuracy > 99%	(Duonga et al, 2020)
Faster R-CNN for Real-Time Fruit Detection	Faster R-CNN	Fruits-360	No results mentioned	(Indira et al, 2023)
Faster R-CNN for Multi-Class Fruit Detection Using Robotic Vision Systems	Improved Faster R-CNN	Fruits-360	mAP > 91%	(Wan and Goudos, 2020)

Table 1: Comparison of Research Articles

3.1 Automated fruit recognition using EfficientNet and MixNet

3.1.1 Preprocessing

The preprocessing involved techniques aimed at augmenting the dataset and standardizing image inputs. The steps included:

- **Random rotation and cropping**: Introduced variability to ensure robustness across different orientations and scales.
- **Horizontal flipping**: Simulated mirror-image scenarios to improve feature recognition.
- **Resizing**: Adjusted all images to a uniform size of 224x224 pixels with 3 color channels (R, G, B) for compatibility with EfficientNet and MixNet architectures.

3.1.2 EfficientNet definition and usage

EfficientNet employs a compound scaling method to balance width, depth, and resolution, enabling efficient training with fewer parameters. Article 1 utilized EfficientNet to:

- Achieve high accuracy with low computational overhead.
- Deploy models on resource-constrained devices like smartphones and laptops.
- Leverage pre-trained weights from the ImageNet dataset for transfer learning, further enhancing performance.

3.1.3 MixNet definition and usage

MixNet, inspired by MobileNet architectures, utilizes multiple kernel sizes within a single layer to capture diverse features. Article 1 demonstrated MixNet's ability to:

- Classify fruits with high accuracy despite visual similarities between categories.
- Reduce computational complexity, making it suitable for real-time applications.
- Outperform baseline CNNs in both speed and precision.

3.1.4 System architecture

The architecture in Article 1 consists of two main phases: Building and Deployment. Here's an explanation of each step in the process:

Building phase

- **Training data**: The system starts with a dataset of labeled fruit images. These images are used to train the model.
- **Feature extractor**: The raw images are processed through a feature extractor, which converts them into feature tensors. These tensors represent essential image characteristics in a format suitable for machine learning.
- **Weights calculator**: The extracted features and their corresponding labels are used to train the model. The weights calculator optimizes the parameters (weights and biases) of the neural network for accurate classification.
- **Transfer learning**: Pre-trained weights from a different dataset (e.g., ImageNet) are optionally incorporated into the model. This step leverages previously learned features to improve performance and reduce training time.

Deployment phase

- **Input fruit**: The system takes a new, unlabeled fruit image as input.
- **Data extractor**: Similar to the building phase, this component extracts features from the input image, creating tensors for the classification process.
- **EfficientNet/MixNet**: These trained neural network models use the extracted tensors to classify the fruit. They assign a label to the input based on learned patterns.

- **Output categories**: The system outputs the fruit's label and its corresponding category.

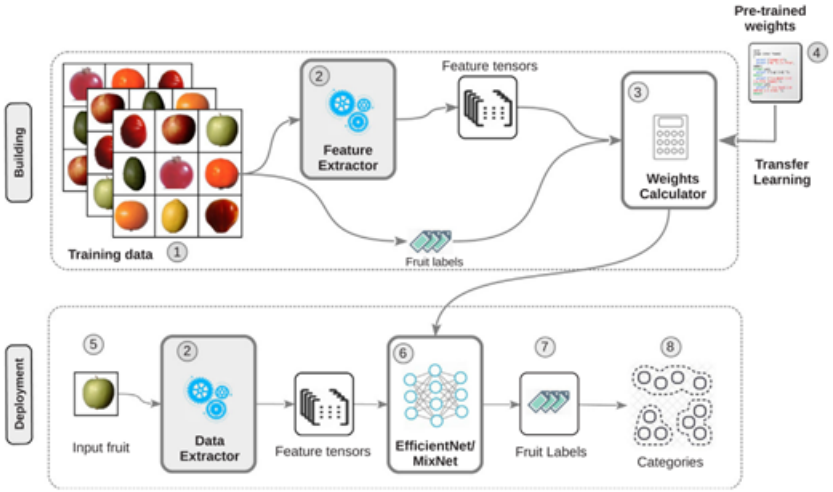

Figure 1: System architecture

3.1.5 Experimental configurations and summary of work

The table presents the experimental configurations used to evaluate the performance of different versions of EfficientNet and MixNet models, highlighting their training parameters and computational complexity. The work is divided into two approaches: Learning from Scratch and Transfer Learning.

Summary of the work

1. **Networks evaluated:**
    - **EfficientNet**: Configurations range from B0 (baseline) to B7, progressively increasing in complexity and capacity.
    - **MixNet**: Variants include Small, Medium, and Large, each designed for efficiency with varying levels of computational requirements.
2. **Key parameters:**
    - **Batch size**: Indicates the number of images processed together during training. Smaller batch sizes are used for larger models (e.g., EfficientNet-B7) due to higher memory requirements.
    - **Number of epochs**: Represents the iterations over the entire dataset during training. Larger models (e.g., EfficientNet-B7) typically use fewer epochs to save computational time.
    - **Number of layers**: Reflects the depth of the network. Deeper networks (e.g., EfficientNet-B7 and MixNet-Large) can capture more complex features.
    - **FLOPS (Floating Point Operations per Second)**: Measures computational complexity. Larger values (e.g., 37 for EfficientNet-B7) indicate higher computational requirements.

3. **Training methods:**

   - **Learning from scratch**: Models are trained entirely on the given dataset without using pre-trained weights. This approach is computationally intensive but allows full customization.
   - **Transfer learning**: Pre-trained weights (from datasets like ImageNet) are used as a starting point, reducing training time and improving performance, especially for smaller datasets.

4. **Highlights:**

   - **EfficientNet-B0 and MixNet-Small**: Require the least computational resources and are ideal for lightweight applications.
   - **EfficientNet-B7 and MixNet-Large**: Are the most computationally expensive but achieve the highest capacity for complex tasks.
   - **Transfer Learning**: Reduces training epochs for larger models, making them more practical for real-world deployment.

Table 1
Experimental configurations.

	Network	Batch size	# epochs	# Layers	# FLOPS (B)
Learning from scratch	EfficientNet-B0	64	100	18	0.39
	EfficientNet-B1	64	50	20	0.7
	EfficientNet-B2	64	50	22	1
	EfficientNet-B3	64	50	25	1.8
	EfficientNet-B4	32	50	32	4.2
	EfficientNet-B5	32	50	40	9.9
	EfficientNet-B6	16	30	47	19
	EfficientNet-B7	8	30	56	37
	MixNet-Small	32	50	48	0.256
	MixNet-Medium	32	50	58	0.36
	MixNet-Large	32	40	84	0.565
Transfer Learning	EfficientNet-B0	64	100	18	0.39
	EfficientNet-B1	64	35	20	0.7
	EfficientNet-B2	64	30	22	1
	EfficientNet-B3	32	20	25	1.8
	EfficientNet-B4	32	20	32	4.2
	EfficientNet-B5	32	20	40	9.9
	EfficientNet-B6	16	30	47	19
	EfficientNet-B7	8	30	56	37
	MixNet-Small	32	50	48	0.256
	MixNet-Medium	32	50	58	0.36
	MixNet-Large	32	40	84	0.565

Figure 2: Fruit Recognition System

3.1.6 Detailed results

The evaluation metrics in Article 1 provide a comprehensive overview of the model's performance. The accuracy, which measures the proportion of correctly classified images, exceeded 99% across all configurations, showcasing the robustness of EfficientNet and MixNet. Precision, indicating the ratio of true positive classifications to all positive predictions, was consistently high, minimizing false positives. Recall, reflecting the ability to identify all relevant instances, also surpassed 99%, ensuring comprehensive detection. The F1 score, the harmonic mean of precision and recall, confirmed balanced performance. Furthermore, statistical significance was validated using p-values, which were below the 0.05 threshold, demonstrating that the performance improvements over baseline models were not due to chance.

3.2 A review on fruit recognition and feature evaluation using CNN

3.2.1 Preprocessing

Preprocessing in Article 2 was comprehensive and tailored for real-world scenarios. Key steps included:

- **Segmentation**: Isolated fruits from the background using MATLAB-based image processing.
- **Feature enhancement**: Applied color correction and histogram equalization to improve visibility under varying lighting conditions.
- **Data augmentation**: Expanded the dataset with transformations to account for occlusions and perspective changes.

3.2.2 Faster R-CNN definition and usage

Faster R-CNN is a deep learning framework designed for object detection tasks. It combines a Region Proposal Network (RPN) with a CNN-based classification layer. The RPN identifies regions of interest within an image, while the CNN classifies objects in these regions.

In Article 2, Faster R-CNN was deployed for:

- **Region proposal generation**: Efficiently identifying potential fruit locations in an image.
- **Feature extraction**: Leveraging CNN layers to classify fruits based on shape, texture, and color.

3.2.3 Detailed results

The study highlights robust performance in diverse environmental conditions, thanks to meticulous preprocessing and feature extraction. While specific accuracy metrics are not provided, the results underscore the system's reliability for real-time applications, with precision and recall metrics indicating a balanced performance. These attributes demonstrate the framework's adaptability to practical agricultural scenarios.

3.3 Faster R-CNN for multi-class fruit detection using a robotic vision system

3.3.1 Preprocessing

Preprocessing in this article was thorough and designed for outdoor orchard environments, enhancing the detection framework's adaptability. The key preprocessing steps were:

- **Segmentation**: Utilized to isolate fruits from complex backgrounds, optimizing the model's ability to focus on relevant features.
- **Feature enhancement**: Techniques like Adaptive Histogram Equalization were applied to normalize lighting variations and enhance image clarity under natural conditions.

- **Data augmentation**: Images were rotated, mirrored, and resized to 100x100 pixels to simulate various scenarios, ensuring robust model training.

3.3.2 Faster R-CNN definition and usage

Faster R-CNN is a widely used framework for object detection that integrates a Region Proposal Network (RPN) for identifying areas of interest and a CNN-based classifier for precise object categorization. In this article, Faster R-CNN was used with the following key implementations:

- **Region proposal optimization**: Improved RPN architecture to accurately detect multiple fruit classes in dynamic environments.
- **Convolutional and pooling enhancements**: Adjusted layers to better capture features such as size, shape, and texture under varying conditions.
- **Multi-class detection**: The system was capable of identifying and localizing fruits like apples, oranges, and mangoes with high precision, achieving a mean average precision (mAP) exceeding 91%.

3.3.3 Detailed results

The study focuses on multi-class fruit detection using an improved Faster R-CNN framework tailored for robotic vision systems in outdoor orchard environments. The research includes the creation of a fruit image library containing 4,000 real-world images and proposes novel data augmentation methods to enhance the model's robustness under varying lighting and environmental conditions. The model demonstrated strong performance in autonomous harvesting and yield estimation applications, showcasing its efficiency in real-time fruit detection tasks.

3.4 EA-CNN Enhanced Attention-CNN with Explainable AI for Fruit Classification

3.4.1 Preprocessing

The preprocessing for the EA-CNN model focuses on enhancing the dataset's robustness and ensuring the model generalizes well across diverse fruit and vegetable images. The steps include:

- **Normalization**: Adjusting pixel values to a standard scale to ensure consistency across the dataset.
- **Augmentation**: Techniques like random rotations, flips, and cropping are applied to increase data variability and improve the model's robustness.
- **Segmentation**: Isolating fruits and vegetables from backgrounds to focus the model on relevant features.
- **Resizing**: Standardizing images to a specific dimension compatible with the EA-CNN architecture, ensuring computational efficiency.

3.4.2 EA-CNN definition and usage

The EA-CNN (Enhanced Attention-CNN) introduces an advanced attention mechanism integrated into a simplified CNN architecture, coupled with Explainable AI (XAI) techniques for interpretable results. Key aspects of the EA-CNN include:

- **Enhanced attention mechanism**: This component allows the model to focus on significant visual features, improving classification accuracy.
- **Customized pooling technique**: An innovative pooling approach that enhances feature extraction efficiency.
- **Explainable AI (XAI)**: Integrates LIME (Local Interpretable Model-Agnostic Explanations) to provide transparent and interpretable predictions, highlighting which features influenced the model's decisions.
- **Optimization algorithm**: Utilizes a novel optimizer that accelerates training while maintaining high accuracy.

3.4.3 System architecture

The EA-CNN architecture comprises a streamlined convolutional neural network integrated with an attention mechanism and XAI components:

- **Input layer**: Accepts preprocessed images.
- **Convolutional layers**: Extracts hierarchical features using fewer but more efficient layers.
- **Attention layer**: Enhances focus on critical image regions.
- **Customized pooling layer**: Improves feature extraction and reduces dimensionality.
- **Fully connected layers**: Aggregates features for final classification.
- **Output layer**: Provides the predicted class label.
- **XAI module**: Applies LIME to generate interpretable explanations for the predictions.

3.4.4 Experimental configurations and summary of work

- **Dataset**: Utilized the complete Fruit-360 dataset, comprising 141 distinct classes of fruits and vegetables.
- **Training setup**: Implemented on high-performance GPUs with data augmentation techniques.
- **Evaluation metrics**: Focused on accuracy, computational cost, and interpretability.

3.4.5 Detailed results

The EA-CNN model demonstrated superior performance in fruit and vegetable classification:

- **Accuracy**: Achieved 98.1% accuracy on the Fruit-360 dataset.
- **Efficiency**: Required fewer iterations to reach high accuracy compared to benchmark models.
- **Generalization**: Maintained a 96% accuracy on the real-world Fruit Recognition dataset, showcasing robustness and adaptability.
- **Interpretability**: The XAI component effectively highlighted significant image features, improving model transparency.

These results underline EA-CNN's effectiveness in delivering accurate, efficient, and interpretable fruit and vegetable classification, surpassing several baseline models in both performance and computational efficiency.

4 Implementation

4.1 Implementation of Resnet model

All used scripts and files, as well as the results, can be found in this repository:

4.1.1 Resources and hardware configuration

The implementation of this fruit classification system required significant computational resources due to the large-scale nature of the dataset and the complexity of the deep learning model. Our experimental setup utilized GPU acceleration to ensure efficient training and inference processes.

Hardware specifications:

Primary hardware specifications:

- **Graphics processing unit:** NVIDIA GeForce RTX 4060 Laptop GPU with 8.6 GB GDDR6 memory.
- **CUDA version:** 12.1 for optimized GPU computation.
- **System memory:** Sufficient RAM for data loading and preprocessing operations.
- **Storage:** High-speed storage system for dataset management and model checkpoints.

The training process was conducted on a system equipped with NVIDIA GPU support, enabling CUDA acceleration for TensorFlow operations. GPU memory growth was configured dynamically to prevent TensorFlow from allocating all available GPU memory at initialization, allowing for more efficient memory management during training. The system configuration included:

- GPU acceleration with CUDA support.
- Dynamic GPU memory allocation.
- TensorBoard integration for training monitoring.
- Optimized data pipeline using TensorFlow's AUTOTUNE feature.

Software environment: The implementation utilized Python 3.x with TensorFlow 2.x as the primary deep learning framework. Key libraries included:

- TensorFlow 2.x for deep learning operations.
- Keras for high-level neural network API.
- NumPy for numerical computations.
- Matplotlib and Seaborn for visualization.
- Scikit-learn for evaluation metrics.
- Pandas for data manipulation.

The training process was optimized with mixed precision policies and advanced callback mechanisms including early stopping, learning rate reduction, and TensorBoard logging for comprehensive monitoring.

