Fruit Detection System Documentation

Chosen Field and Motivation

The chosen field for this project is **fruit detection and classification**, specifically focusing on identifying and classifying common fruits such as apples, bananas, grapes, mangoes, and strawberries in static images. The motivation behind this project stems from the growing need for automated systems in agriculture, food processing, and retail industries. Accurate fruit detection can streamline quality control, inventory management, and automated harvesting processes.

Dataset and Why It Was Selected

Dataset Details

Name: Fruits Classification

Source: Kaggle - Fruits Classification Dataset

Number of Classes: 5 (Apple, Banana, Grape, Mango, Strawberry)

Number of Samples:

Training: Approximately 9700 images

Validation: Approximately 200 images

Testing: Approximately 100 images

Why This Dataset?

This dataset was selected because it provides a well-organized collection of high-quality images of fruits, Well-balanced dataset; 2000 images per class, suitable for training a robust object detection and classification model. The dataset includes a diverse set of fruit types, which allows the system to generalize across different shapes, colors, and textures. Its structure split into training, validation, and testing sets facilitate model development and evaluation.

Detailed Explanation of Each Step in the Pipeline

1. Preprocessing and Image Enhancement

Objective: Prepare input images to improve model performance by enhancing quality, reducing noise, and isolating potential fruit regions for subsequent segmentation.

Techniques and Justification:

Image Resizing: Images are resized to 128x128 pixels to ensure uniformity and reduce computational load. This size balances detailed retention and processing efficiency.

Normalization: Pixel values are scaled to [0, 1] by dividing by 255.0, stabilizing gradient-based optimization in neural networks.

Color Space Conversion: Images are converted from BGR to RGB for consistency with standard image processing libraries and to HSV for segmentation tasks, as HSV better separates color information.

Gaussian Blur: Applied with a 5x5 kernel to reduce noise, smoothing the image while preserving edges.