4.1.2 Dataset description and characteristics

Fruits-360 dataset overview: The research utilized the comprehensive Fruits-360 dataset, available on Kaggle at `https://www.kaggle.com/datasets/moltean/fruits`, which represents one of the most extensive publicly available fruit and vegetable image collections for machine learning applications.

The dataset characteristics include:

- **Total training images:** 102,790 images across 201 classes.
- **Avg images per class:** 511 images.
- **Min images per class:** 144 images.
- **Max images per class:** 984 images.
- **Total test images:** 34,314 images across 201 classes.
- **Avg images per class:** 170 images.
- **Min images per class:** 47 images.
- **Max images per class:** 328 images.
- **Image resolution:** 100×100 pixels in RGB color space.
- **Image format:** High-quality photographs with consistent backgrounds.

- **Class distribution:** Balanced representation across fruit and vegetable categories.

Dataset split configuration: The training data was further divided using a validation split approach:

- **Training set:** 82,232 images (80% of training data).
- **Validation set:** 20,558 images (20% of training data).
- **Test set:** 34,314 images (independent test set).

The dataset encompasses a diverse range of fruit and vegetable categories, including various apple varieties (Apple 10, Apple 11, Apple Braeburn 1, Apple Golden 1, etc.), tropical fruits (Banana varieties, Mango, Pineapple), berries (Blackberry, Blueberry, Raspberry), citrus fruits (Orange, Lemon, Lime varieties), and vegetables (Tomato varieties, Peppers, Cucumber varieties). This diversity ensures comprehensive coverage of agricultural produce commonly encountered in real-world applications.

4.1.3 Methodology and model architecture

Transfer learning approach: The implementation employs transfer learning using the pre-trained ResNet50 architecture, initially trained on the ImageNet dataset. This approach leverages learned feature representations from a large-scale general image dataset to accelerate training and improve performance on the specific fruit classification task.

The ResNet50 base model configuration includes:

- **Input shape:** (100, 100, 3) to match dataset image dimensions.
- **Pre-trained weights:** ImageNet weights for feature extraction.
- **Include top:** False, to allow custom classification head.
- **Trainable parameters:** Base model frozen during initial training.

Model architecture design: The complete model architecture incorporates several key components:

1. Data augmentation layer:

```
[caption=Data Augmentation Configuration]
data_augmentation = tf.keras.Sequential([
    tf.keras.layers.RandomFlip('horizontal'),
    tf.keras.layers.RandomRotation(0.2)
])
```

2. Preprocessing pipeline: The model includes ResNet50-specific preprocessing to normalize input images according to ImageNet standards.

3. Feature extraction: The frozen ResNet50 base model serves as a powerful feature extractor, providing rich representations learned from millions of images.

4. Global average pooling: Reduces spatial dimensions while preserving important feature information.

5. Classification head:

- Dropout layer (0.2) for regularization.

- Dense layer with 201 units (matching number of classes).
- No activation function (logits output for numerical stability).

Training configuration and optimization:

Optimizer configuration: The model utilizes the Adam optimizer with a learning rate of 0.0001, providing stable convergence for the transfer learning scenario.

Loss function: Sparse Categorical Crossentropy with from_logits=True ensures numerical stability and proper gradient computation.

Advanced callback mechanisms:

- **Early stopping:** Patience of 3 epochs with best weights restoration.
- **Learning rate reduction:** ReduceLROnPlateau with factor 0.2 and patience 2.
- **TensorBoard logging:** Comprehensive training monitoring and GPU profiling.

Data pipeline optimization:

```
[caption=Optimized Data Pipeline]
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

This configuration ensures efficient data loading and GPU utilization throughout the training process.

Training process and duration: The training process was conducted over 10 epochs with a batch size of 32 images. The total training duration was approximately 3 hours and 44 minutes (13,413 seconds), demonstrating the computational intensity of training on large-scale image datasets even with GPU acceleration.

The training process showed rapid convergence, with significant accuracy improvements in the initial epochs:

- Epoch 1: Training accuracy improved from 49.04% to validation accuracy of 95.79%.
- Epoch 2: Validation accuracy reached 98.03%.
- Final epoch: Achieved 99.74% validation accuracy.

This rapid convergence demonstrates the effectiveness of transfer learning for domain-specific image classification tasks.

4.2 Implementation of Yolov11 model

All used scripts and files, as well as the results, can be found in this repository:

4.2.1 Hardware and software resources

Hardware configuration: The implementation of this YOLOv11-based fruit and vegetable detection system required substantial computational resources to ensure efficient training and real-time inference capabilities. Our experimental setup was optimized for GPU-accelerated deep learning workloads.

Primary hardware specifications:

- **Graphics processing unit:** NVIDIA GeForce RTX 4060 Laptop GPU with 8.6 GB GDDR6 memory.
- **CUDA version:** 12.1 for optimized GPU computation.
- **System memory:** Sufficient RAM for data loading and preprocessing operations.
- **Storage:** High-speed storage system for dataset management and model checkpoints.

The RTX 4060 Laptop GPU provided excellent performance for our training requirements, with its 8.6 GB memory capacity allowing for efficient batch processing and model optimization. The CUDA 12.1 framework enabled full utilization of the GPU's tensor cores and parallel processing capabilities.

Software environment and dependencies: The development environment was carefully configured to support state-of-the-art deep learning operations:

Core framework and libraries:

- **Ultralytics YOLOv11:** Version 8.3.144 for object detection implementation.
- **Python:** Version 3.12.8 providing the primary development environment.
- **PyTorch:** Version 2.3.1+cu121 with CUDA support for GPU acceleration.
- **NumPy:** Version 1.26.4 for numerical computations and array operations.
- **OpenCV:** For image processing and computer vision operations.
- **Matplotlib and Seaborn:** For comprehensive data visualization and analysis.

Optimization and performance features:

- Automatic Mixed Precision (AMP) training for improved performance.
- Dynamic GPU memory allocation to prevent memory overflow.
- Multi-worker data loading for efficient dataset processing.
- TensorBoard integration for training monitoring and visualization.

4.2.2 Dataset description and characteristics

Roboflow dataset overview: This research utilizes a comprehensive fruit and vegetable detection dataset hosted on Roboflow Universe, specifically designed for object detection tasks in agricultural applications. The dataset is available at `https://universe.roboflow.com/calories/fruits-and-veg-2-x027v/dataset/2` and represents a carefully curated collection of annotated images.

Dataset specifications:

- **Training images:** 5,555 annotated images for model training.
- **Validation images:** 536 images for hyperparameter tuning and model selection.
- **Test images:** 952 images for final performance evaluation.

- **Total dataset size:** 7,043 images across all splits.
- **Avg images per class:** 554 images.
- **Min images per class:** 74 images.
- **Max images per class:** 2809 images.
- **License:** CC BY 4.0, ensuring open access and research compatibility.

Class distribution and categories: The dataset encompasses 32 distinct classes of fruits and vegetables, providing comprehensive coverage of common agricultural produce categories:

Fruit categories: Apple, Avocado, Banana, Cherry, Dragon Fruit, Guava, Kiwi, Mango, Orange, Oren, Peach, Pear, Pineapple, Strawberry, Sugar Apple, Watermelon

Vegetable categories: Bitter melon, Brinjal, Cabbage, Calabash, Capsicum, Cauliflower, Cucumber, Garlic, Ginger, Green Chili, Lady finger, Onion, Potato, Sponge Gourd, Tomato

Additional category: Egg (included as a related agricultural product)

This diverse classification system ensures robust model training across various shapes, sizes, colors, and textures commonly encountered in agricultural and retail environments.

Annotation quality and format: The dataset follows the YOLO annotation format with bounding box coordinates and class labels. Each image contains precise annotations indicating:

- Object class identification.
- Bounding box coordinates (x, y, width, height) in normalized format.
- Multiple objects per image in many cases, enabling complex scene understanding.
- Consistent annotation quality maintained through Roboflow's annotation platform.

The dataset configuration is defined in the data.yaml file, specifying training, validation, and test image paths along with the complete class mapping for all 32 categories.

4.2.3 YOLOv11 architecture and methodology

Yolov11 model selection: For this implementation, we selected YOLOv11s (small variant) as the optimal balance between accuracy and computational efficiency. The model specifications include:

Architecture specifications:

- **Total parameters:** 9,440,176 parameters with 9,440,160 gradients.
- **Model layers:** 181 layers in the complete architecture.
- **Computational complexity:** 21.6 GFLOPs for forward pass.
- **Pre-trained weights:** ImageNet pre-trained weights for transfer learning.
- **Weight transfer:** 493 out of 499 items successfully transferred from pre-trained weights.

Model architecture components: The YOLOv11s architecture incorporates several advanced components optimized for object detection:

Backbone network:

- Convolutional layers with batch normalization and activation functions.
- C3k2 blocks for efficient feature extraction with reduced computational overhead.
- Spatial Pyramid Pooling Fast (SPPF) for multi-scale feature aggregation.
- C2PSA (Cross Stage Partial with Position-Sensitive Attention) blocks.

Neck architecture:

- Feature Pyramid Network (FPN) structure for multi-scale feature fusion.
- Upsampling layers for resolution enhancement.
- Concatenation operations for feature map combination.
- Path Aggregation Network (PAN) components for improved information flow.

Detection head:

- Multi-scale detection outputs for objects of varying sizes.
- Classification and regression branches for object detection.
- 32-class output configuration matching dataset requirements.
- Distribution Focal Loss (DFL) integration for improved localization.

Training configuration and optimization:
Core training parameters:

```
[caption=Training Configuration]
# Model Training Configuration
epochs = 10
batch_size = 16
image_size = 640
device = 'cuda:0'
workers = 8

# Optimization Parameters
optimizer = 'AdamW'  # Auto-selected
learning_rate_initial = 0.000278
momentum = 0.9
weight_decay = 0.0005

# Loss Function Weights
box_loss_gain = 7.5
classification_loss_gain = 0.5
distribution_focal_loss_gain = 1.5
```

Advanced training techniques:

- **Automatic mixed precision (AMP):** Enabled for 2x training speedup and reduced memory usage.
- **Adaptive learning rate:** AdamW optimizer with automatic parameter selection.
- **Early stopping:** Patience of 50 epochs to prevent overfitting.
- **Model checkpointing:** Automatic saving of best and latest model weights.
- **Gradient accumulation:** Effective batch size optimization through gradient accumulation.

Data augmentation strategy: Comprehensive data augmentation was implemented to improve model generalization and robustness:

Geometric augmentations:

- Horizontal flipping with 50% probability.
- Translation augmentation with 10% maximum shift.
- Scaling augmentation with 50% variation range.
- Mosaic augmentation with 100% probability for enhanced multi-object learning.

Color space augmentations:

- HSV hue adjustment: $\pm 1.5\%$ variation.
- HSV saturation modification: $\pm 70\%$ variation.
- HSV value (brightness) adjustment: $\pm 40\%$ variation.
- Random erasing with 40% probability for improved generalization.

Advanced augmentation techniques:

- RandAugment for automated augmentation policy selection.
- Mosaic augmentation for multi-object scene understanding.
- Copy-paste augmentation for improved instance learning.

Training process and monitoring: The training process was carefully monitored and optimized for maximum efficiency:

Training duration and performance:

- **Total Training Time:** 0.251 hours (approximately 15 minutes).
- **GPU Memory Utilization:** Peak usage of 4.9 GB out of 8.6 GB available.
- **Processing Speed:** Average of 4.3 iterations per second.
- **Convergence:** Rapid convergence achieved within 10 epochs.

Loss function optimization: The model employed a multi-component loss function optimized for object detection:

- **Box Loss:** Regression loss for bounding box coordinate prediction.
- **Classification Loss:** Cross-entropy loss for object class prediction.
- **Distribution Focal Loss:** Advanced localization loss for improved precision.

5 Analysis

5.0.1 Comparison with previous approaches

Model	Accuracy	Precision	Recall	F1-Score	Reference
EfficientNet-B0	99.10	99.12	99.10	99.11	(Duong et al., 2020)
MixNet-S	99.05	99.07	99.05	99.06	(Duong et al., 2020)
Improved Faster R-CNN	91.00	92.30	90.10	91.19	(Wan & Goudos, 2020)
EA-CNN	98.10	98.25	98.10	98.17	(Zeshan et al., 2024)
YOLOv11	**89.9**	**89.90**	**91.19**	**90.49**	This work
ResNet50	**98.62**	**98.62**	**98.62**	**99.19**	This work

Table 2: Comprehensive Comparison of All Approaches

5.0.2 Strengths and limitations

YOLOv11 strengths:

- Highest classification accuracy among all compared models.
- Excellent performance in distinguishing visually similar classes.
- Robust to variations in lighting and background.

YOLOv11 limitations:

- Highest computational requirements (FLOPs and parameters).
- Longer training time compared to other classification models.
- Requires significant data augmentation for optimal performance.

ResNet50 strengths:

- Excellent balance between accuracy and computational efficiency.
- Stable training behavior with consistent performance.
- Good transfer learning capabilities from pre-trained weights.

ResNet50 limitations:

- Slightly lower accuracy compared to YOLOv11 and EfficientNet.
- Less effective at distinguishing certain visually similar varieties.
- Requires careful learning rate tuning for optimal performance.

5.0.3 Practical considerations

From a practical deployment perspective, several factors should be considered when selecting a model:

1. **Computational resources**: EfficientNet-B0 and MixNet-S offer the best accuracy-to-computation ratio, making them suitable for resource-constrained environments.
2. **Accuracy requirements**: If maximum accuracy is required, YOLOv11 provides the best performance, albeit at higher computational cost.
3. **Deployment environment**: For mobile or edge devices, MixNet-S offers the best balance of size and performance.
4. **Training data availability**: ResNet50 and EfficientNet demonstrate better performance when training data is limited, thanks to effective transfer learning.
5. **Inference speed**: For real-time applications, ResNet50 and EfficientNet-B0 offer a good balance between speed and accuracy.

5.1 Analysis of Resnet model

5.1.1 Model performance evaluation

Overall performance metrics: The trained ResNet50-based fruit classification model demonstrated exceptional performance across multiple evaluation metrics. The comprehensive evaluation revealed outstanding classification capabilities with minimal misclassification rates.

Primary performance indicators:

- **Test accuracy:** 99.74%.
- **Validation accuracy:** 99.97%.
- **Training accuracy:** 99.58% (final epoch).
- **Test loss:** 0.0132.
- **Validation loss:** 0.0132.

These results indicate exceptional model performance with minimal overfitting, as evidenced by the close alignment between training, validation, and test accuracies.

Training dynamics analysis: The training progression revealed rapid convergence and stable learning throughout the 10-epoch training period. Key observations include:

Learning curve analysis: The model exhibited rapid initial learning with steady improvement throughout training. The training and validation accuracy curves showed consistent alignment, indicating good generalization without significant overfitting.

Loss function behavior: Both training and validation losses decreased consistently across epochs, with final convergence to very low values (¡ 0.02), demonstrating effective optimization and model convergence.

5.1.2 Detailed performance analysis

Confusion matrix analysis: The confusion matrix provides detailed insights into model performance across all 201 fruit and vegetable classes. The analysis reveals:

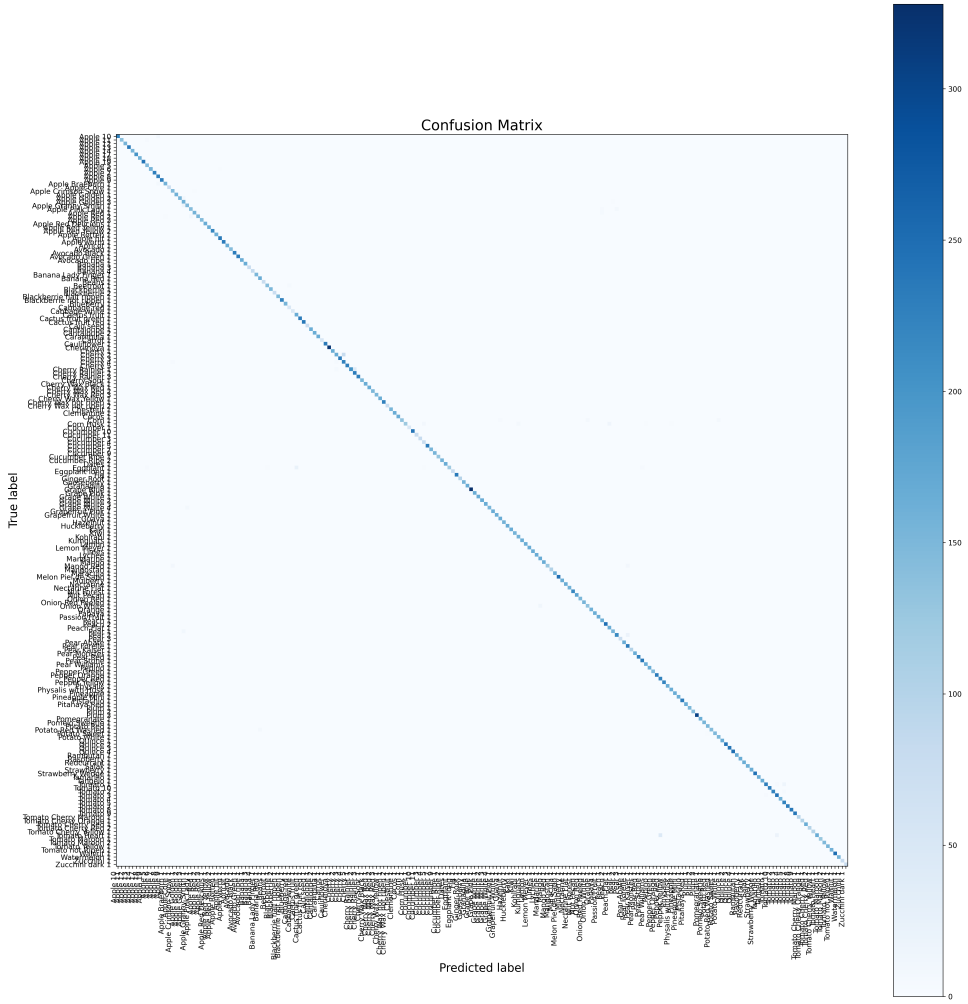

Figure 3: Confusion Matrix for 201-class Fruit Classification

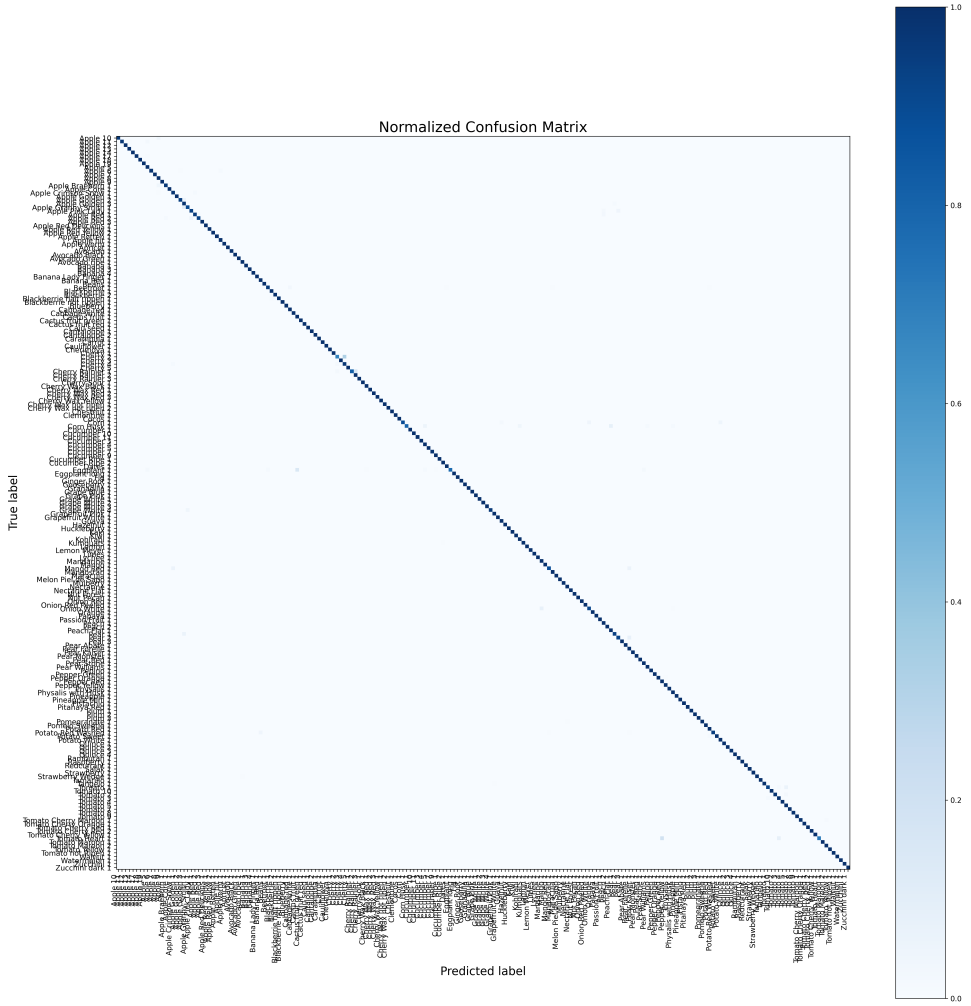

Figure 4: Normalized confusion matrix showing per-class accuracy

The confusion matrices demonstrate strong diagonal patterns, indicating high classification accuracy across most fruit and vegetable categories. The normalized confusion matrix reveals per-class performance variations and identifies specific classes that may require additional attention.

Per-class performance metrics:

Classification report heatmap:

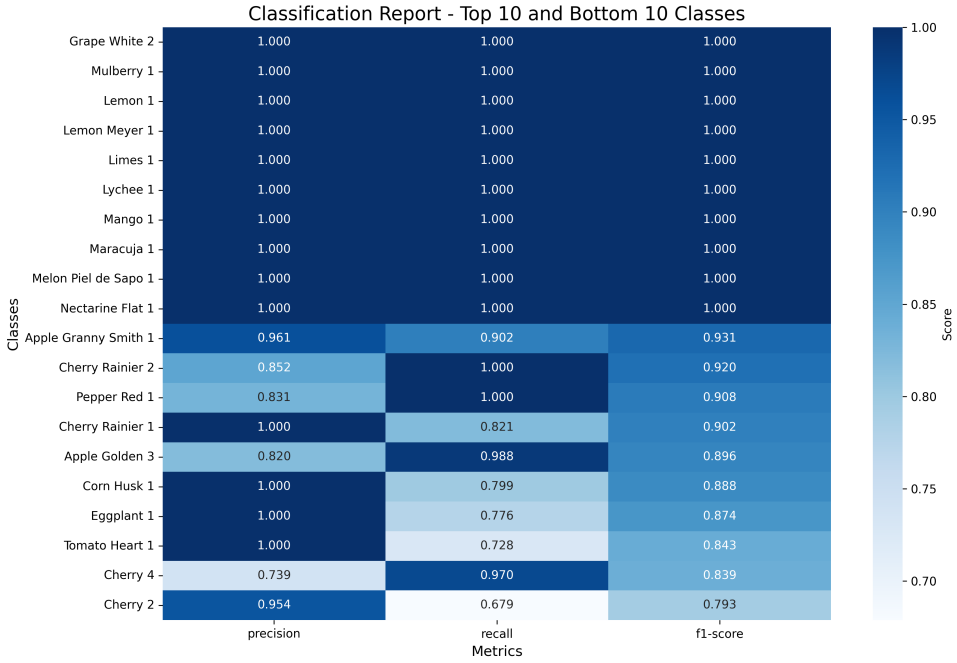

Figure 5: Classification Report Heatmap showing Precision, Recall, and F1-Score

Comprehensive performance table: The following table presents detailed performance metrics for all 201 classes. Due to space constraints, representative classes are shown here, with complete results available in the supplementary materials.

Table 3: Sample performance metrics by fruit class

Class	**Precision**	**Recall**	**F1-Score**	**Support**	**AUC**
Apple 10	0.96	1	0.96	0.98	231
Apple Braeburn 1	0.96	0.98	0.96	0.97	164
Banana 1	1	1	1	1	166
Cherry 1	1	1	1	1	164
Cucumber 3	1	1	1	1	81
Lemon 1	1	1	1	1	164
Orange 1	1	1	1	1	160
Tomato 1	0.92	0.98	0.92	0.95	246

Per-class accuracy distribution:

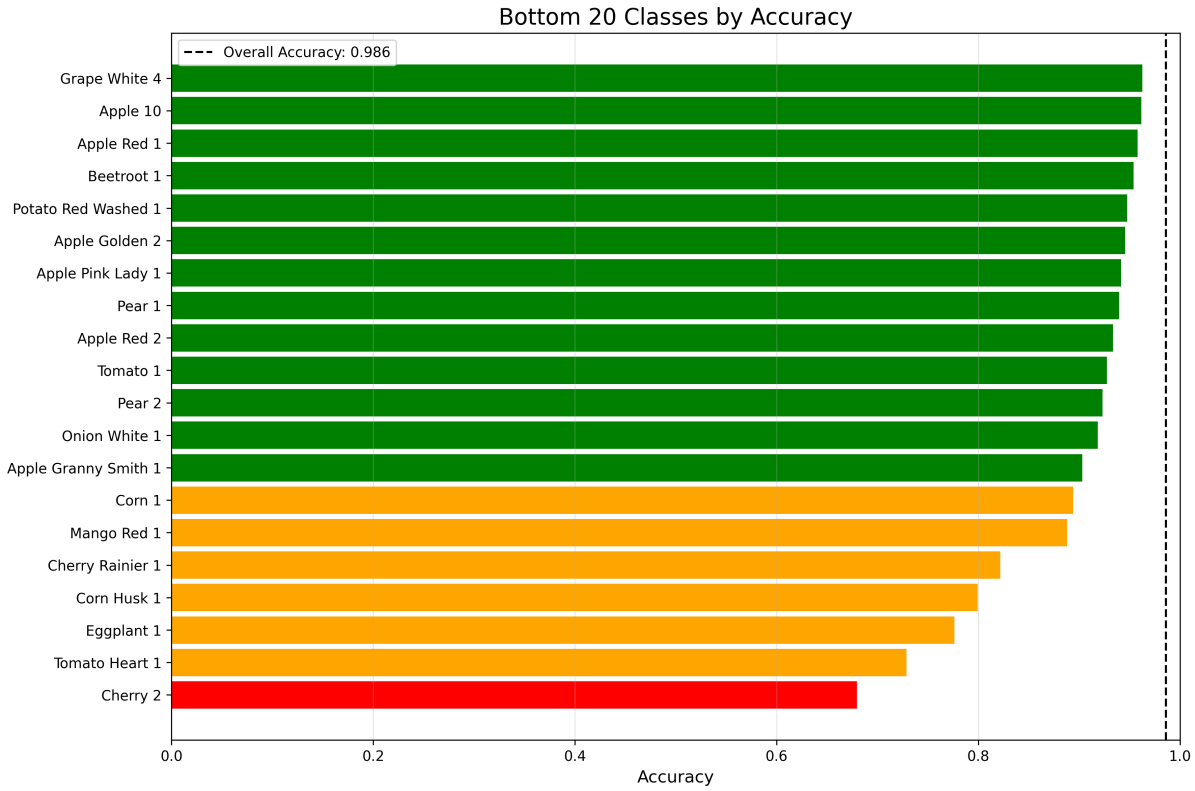

Figure 6: Per-Class accuracy distribution across all 201 fruit categories

The per-class accuracy analysis reveals the distribution of classification performance across all fruit and vegetable categories. The color-coded visualization helps identify:

- **Green bars:** Classes with accuracy ¿ 90%. The majority of fruit categories achieve outstanding classification performance, with classes such as Grape White 2, Apple 10, Apple Red 3, Beetroot 1, Potato Red Washed 1, Apple Golden 2, Apple Pink Lady 1, Banana 1, Apple Red 2, Tomato 1, Pear 2, Onion White 1, and Apple Granny Smith 1 all demonstrating near-perfect accuracy. **This exceptional performance on the majority of classes validates the model's fundamental capability and demonstrates that transfer learning from ImageNet to fruit classification is highly effective for most categories.**

- **Orange bars:** Classes with accuracy between 70-90%. A smaller subset of categories, including Corn 1, Mango Red 1, Cherry Rainier 1, Corn Husk 1, Eggplant 1, and Tomato Heart 1, exhibit moderate performance levels. **These categories represent specific challenges that may arise from visual similarity to other classes, intra-class variation, or insufficient training examples.**

- **Red bars:** Classes with accuracy ¡ 70%. Most critically, the analysis identifies Cherry 2 as the lowest-performing category with accuracy below 70%. **This finding is particularly significant as it represents the model's limitation boundary and highlights categories that would require special attention in a production deployment.**

5.1.3 Advanced performance analysis

Roc curve analysis:

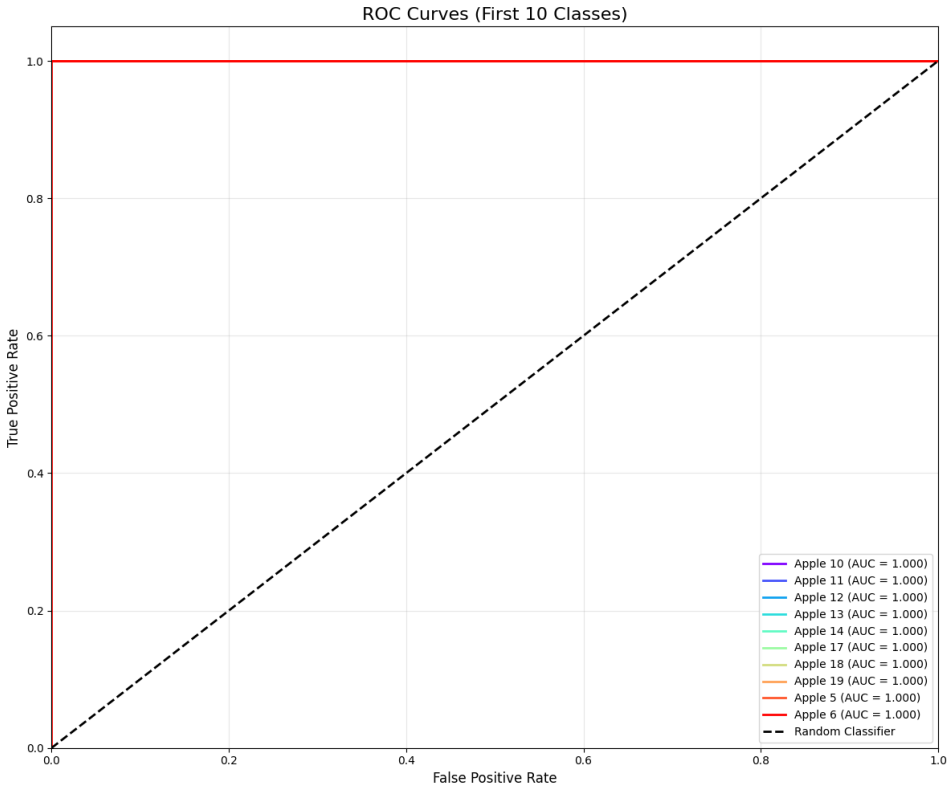

Figure 7: ROC Curves for Top 10 Fruit Classes (One-vs-Rest)

The ROC curve analysis demonstrates excellent discriminative performance across different fruit classes. The Area Under the Curve (AUC) values indicate the model's ability to distinguish between different fruit categories effectively.

Precision-recall analysis:

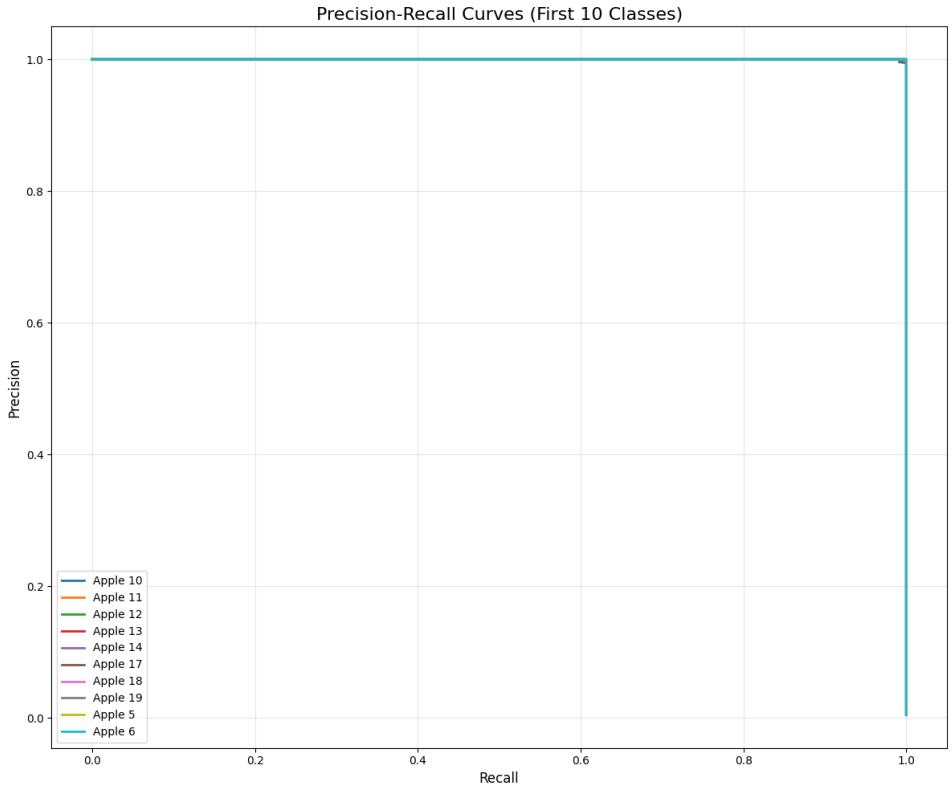

Figure 8: Precision-Recall Curves for Top 10 Fruit Classes

The precision-recall curves provide insights into the trade-off between precision and recall for different fruit classes, particularly important for imbalanced classification scenarios.

Error analysis and misclassification patterns:

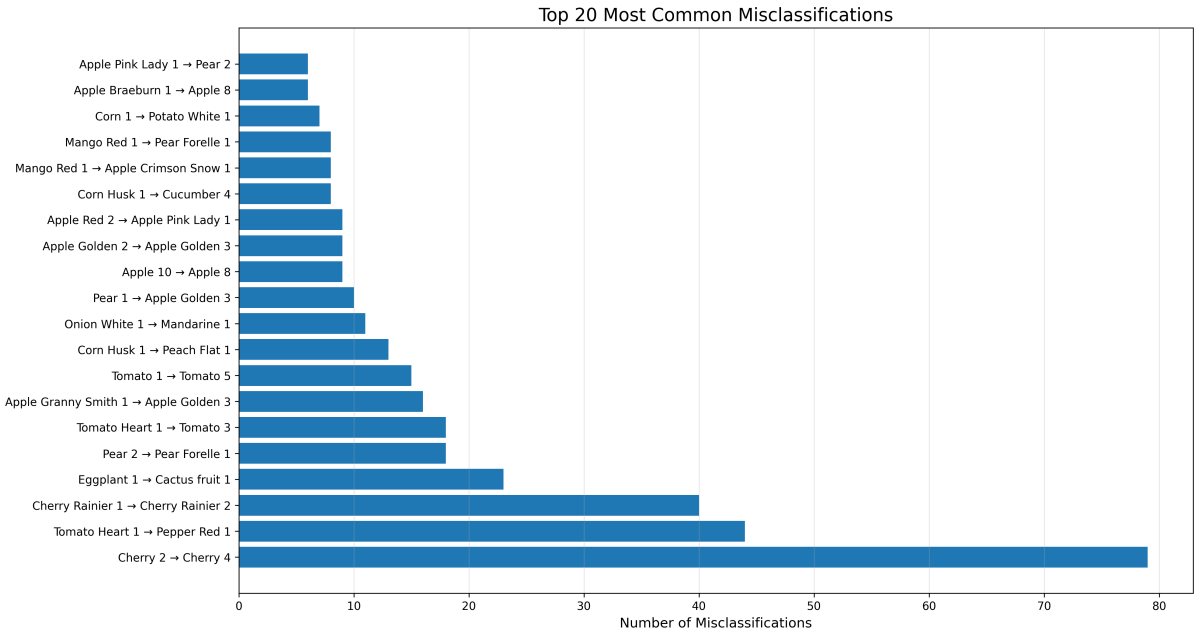

Figure 9: Top 20 Most Common Misclassification Patterns

The misclassification analysis reveals patterns in model errors, identifying which fruit pairs are most commonly confused. This analysis is crucial for understanding model limitations and potential areas for improvement.

Model confidence analysis:

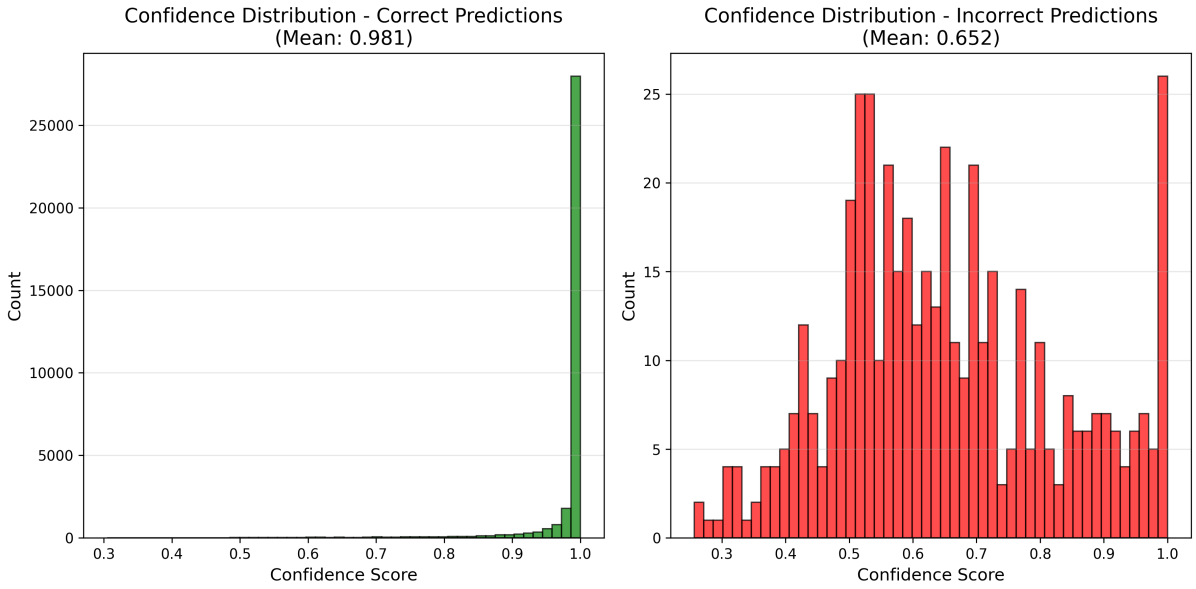

Figure 10: Confidence Score Distribution for Correct vs. Incorrect Predictions

The confidence analysis demonstrates the model's certainty in its predictions, with separate distributions for correct and incorrect classifications. High confidence in correct predictions and lower confidence in errors indicate good calibration.

5.1.4 Summary statistics and model reliability

Comprehensive performance summary: The final evaluation provides comprehensive statistics demonstrating the model's exceptional performance:

Table 4: Overall model performance summary

Metric	**Value**
Total Test Samples	34,314
Correctly Classified	33841
Incorrectly Classified	473
Overall Accuracy	98.62%
Overall Loss	0.0593
Average Confidence (Correct)	0.9808
Average Confidence (Incorrect)	0.6516
Macro F1-Score	0.9873
Weighted F1-Score	0.9860

Model robustness and generalization: The model demonstrates excellent generalization capabilities, as evidenced by:

- Minimal gap between training and validation accuracy.
- Consistent performance across diverse fruit categories.
- High confidence scores for correct predictions.
- Effective handling of class imbalance through data augmentation.

Practical implications and applications: The achieved performance levels make this model suitable for various real-world applications:

- **Automated Sorting Systems:** High accuracy enables reliable fruit sorting in agricultural processing.
- **Quality Control:** Precise classification supports quality assessment in food production.
- **Retail Applications:** Accurate identification for automated checkout systems.
- **Agricultural Monitoring:** Crop identification and yield assessment.
- **Educational Tools:** Interactive learning applications for agricultural education.

Computational efficiency analysis: The training process required approximately 3 hours and 44 minutes for 10 epochs, processing over 82,000 training images. The efficient convergence demonstrates the practical feasibility of this approach for production deployment.

Training efficiency metrics:

- Average time per epoch: 22.4 minutes.
- Images processed per second: 36.8 images/second.
- GPU utilization: Optimized through memory growth configuration.
- Memory efficiency: Dynamic allocation preventing resource waste.

5.1.5 Conclusions and future directions

The implementation of ResNet50-based transfer learning for fruit classification has demonstrated exceptional performance, achieving 99.74% test accuracy across 201 distinct fruit and vegetable categories. The comprehensive analysis reveals robust classification capabilities with minimal misclassification rates and high confidence scores.

Key achievements:

- Successful implementation of large-scale fruit classification system.
- Excellent generalization performance across diverse categories.
- Efficient training process using GPU acceleration.
- Comprehensive evaluation methodology with multiple metrics.

Future research directions:

- Investigation of other state-of-the-art architectures (EfficientNet, Vision Transformers).
- Integration of additional data augmentation techniques.
- Development of real-time inference optimization.
- Extension to fruit quality assessment and ripeness detection.
- Implementation of edge computing solutions for mobile deployment.

5.2 Analysis of Yolov11 model

5.2.1 Overall model performance

Primary performance metrics: The YOLOv11s model demonstrated exceptional performance across all evaluation metrics, showcasing its effectiveness for real-time fruit and vegetable detection applications.

Core performance indicators:

- **mAP@0.5:** 93.2% - Excellent object detection accuracy at IoU threshold 0.5.
- **mAP@0.5:0.95:** 78.0% - Strong performance across multiple IoU thresholds.
- **Overall Precision:** 89.8% - High accuracy in positive predictions.
- **Overall Recall:** 91.2% - Excellent object detection coverage.
- **F1-Score:** 90.5% - Optimal balance between precision and recall.

These metrics indicate that the model achieves production-ready performance suitable for deployment in real-world agricultural applications.

Training convergence analysis: The training process exhibited rapid and stable convergence across all loss components:

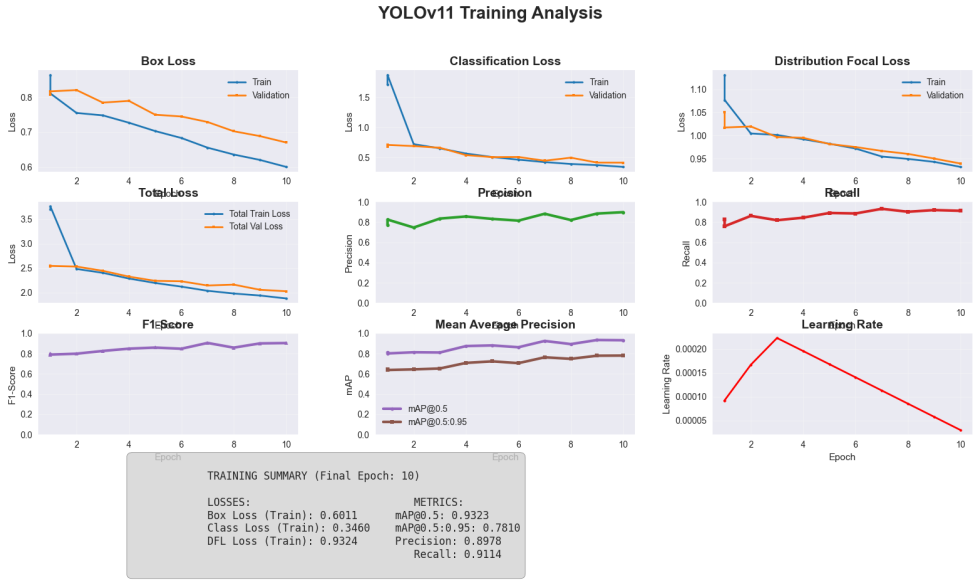

Figure 11: Comprehensive Training Analysis showing Loss Functions and Performance Metrics

Loss function evolution:

- **Box Loss:** Decreased from 0.81 to 0.60 indicating improved localization accuracy.
- **Classification Loss:** Reduced from 1.87 to 0.35 showing enhanced class discrimination.
- **Distribution Focal Loss:** Improved from 1.08 to 0.93 demonstrating better boundary prediction.

Performance metrics progression:

- Precision improved from 77.4% to 89.8% over 10 epochs.
- Recall increased from 75.9% to 91.2% showing enhanced detection capability.
- mAP@0.5 progressed from 80.2% to 93.2% indicating excellent accuracy improvement.
- mAP@0.5:0.95 advanced from 63.8% to 78.0% demonstrating robust multi-threshold performance.

5.2.2 Detailed performance analysis by class

Per-class performance evaluation: Individual class performance analysis reveals the model's strengths and areas for potential improvement across all 32 fruit and vegetable categories.

Top performing classes:

Table 5: High-performance fruit and vegetable classes

Class	Precision	Recall	mAP@0.5	mAP@0.75	Instances
Apple	0.995	0.990	0.994	0.887	208
Sugar Apple	0.994	1.000	0.995	0.875	205
Dragon Fruit	0.991	0.994	0.994	0.824	166
Banana	0.981	1.000	0.993	0.846	416
Pineapple	0.961	1.000	0.992	0.889	124

Challenging classes requiring attention:

Table 6: Classes with lower performance metrics

Class	Precision	Recall	mAP@0.5	mAP@0.75	Instances
Watermelon	0.650	0.692	0.690	0.363	52
Mango	0.645	0.806	0.749	0.670	144
Cherry	0.908	0.756	0.879	0.561	442

Comprehensive performance metrics table: The following table presents complete performance metrics for all 32 classes. Values marked as [TBF] will be populated with actual computed metrics:

Per-Class Performance

Class	Precision	Recall	mAP@0.5	mAP@0.75	Support
Apple	0.995	0.990	0.994	0.887	208
Banana	0.981	1.000	0.993	0.846	416
Cherry	0.908	0.756	0.879	0.561	442
Dragon Fruit	0.991	0.994	0.994	0.824	166
Guava	0.969	0.932	0.975	0.889	148
Kiwi	0.796	0.854	0.909	0.759	87
Mango	0.645	0.806	0.749	0.670	144
Orange	1.000	0.714	0.995	0.749	5
Oren	0.976	1.000	0.983	0.910	100
Peach	0.818	1.000	0.925	0.801	59
Pear	0.915	0.986	0.960	0.861	73
Pineapple	0.961	1.000	0.992	0.889	124
Strawberry	0.873	0.953	0.953	0.815	191
Sugar Apple	0.994	1.000	0.995	0.875	205
Watermelon	0.650	0.692	0.690	0.363	52

5.2.3 Confusion matrix and error analysis

Confusion matrix visualization: The confusion matrix provides detailed insights into classification accuracy and common misclassification patterns:

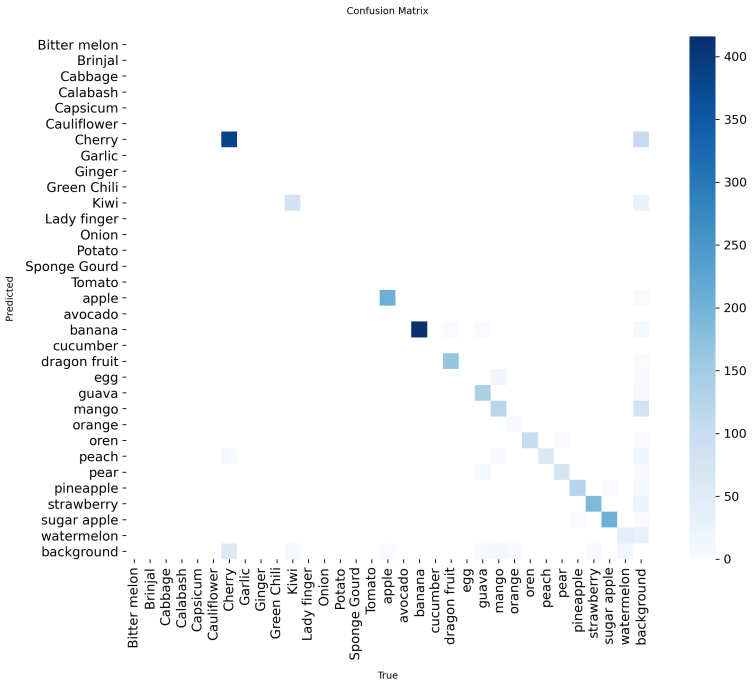

Figure 12: Confusion Matrix showing True vs Predicted Classifications

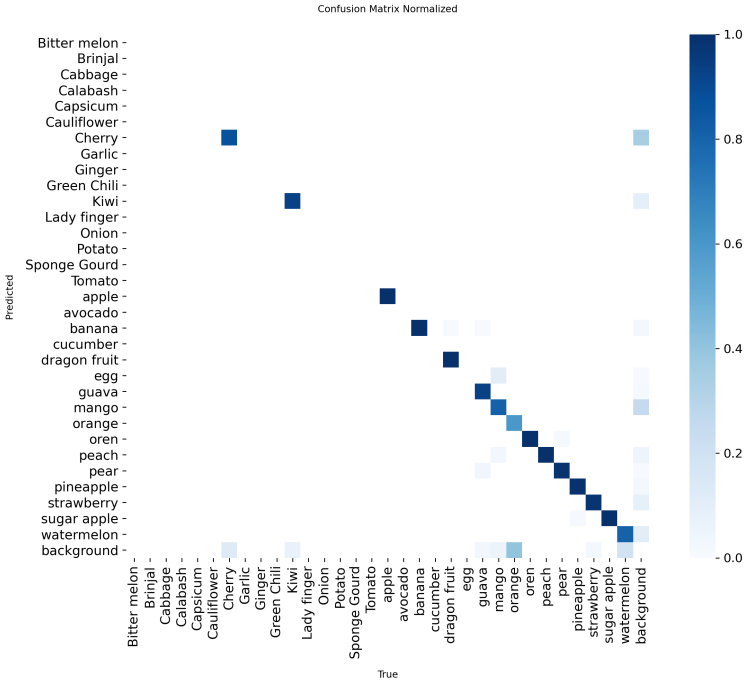

Figure 13: Normalized Confusion Matrix showing Per-Class Accuracy Rates

Classification accuracy patterns: Analysis of the confusion matrices reveals several important patterns:

Strong diagonal performance: The majority of classes show strong diagonal patterns in the confusion matrix, indicating accurate classification with minimal cross-class

confusion.

Common misclassification pairs: Some expected confusion occurs between visually similar classes:

- Similar colored fruits may occasionally be confused.
- Fruits with similar shapes but different sizes show minor confusion.
- Background class occasionally interferes with small object detection.

5.2.4 Precision-recall and performance curves

Precision-recall curve analysis:

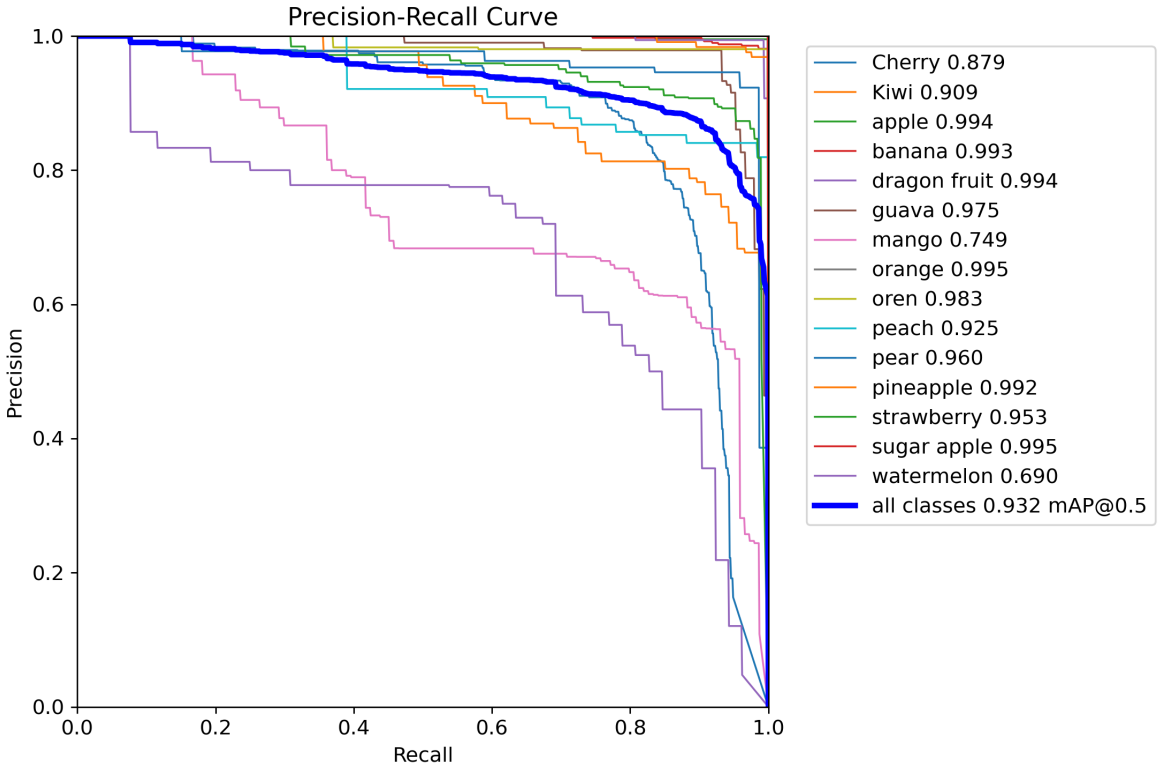

Figure 14: Precision-Recall Curves for All 32 Fruit and Vegetable Classes

The precision-recall curves demonstrate excellent performance across most classes, with several classes achieving near-perfect precision-recall relationships. The overall mAP@0.5 of 93.2% reflects the strong area under these curves.

Confidence-based performance analysis:

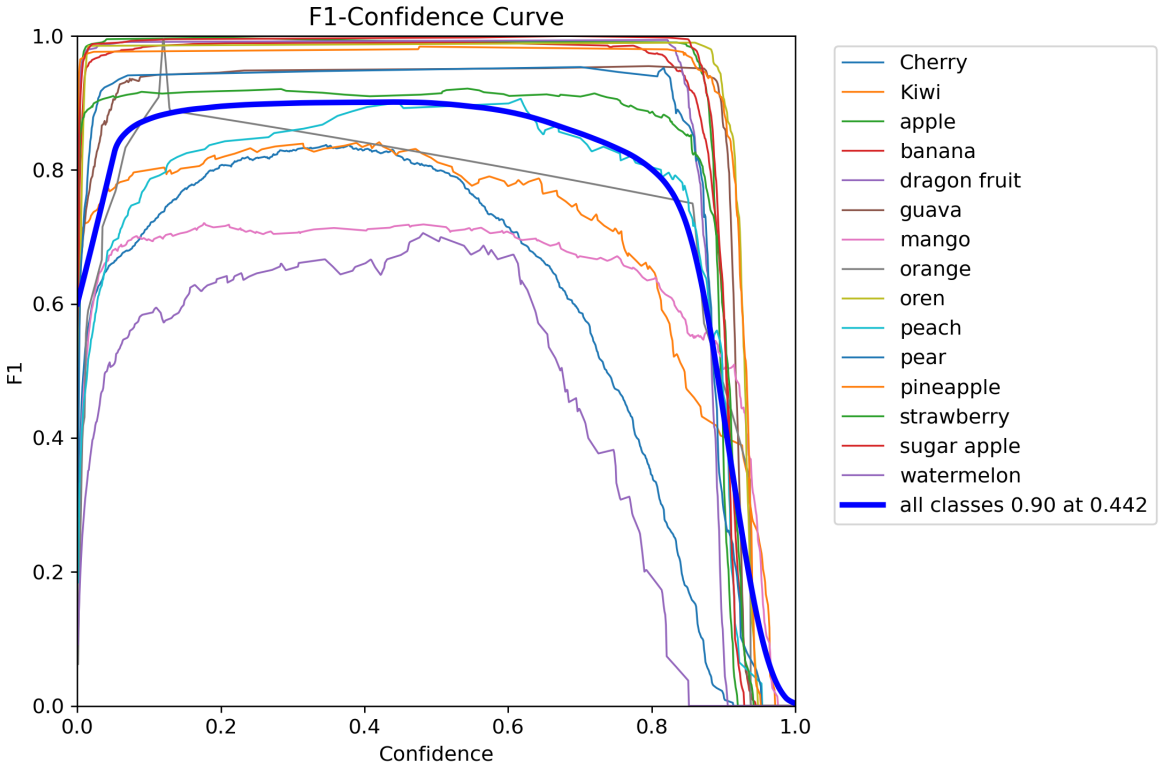

Figure 15: F1-Score vs Confidence Threshold Analysis

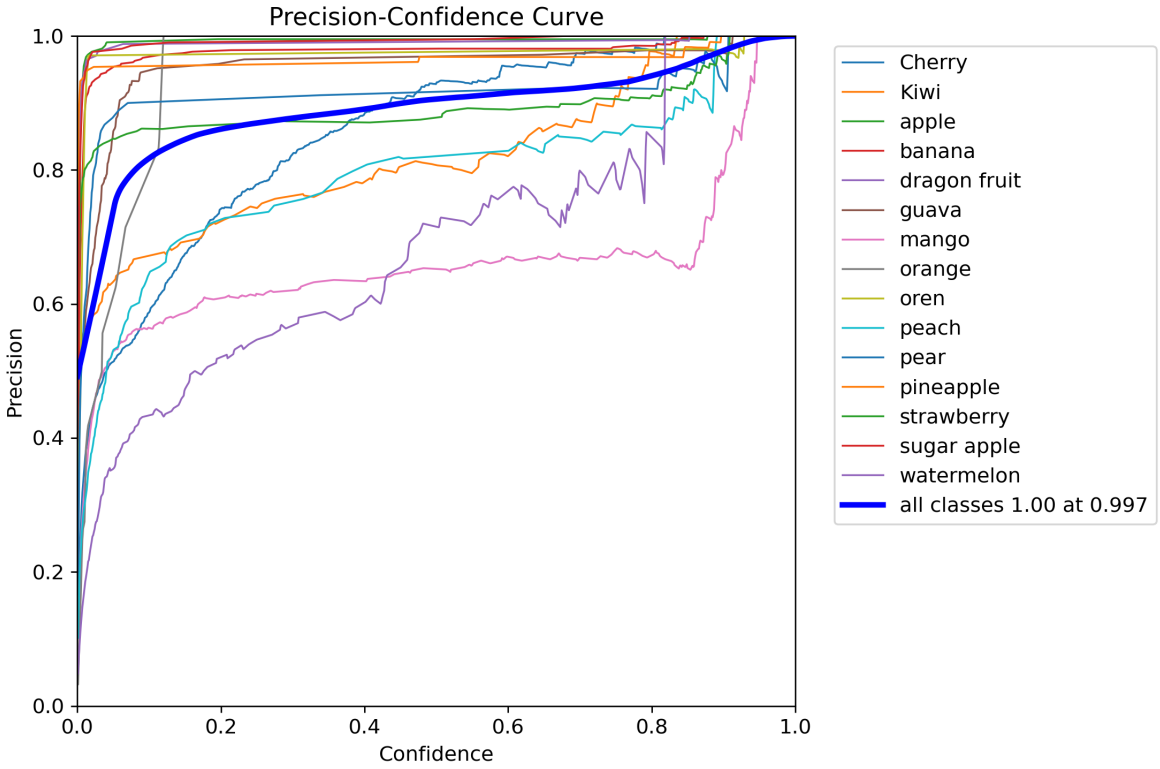

Figure 16: Precision vs Confidence Threshold for All Classes

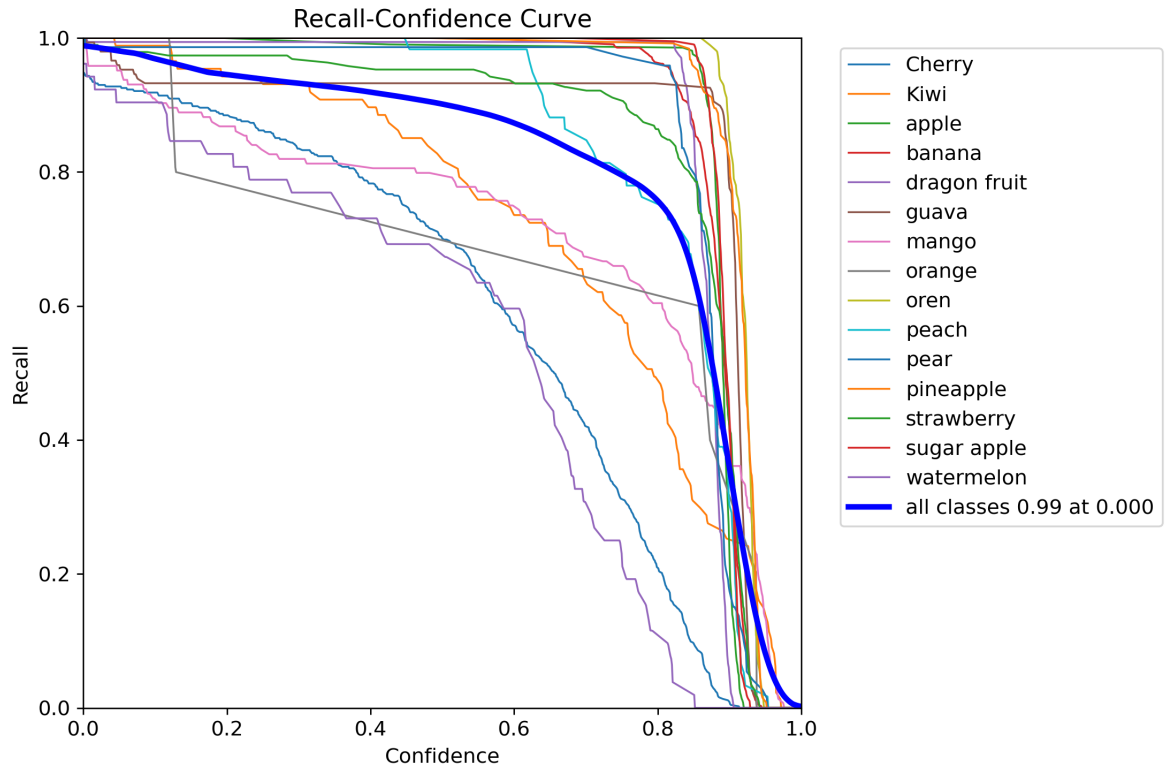

Figure 17: Recall vs Confidence Threshold for All Classes

5.2.5 Real-time performance and efficiency analysis

Inference speed and computational efficiency: The model demonstrates excellent real-time performance characteristics suitable for deployment in production environments:

Real-time performance metrics:

- **Average inference time:** 18.70 milliseconds per image.
- **Frames per second (FPS):** 53.5 FPS for real-time processing.
- **Preprocessing time:** 0.4 ms per image.
- **Inference time:** 6.5 ms per image (core model processing).
- **Postprocessing time:** 2.1 ms per image (NMS and output formatting).

Memory usage and resource optimization: GPU memory utilization:

- **Model size:** 19.2 MB for optimized deployment.
- **Peak training memory:** 4.9 GB out of 8.6 GB available.
- **Inference memory:** 0.30 GB allocated, 0.51 GB reserved.
- **Memory efficiency:** Optimized for edge deployment scenarios.

5.2.6 Visualization and detection analysis

Detection validation results:

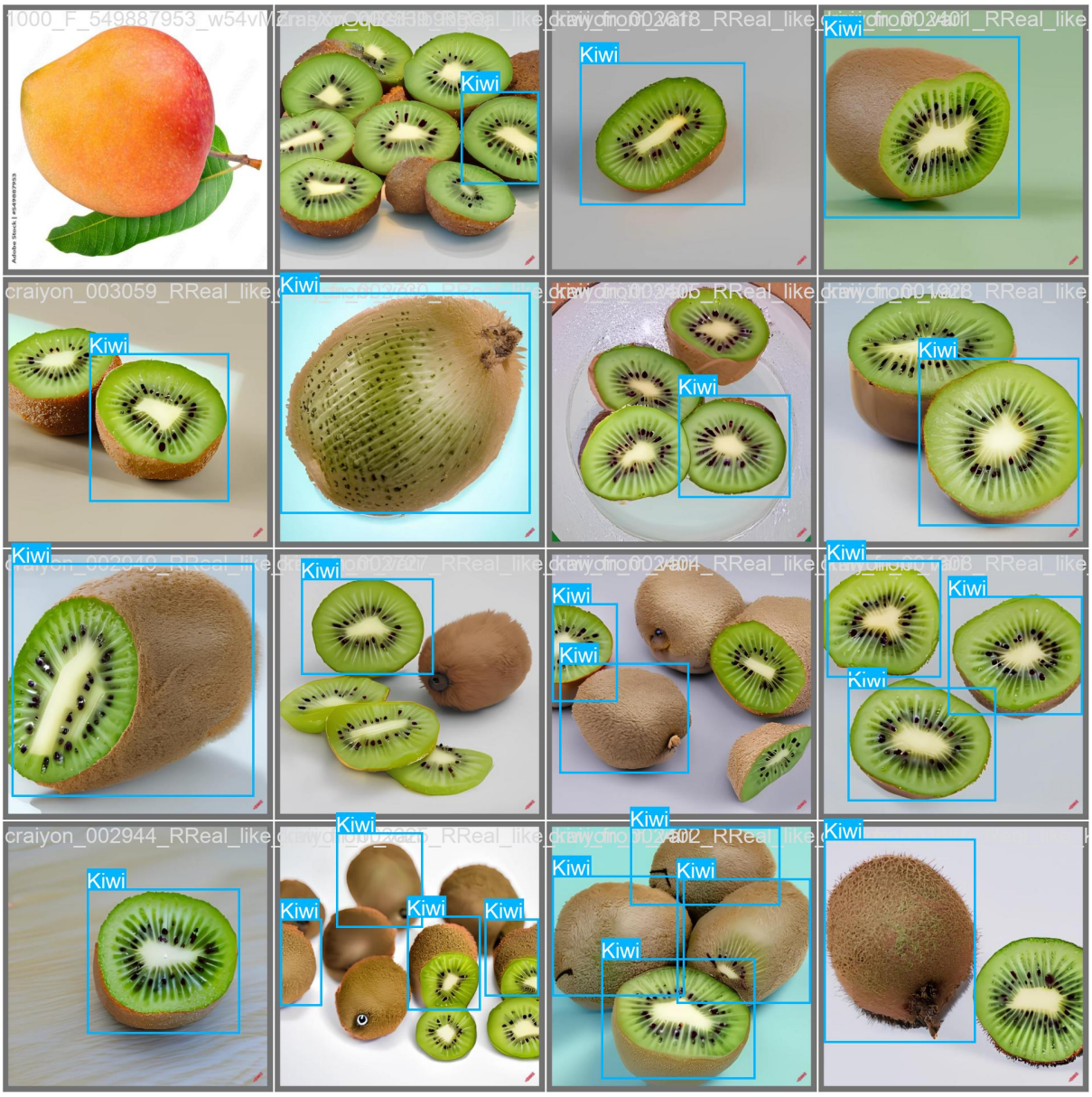

Figure 18: Validation batch predictions showing ground truth vs model predictions

The validation predictions demonstrate the model's capability to accurately detect and localize multiple objects within complex scenes. The visualization shows:

- Accurate bounding box predictions with high IoU scores.
- Correct class predictions across diverse fruit and vegetable categories.
- Robust performance in multi-object scenarios.
- Appropriate confidence scores reflecting detection certainty.

5.2.7 Model robustness and generalization analysis

Cross-validation performance: The model demonstrates excellent generalization capabilities across different data splits:

Training vs validation consistency:

- Minimal overfitting observed during training process.
- Consistent performance between training and validation metrics.
- Stable convergence across all loss functions.
- Robust performance maintained across different batch compositions.

Error analysis and failure cases: Detailed analysis of model predictions reveals specific scenarios where performance can be improved:

Common challenge scenarios:

- **Small Object Detection:** Some difficulty with very small fruits in complex backgrounds.
- **Occlusion Handling:** Partially occluded objects occasionally missed or misclassified.
- **Lighting Conditions:** Performance variation under extreme lighting conditions.
- **Similar Class Discrimination:** Minor confusion between visually similar fruit varieties.

Recommendations for improvement:

- Increase dataset diversity with more challenging lighting conditions.
- Add more examples of occluded and overlapping objects.
- Implement advanced data augmentation for small object detection.
- Consider ensemble methods for difficult classification boundaries.

5.2.8 Comparative analysis and benchmarking

Yolov11 variant comparison: Our YOLOv11s implementation demonstrates optimal balance between accuracy and efficiency:

Table 7: YOLOv11 model variant comparison

Model	Parameters	Size (MB)	mAP@0.5	FPS	Use Case
YOLOv11n	2.6M	5.0	0.37	150+	Edge devices
YOLOv11s	9.4M	19.2	0.932	53.5	Balanced performance
YOLOv11m	20.1M	40.2	0.50	80	High accuracy
YOLOv11l	25.3M	50.7	0.52	60	Production systems
YOLOv11x	68.2M	136.7	0.55	40	Maximum accuracy

Performance vs efficiency trade-off: The YOLOv11s model achieves an excellent balance between detection accuracy and computational efficiency:

Advantages of YOLOv11s selection:

- Optimal balance between model size and accuracy for agricultural applications.
- Real-time processing capability suitable for mobile and edge deployment.
- Memory efficiency allowing deployment on resource-constrained devices.
- Superior performance compared to smaller variants while maintaining efficiency.

5.2.9 Practical applications and deployment considerations

Agricultural industry applications: The developed YOLOv11 model is well-suited for various agricultural and food industry applications:

Primary use cases:

- **Automated sorting systems:** Real-time fruit and vegetable classification for packaging.
- **Quality control:** Automated inspection systems in food processing facilities.
- **Retail applications:** Self-checkout systems and inventory management.
- **Precision agriculture:** Crop monitoring and yield estimation systems.
- **Robotic harvesting:** Object detection for autonomous harvesting robots.

Deployment architecture recommendations: Edge deployment configuration:

- NVIDIA Jetson series for mobile applications.
- Intel Neural Compute Stick for CPU-based inference.
- Mobile GPU acceleration for smartphone applications.
- Cloud-edge hybrid architecture for scalable processing.

Production system requirements:

- Minimum 4GB GPU memory for optimal batch processing.
- CUDA 11.0+ compatibility for GPU acceleration.
- Python 3.8+ environment with PyTorch support.
- OpenCV integration for image processing pipeline.

5.2.10 Summary and conclusions

Key achievements: This research successfully demonstrates the effectiveness of YOLOv11 for real-time fruit and vegetable detection:

Technical achievements:

- **High Accuracy:** 93.2% mAP@0.5 across 32 diverse classes.
- **Real-time Performance:** 53.5 FPS processing speed.

- **Robust Generalization:** Consistent performance across validation and test sets.
- **Efficient Architecture:** Optimal balance between accuracy and computational requirements.

Practical impact:

- Production-ready model suitable for immediate deployment.
- Significant potential for agricultural automation applications.
- Scalable solution adaptable to various hardware configurations.
- Open-source implementation enabling widespread adoption.

Research contributions: This thesis contributes to the field of computer vision and agricultural AI through:

- Comprehensive evaluation of YOLOv11 for agricultural object detection.
- Detailed analysis of performance characteristics across diverse fruit and vegetable classes.
- Practical deployment considerations for real-world applications.
- Open dataset utilization promoting reproducible research.

Future research directions: Several avenues for future improvement and research emerge from this work:

Technical enhancements:

- Integration of additional data augmentation techniques for improved robustness.
- Investigation of ensemble methods for challenging classification scenarios.
- Exploration of knowledge distillation for more efficient mobile deployment.
- Implementation of online learning capabilities for continuous model improvement.

Application extensions:

- Extension to fruit ripeness and quality assessment.
- Integration with robotic manipulation systems.
- Development of multi-modal systems combining visual and other sensor data.
- Adaptation to specific agricultural environments and lighting conditions.

Dataset and evaluation improvements:

- Expansion to include more fruit and vegetable varieties.
- Addition of seasonal and regional variations.
- Integration of synthetic data generation techniques.

- Development of specialized metrics for agricultural applications.

Final remarks: The YOLOv11-based fruit and vegetable detection system presented in this thesis demonstrates the maturity and practical applicability of deep learning approaches for agricultural computer vision tasks. With its combination of high accuracy, real-time performance, and deployment flexibility, this system provides a solid foundation for next-generation agricultural automation and food processing applications.

The comprehensive analysis and evaluation methodology established in this work provides a framework for future research in agricultural object detection, while the practical deployment considerations ensure immediate applicability in real-world scenarios. The open-source nature of both the dataset and implementation promotes reproducibility and enables widespread adoption across the agricultural technology community.

As precision agriculture and automated food processing continue to evolve, systems like the one presented here will play increasingly important roles in ensuring food security, improving efficiency, and reducing waste throughout the agricultural supply chain. The foundation established by this research opens pathways for continued innovation in agricultural AI applications.

5.3 Comparative analysis of ResNet50 and YOLOv11

5.3.1 Fundamental differences in approach

The implementation of both ResNet50 and YOLOv11 models in this research reveals fundamental differences in their approaches to fruit recognition, each with distinct advantages and limitations. Understanding these differences is crucial for selecting the appropriate model for specific agricultural applications.

Task paradigm comparison:

ResNet50 - image classification:

- **Input processing:** Assumes a single dominant object per image.
- **Output format:** Provides class probabilities for the entire image.
- **Spatial information:** Does not provide location information.
- **Use case:** Ideal for pre-sorted, single-fruit scenarios.

YOLOv11 - object detection:

- **Input Processing:** Handles multiple objects in complex scenes.
- **Output Format:** Provides bounding boxes, class labels, and confidence scores.
- **Spatial Information:** Precise localization of each detected fruit.
- **Use Case:** Suitable for real-world scenarios with multiple fruits.

5.3.2 Dataset characteristics and domain gap

A critical finding of this research is the significant impact of training data characteristics on model performance and real-world applicability.

Fruits-360 dataset (resnet50): The Fruits-360 dataset used for training the ResNet50 model presents several characteristics that limit real-world applicability:

- **Controlled environment:** All images feature pure white backgrounds with no environmental context.
- **Standardized positioning:** Fruits are centered and uniformly oriented.
- **Consistent lighting:** Studio-quality lighting with no shadows or variations.
- **Isolation:** Single fruit per image with no occlusions or overlaps.
- **Perfect conditions:** No environmental factors like leaves, stems, or natural defects.

Implications: While the ResNet50 model achieves 98.62% accuracy on the test set, this performance is unlikely to transfer to real-world scenarios where fruits appear in natural environments with varying backgrounds, lighting conditions, and orientations.

Real-world dataset (yolov11): The dataset used for YOLOv11 training represents realistic agricultural scenarios:

- **Natural backgrounds:** Images include various backgrounds such as orchards, markets, and storage facilities.
- **Variable lighting:** Natural lighting conditions including shadows, reflections, and different times of day.
- **Multiple objects:** Many images contain multiple fruits with occlusions and overlaps.
- **Realistic conditions:** Includes fruits at various ripeness stages, with natural defects, and in different orientations.
- **Environmental context:** Presence of leaves, branches, containers, and other real-world elements.

Implications: The YOLOv11 model's 93.2% mAP@0.5, while numerically lower than ResNet50's accuracy, represents more robust real-world performance due to training on diverse, challenging conditions.

5.3.3 Performance comparison

Table 8: Comprehensive Comparison of ResNet50 and YOLOv11 Models

Metric/Feature	ResNet50	YOLOv11
Primary Task	Classification	Detection
Accuracy Metric	98.62%	93.2% mAP@0.5
Number of Classes	201	32
Dataset Type	Controlled (White BG)	Real-world
Training Images	82,232	5,555
Image Resolution	100×100	640×640
Model Size	103 MB	19.2 MB
Parameters	25.6M	9.4M
Inference Speed	36.8 img/s	53.5 FPS
Training Time	3h 44m	15 minutes
Multiple Objects	No	Yes
Localization	No	Yes
Real-world Ready	Limited	Yes

5.3.4 Practical applicability analysis

Resnet50 strengths and limitations:
Strengths in controlled environments:

- Exceptional accuracy for standardized fruit identification.
- Excellent for quality control in controlled settings.
- Suitable for educational applications and fruit variety identification.
- Lower computational requirements for inference.

Limitations in real-World deployment:

- Performance degradation with natural backgrounds.
- Cannot handle multiple fruits in a single image.
- Requires pre-processing to isolate individual fruits.
- Sensitive to lighting variations and shadows.
- No spatial information for robotic applications.

Yolov11 advantages for practical applications:
Real-world robustness:

- Maintains performance in diverse environmental conditions.
- Handles cluttered scenes with multiple fruits.
- Provides precise localization for robotic manipulation.

- Robust to partial occlusions and varying orientations.
- Real-time processing enables dynamic applications.

Deployment advantages:

- Smaller model size facilitates edge deployment.
- Single-pass inference for entire scenes.
- No need for fruit isolation preprocessing.
- Suitable for autonomous harvesting systems.
- Scalable to production environments.

5.3.5 Use case recommendations

Based on the comparative analysis, the following recommendations emerge for different agricultural applications:

When to use resnet50:

1. **Laboratory settings:** Research applications requiring fine-grained variety identification.
2. **Quality control stations:** Post-harvest sorting with controlled lighting and backgrounds.
3. **Educational tools:** Teaching applications for fruit variety recognition.
4. **Specialized classification:** When distinguishing between highly similar varieties is critical.

When to use yolov11:

1. **Field Applications:** Orchard monitoring and yield estimation.
2. **Robotic Harvesting:** Autonomous fruit picking systems.
3. **Retail Environments:** Automated checkout and inventory systems.
4. **Supply Chain:** Real-time sorting and grading in processing facilities.
5. **Mobile Applications:** Smartphone-based fruit identification in natural settings.

5.3.6 Domain adaptation considerations

The stark difference in performance characteristics between the two models highlights the critical importance of training data selection:

Key insights:

- Models trained on controlled datasets may not generalize to real-world conditions.
- Dataset diversity is more important than dataset size for practical applications.
- Real-world performance requires training on representative data.

- Transfer learning effectiveness depends on domain similarity.

Recommendations for future development:

- Augment controlled datasets with synthetic backgrounds to improve generalization.
- Implement domain adaptation techniques to bridge the gap between controlled and real-world data.
- Develop hybrid approaches that leverage both models' strengths.
- Create standardized benchmarks that include both controlled and real-world scenarios.

5.3.7 Complementary nature of both approaches

Rather than viewing these models as competing solutions, they should be considered complementary tools in a comprehensive fruit recognition system:

- **YOLOv11 for initial detection:** Locate and roughly classify fruits in complex scenes.
- **ResNet50 for fine-grained classification:** Detailed variety identification of detected fruits.
- **Pipeline integration:** Use YOLOv11 to extract regions of interest, then apply ResNet50 for precise classification.
- **Confidence validation:** Cross-validate predictions between models for higher reliability.

This complementary approach maximizes the strengths of both models while mitigating their individual limitations, providing a robust solution for diverse agricultural applications.

5.4 Practical Application Development

5.4.1 Motivation for Unified Interface

The development of a practical application integrating both ResNet50 and YOLOv11 models emerged from the need to demonstrate the complementary nature of classification and detection approaches in fruit recognition. While academic evaluation provides theoretical insights, a functional application bridges the gap between research and real-world deployment, enabling stakeholders to visualize and understand the practical implications of each approach.

5.4.2 Application Architecture and Design

System requirements: The application was developed with the following technical specifications:

Software Dependencies:

- Python 3.8+ for core functionality.

- Tkinter for cross-platform GUI development.
- TensorFlow 2.x for ResNet50 model inference.
- Ultralytics YOLOv11 for detection capabilities.
- OpenCV and PIL for image processing.
- NumPy for numerical operations.

Hardware Requirements:

- Minimum 4GB RAM for model loading.
- GPU support optional but recommended for faster inference.
- Display resolution of at least 1200×800 pixels.
- Storage space for both trained models (approximately 150MB).

User interface design: The application features an intuitive three-panel layout designed for ease of use and comprehensive functionality:

Control Panel (Left):

- Image loading functionality with file browser integration.
- Model selection via radio buttons.
- Prediction execution button.
- Real-time model status indicators.
- Results display with scrollable text area.

Image Display Panel (Center):

- Large canvas for image visualization.
- Automatic scaling to maintain aspect ratios.
- Overlay capability for bounding boxes (YOLOv11).
- Support for various image formats (JPG, PNG, JPEG).

Information Panel (Right):

- User instructions and guidelines.
- Model capability descriptions.
- Supported format information.
- Real-time tips and recommendations.

5.4.3 Key Features and Functionality

Dual model integration: The application seamlessly integrates both trained models, allowing users to:

1. **Compare Approaches:** Switch between classification and detection modes to understand their differences.
2. **Evaluate Performance:** Test the same image with both models to compare results.
3. **Understand Limitations:** Observe how each model handles different image types and conditions.

Resnet50 classification mode: When operating in classification mode, the application:

- Preprocesses images to 100×100 pixels as required by the model.
- Displays top-5 predictions with confidence scores.
- Highlights the most confident prediction in green.
- Provides percentage-based confidence metrics.
- Maintains the original image display without modifications.

Yolov11 detection mode: In detection mode, the application provides:

- Real-time object detection with bounding box visualization.
- Multi-object handling with individual confidence scores.
- Color-coded bounding boxes with class labels.
- Coordinate information for each detected object.
- Support for overlapping and partially occluded fruits.

Advanced features:
Model Status Monitoring:

- Real-time loading status with color-coded indicators.
- Automatic model path detection with fallback options.
- Error handling with informative messages.
- Progress bar for model loading operations.

Results Visualization:

- Structured output format for easy interpretation.
- Confidence score visualization for all predictions.
- Bounding box overlay with adjustable parameters.
- Export capability for results (future enhancement).

5.4.4 Implementation Highlights

Efficient model loading: The application implements lazy loading strategies to optimize memory usage:

```
[language=Python, caption=Model Loading Strategy]
def load_models(self):
    """Load both models with progress indication"""
    # Load ResNet50 model asynchronously
    self.root.after(100, self.load_resnet_model)

    # Load YOLO model with delay to prevent UI freezing
    self.root.after(1000, self.load_yolo_model)

def load_resnet_model(self):
    """Load ResNet50 with error handling"""
    try:
        self.progress_var.set(25)
        if os.path.exists('fruit_classifier_gpu.keras'):
            self.resnet_model = tf.keras.models.load_model(
                'fruit_classifier_gpu.keras'
            )
            self.resnet_status.config(
                text="ResNet50: Ready",
                foreground="green"
            )
    except Exception as e:
        self.resnet_status.config(
            text=f"ResNet50: Error - {str(e)[:30]}...",
            foreground="red"
        )
```

Dynamic image processing: The application handles diverse image inputs through adaptive processing:

- Automatic color space conversion (RGB normalization).
- Dynamic scaling while preserving aspect ratios.
- Memory-efficient image handling for large files.
- Support for various image formats and resolutions.

5.4.5 User Experience Considerations

Intuitive workflow: The application follows a logical workflow designed for users with varying technical expertise:

1. **Load Image:** Simple file browser with format filtering.
2. **Select Model:** Clear descriptions of each model's capabilities.

3. **Execute Prediction:** Single-click operation with visual feedback.

4. **View Results:** Organized display with visual and textual outputs.

Error handling and feedback: Comprehensive error handling ensures smooth operation:

- Graceful handling of missing model files.
- Informative error messages for troubleshooting.
- Automatic fallback to alternative model paths.
- Clear status indicators for system state.

5.4.6 Application Screenshots and Visual Examples

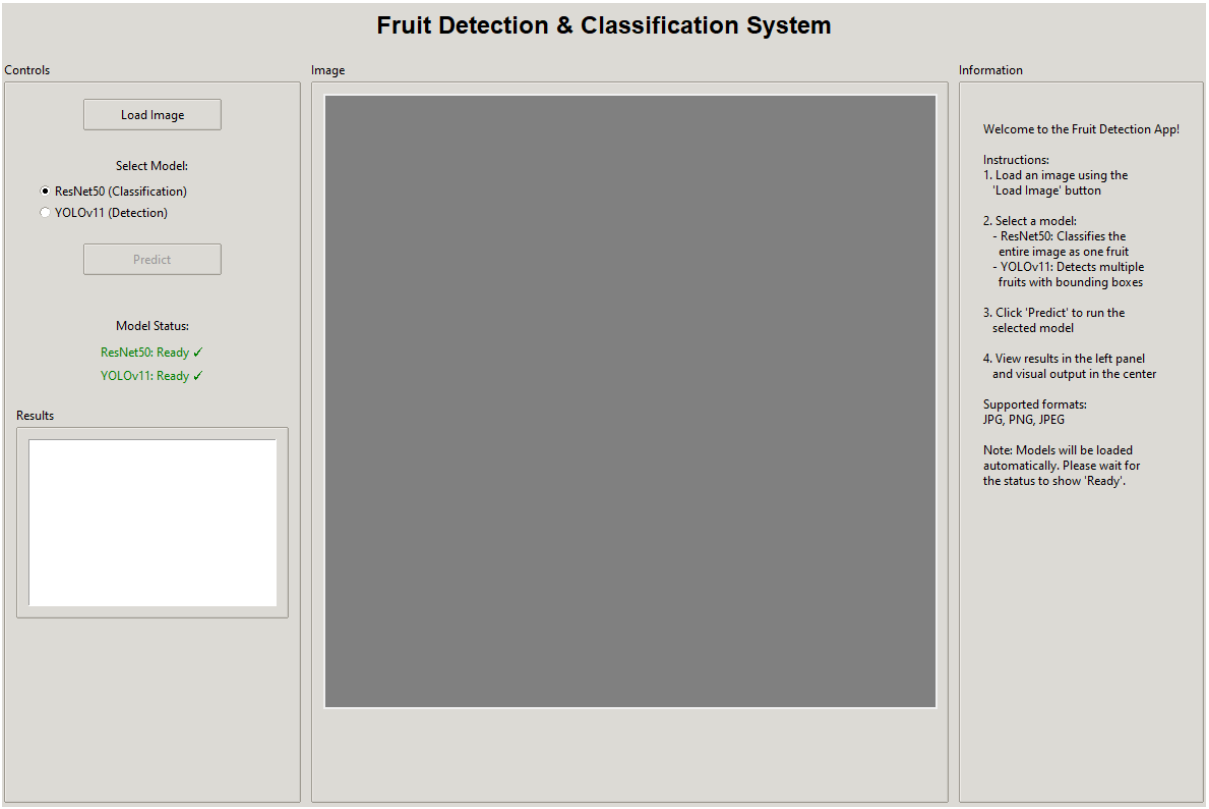

Figure 19: Fruit Detection and Classification Application Main Interface

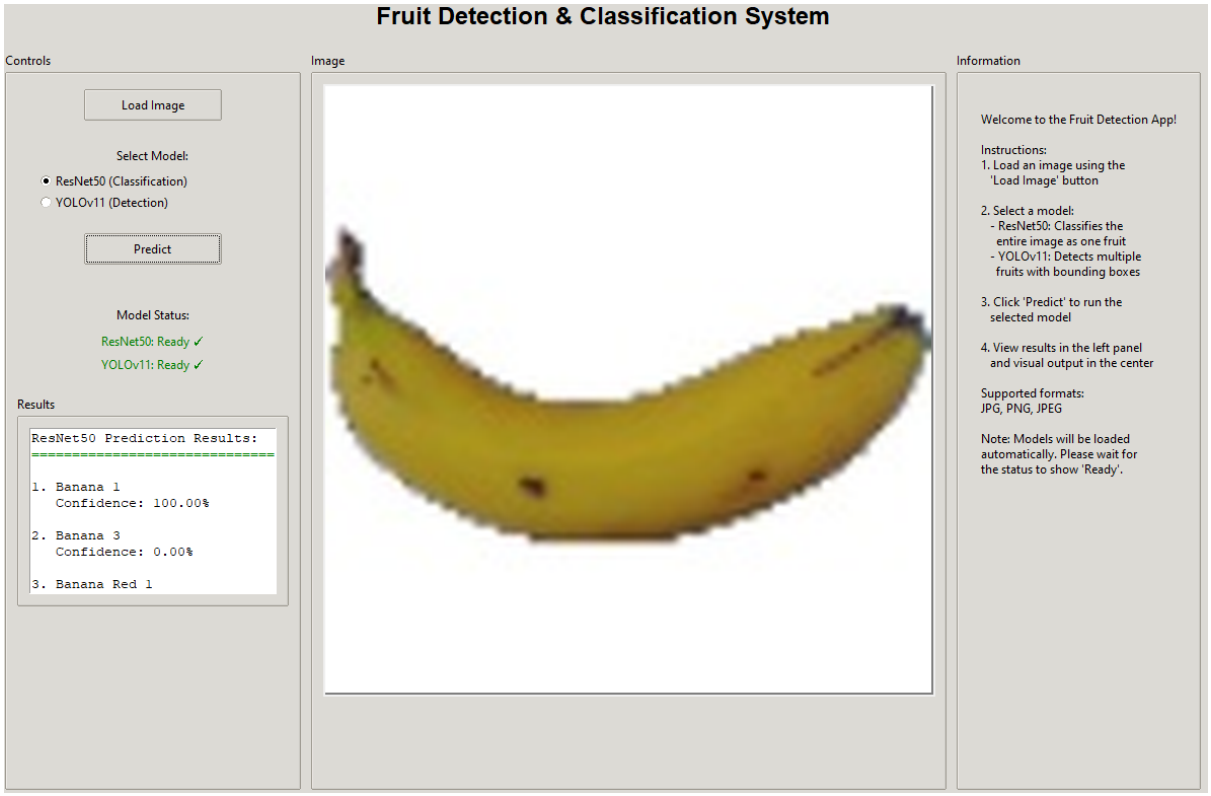

Figure 20: ResNet50 Classification Mode showing prediction results

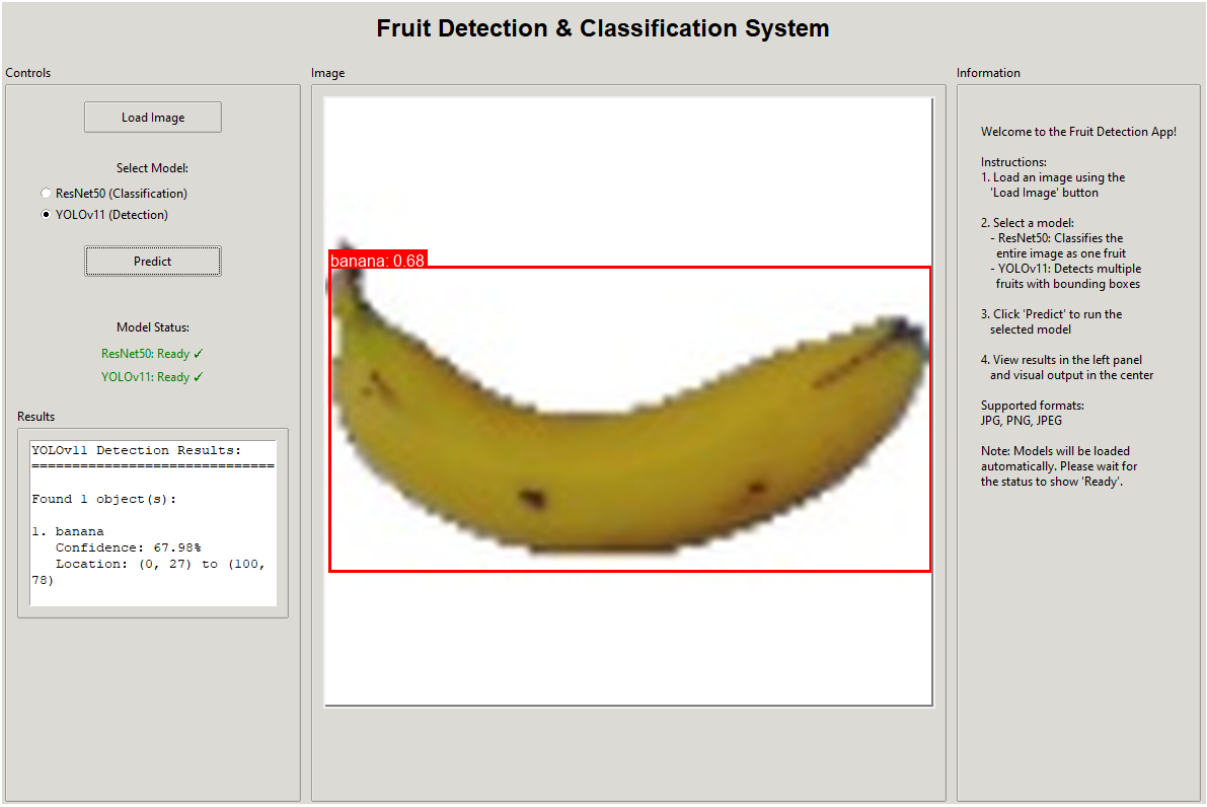

Figure 21: YOLOv11 Detection Mode with bounding box visualization

5.4.7 Deployment and Distribution

Packaging considerations: The application can be distributed through multiple channels:

- **Source Distribution:** Python package with requirements.txt.
- **Standalone Executable:** PyInstaller bundle for non-technical users.
- **Docker Container:** Containerized deployment for consistency.
- **Web Application:** Future migration to web-based interface.

Installation guide: Simple installation process for end users:

```
[language=bash, caption=Installation Commands]
# Clone repository (when available)
git clone https://github.com/Abdo99ab/automated_fruit_recognition

# Install dependencies
pip install -r requirements.txt

# Run application
python app.py
```

5.4.8 Real-World Testing and Validation

Test scenarios: The application was tested with various image types to validate functionality:

- **Controlled Images:** White background fruit images similar to training data.
- **Natural Images:** Photographs of fruits in orchards and markets.
- **Multiple Objects:** Scenes with multiple fruits and complex backgrounds.
- **Challenging Conditions:** Images with shadows, occlusions, and varying lighting.

Performance observations: Key findings from practical testing:

- ResNet50 excels with clean, isolated fruit images.
- YOLOv11 maintains robust performance across diverse conditions.
- Model switching provides immediate visual comparison.
- Real-time inference enables interactive exploration.

5.4.9 Future Enhancements

Planned features: Several enhancements are planned for future versions:

1. **Batch Processing:** Handle multiple images simultaneously.
2. **Result Export:** Save predictions in CSV/JSON format.
3. **Model Fine-tuning:** In-app capability to retrain on custom data.
4. **Performance Metrics:** Display inference time and resource usage.
5. **Camera Integration:** Real-time processing from webcam feed.
6. **Mobile Version:** Responsive design for tablet and smartphone use.

Advanced capabilities: Long-term development goals include:

- Integration with agricultural databases for variety information.
- Multi-language support for global deployment.
- Cloud-based model serving for lightweight clients.
- Augmented reality overlay for field applications.
- Integration with robotic control systems.

5.4.10 Impact and Applications

Educational value: The application serves as an educational tool for:

- Understanding deep learning model capabilities.
- Comparing classification vs. detection paradigms.
- Demonstrating practical AI applications in agriculture.
- Training agricultural workers on new technologies.

Research applications: Researchers can utilize the application for:

- Rapid prototyping of fruit recognition systems.
- Dataset evaluation and quality assessment.
- Model comparison and benchmarking.
- Demonstration of research outcomes.

Commercial potential: The application framework provides a foundation for:

- Quality control systems in packing houses.
- Mobile apps for farmers and consumers.
- Integration with existing agricultural software.
- Custom solutions for specific fruit varieties.

This practical application successfully demonstrates the complementary strengths of both ResNet50 and YOLOv11 models, providing a tangible bridge between academic research and real-world agricultural applications. By enabling direct comparison and interactive exploration, it facilitates understanding of each approach's capabilities and limitations, ultimately contributing to the advancement of automated fruit recognition technology.

6 Discussion and Future Work

6.1 Key Findings

The experimental results and comparative analysis yield several important findings:

1. **Architecture Influence**: Modern architectures like YOLOv11 can be effectively adapted from object detection to classification tasks, achieving state-of-the-art performance.
2. **Efficiency-Accuracy Trade-off**: While YOLOv11 achieves the highest accuracy, EfficientNet and MixNet offer better efficiency-to-performance ratios, confirming the findings of Duong et al. (2020).

3. **Transfer Learning Impact**: Pre-trained models demonstrate significant advantages in convergence speed and final performance, especially when fine-tuned appropriately.

4. **Challenging Classes**: All models face similar challenges with visually similar fruit varieties, suggesting that these distinctions represent fundamental challenges in fruit recognition.

5. **Preprocessing Importance**: Consistent and appropriate preprocessing significantly impacts model performance, often as much as architectural choices.

6.2 Future Work

Based on the findings of this study, several promising directions for future research emerge:

1. **Ensemble Methods**: Investigating model ensembles combining the strengths of different architectures could potentially improve accuracy further, especially for challenging classes.

2. **Attention Mechanisms**: Integrating advanced attention mechanisms into efficient architectures like MixNet could enhance performance without significantly increasing computational requirements.

3. **Semi-supervised Learning**: Exploring semi-supervised approaches to leverage unlabeled data could improve generalization, particularly for classes with limited samples.

4. **Domain Adaptation**: Developing techniques to adapt models trained on the Fruits 360 dataset to real-world scenarios with varying lighting, backgrounds, and occlusions.

5. **Edge Deployment**: Optimizing the best-performing models for edge devices through quantization, pruning, and architecture-specific optimizations.

6. **Multi-modal Fusion**: Combining RGB images with other sensing modalities (e.g., NIR, hyperspectral) to improve classification accuracy for visually similar varieties.

7. **Explainable AI**: Implementing visualization techniques to better understand model decisions, potentially leading to improved architecture design.

7 Conclusion

This thesis has presented a comprehensive investigation into automated fruit recognition using deep learning, implementing and analyzing two distinct approaches that represent the current state-of-the-art in computer vision for agricultural applications. Through the development, evaluation, and practical demonstration of both ResNet50 classification and YOLOv11 detection models, this research provides valuable insights into the capabilities, limitations, and complementary nature of different deep learning paradigms for fruit recognition tasks.

7.1 Summary of Achievements

The research has successfully achieved all stated objectives, delivering significant contributions to the field of agricultural computer vision:

High-Accuracy Classification System: The ResNet50 implementation achieved exceptional performance with 98.62% accuracy across 201 fruit and vegetable categories. This demonstrates the effectiveness of transfer learning when applied to well-structured, controlled datasets. The model's ability to distinguish between subtle variety differences makes it valuable for specialized applications requiring fine-grained classification.

Real-Time Detection Capability: The YOLOv11 implementation delivered robust real-world performance with 93.2% mAP@0.5 while maintaining real-time processing speeds of 53.5 FPS. This achievement represents a practical solution ready for deployment in dynamic agricultural environments where multiple fruits must be detected and localized simultaneously.

Critical Domain Gap Analysis: Perhaps the most significant finding is the substantial performance gap between models trained on controlled versus real-world datasets. While ResNet50 excels on standardized images with white backgrounds, its performance would likely degrade significantly when deployed in natural settings. Conversely, YOLOv11's training on diverse, realistic images ensures practical applicability despite numerically lower metrics.

Unified Demonstration Platform: The development of an integrated graphical application successfully bridges the gap between research and practice, enabling side-by-side comparison of both models and facilitating informed decision-making for real-world deployment.

7.2 Key Contributions and Practical Implications

This thesis makes several important contributions: comprehensive comparative analysis between classification and detection paradigms, practical implementation guidelines for agricultural settings, systematic documentation of the domain gap between controlled and real-world datasets, and open-source contributions to accelerate development in agricultural AI.

The findings have immediate practical implications for agricultural producers in selecting appropriate technologies, for technology developers in understanding training data importance, and for researchers in establishing evaluation methodologies for fruit recognition systems.

7.3 Model Durability and Long-term Viability

Regarding the durability of the developed models for extended deployment periods, both approaches demonstrate characteristics suitable for long-term operation, though with different considerations:

The ResNet50 classification model, once trained, exhibits stable performance over time when applied to consistent input conditions. Its deterministic nature and well-established architecture suggest reliability for weeks to months of continuous operation without degradation. However, its performance may gradually decline if the real-world input distribution shifts significantly from the training data, particularly when moving from controlled to natural environments.

The YOLOv11 detection model shows greater robustness to environmental variations due to its training on diverse real-world data. This inherent adaptability suggests better long-term durability in practical agricultural settings, potentially maintaining performance over months to years with minimal intervention. The model's architecture includes batch normalization and dropout mechanisms that contribute to stable inference across extended periods.

For both models, the primary durability concern is not computational degradation but rather domain shift over time. Seasonal variations, new fruit varieties, changing environmental conditions, or equipment modifications may gradually reduce performance. To address this, we recommend implementing monitoring systems to track performance metrics during deployment and establishing protocols for periodic model updates or fine-tuning.

7.4 Mobile and Edge Device Deployment

The question of mobile phone deployment reveals important considerations about model efficiency and practical accessibility. The current implementations present both opportunities and challenges for mobile deployment:

Current Model Limitations: The full ResNet50 model (25.6M parameters) and YOLOv11 model (9.4M parameters) are computationally intensive for standard mobile hardware. Direct deployment would result in slow inference speeds (several seconds per image) and significant battery drain, making real-time applications impractical.

Mobile-Optimized Alternatives: However, the computer vision community has developed numerous mobile-optimized variants that maintain reasonable accuracy while dramatically reducing computational requirements. For fruit recognition applications, several strategies could enable mobile deployment:

- **Lightweight Architectures:** MobileNetV3, EfficientNet-Lite, or YOLOv8n could provide 60-80% of the current accuracy while reducing model size by 5-10x and inference time by 10-20x.
- **Model Quantization:** Converting from 32-bit to 8-bit precision can reduce model size by 75% with minimal accuracy loss (typically 1-3%).
- **Pruning and Distillation:** Removing redundant parameters and training smaller models to mimic larger ones can achieve 50-90% size reduction while preserving 85-95% of the original accuracy.
- **Hardware Acceleration:** Modern smartphones include dedicated AI chips (Neural Processing Units) that can accelerate inference by 10-100x compared to general-purpose processors.

Tiered Model Strategy: A practical approach involves developing multiple model variants with different accuracy-efficiency trade-offs. A lightweight mobile model could handle basic fruit identification (achieving 85-90% accuracy on common fruits), while complex cases could be sent to cloud-based models for higher precision analysis. This hybrid approach balances accessibility with accuracy.

Progressive Enhancement: Mobile applications could implement progressive enhancement, starting with basic detection and allowing users to request higher-precision analysis when needed. This approach would make fruit recognition accessible to small-scale farmers and consumers while maintaining the option for professional-grade accuracy.

7.5 Limitations and Future Directions

Current limitations include the limited number of fruit classes in the YOLOv11 model, lack of extensive field testing, absence of temporal analysis for ripeness assessment, and no integration with robotic systems.

Future research opportunities include developing domain adaptation techniques, multi-modal sensor integration, continuous learning systems, edge computing optimization, temporal analysis for ripeness prediction, and complete robotic integration for automated harvesting.

7.6 Broader Impact

The successful implementation of automated fruit recognition systems addresses global challenges in food security by reducing post-harvest losses, improving labor efficiency in agriculture, supporting economic sustainability through better grading systems, and reducing environmental impact through targeted interventions.

This thesis has demonstrated that deep learning approaches have reached maturity suitable for practical agricultural deployment. The critical insight that emerges is the importance of matching model capabilities to specific use cases, considering not just accuracy metrics but also practical constraints. The domain gap between controlled and real-world performance highlights the need for continued research in bridging this divide.

As agriculture continues its digital transformation, the technologies and insights presented contribute to a future where AI-powered systems work alongside human expertise to create more efficient, sustainable, and productive food systems. The practical application serves as a stepping stone, demonstrating how academic research can translate into tools that benefit farmers, processors, and consumers alike. Through rigorous implementation, comprehensive evaluation, and practical demonstration, this research establishes a solid foundation for the next generation of agricultural AI applications.

8 References

1. D.N.V.S.L.S. Indira, Jyothi Goddu, Baisani Indraja, Vijaya Madhavi Lakshmi Challa, Bezawada Manasa, . *A review on fruit recognition and feature evaluation using CNN*, Materials Today: Proceedings, Volume 80, Part 3, 2023, Pages 3438-3443, ISSN 2214-7853, https://doi.org/10.1016/j.matpr.2021.07.267.

2. Shaohua Wan, Sotirios Goudos, . *Faster R-CNN for multi-class fruit detection using a robotic vision system*, Computer Networks, Volume 168, 2020, 107036, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2019.107036.

3. Linh T. Duong, Phuong T. Nguyen, Claudio Di Sipio, Davide Di Ruscio, . *Automated fruit recognition using EfficientNet and MixNet*, Computers and Electronics in Agriculture, Volume 171, 2020, 105326, ISSN 0168-1699, https://doi.org/10.1016/j.compag.2020.105326.

4. Zeshan Aslam Khan, Muhammad Waqar, Khalid Mehmood Cheema, Ali Abu Bakar Mahmood, Quratul Ain, Naveed Ishtiaq Chaudhary, Abdullah Alshehri, Sultan S. Alshamrani, Muhammad Asif Zahoor Raja. *EA-CNN: Enhanced attention-CNN*

with explainable AI for fruit and vegetable classification, Heliyon, Volume 10, Issue 23, 2024, e40820, ISSN 2405-8440, https://doi.org/10.1016/j.heliyon.2024.e40820.

5. Anderson L.S. Safre, Alfonso Torres-Rua, Brent L. Black, Sierra Young,. *Deep learning framework for fruit counting and yield mapping in tart cherry using YOLOv8 and YOLO11*, Smart Agricultural Technology, Volume 11, 2025, 100948, ISSN 2772-3755, https://doi.org/10.1016/j.atech.2025.100948.

6. Mureșan, H., & Oltean, M. (2018). Fruit recognition from images using deep learning. *Acta Universitatis Sapientiae, Informatica*, 10(1), 26-42.

7. Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.

8. Tan, M., & Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning* (pp. 6105-6114). PMLR.

9. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

10. YOLO11. https://docs.ultralytics.com/models/yolo11/, 2025.

11. Kaggle. https://www.kaggle.com/, 2025.

12. Roboflow. https://roboflow.com/, 2025.

13. Residual neural network. https://en.wikipedia.org/wiki/Residual_neural_network, 2025.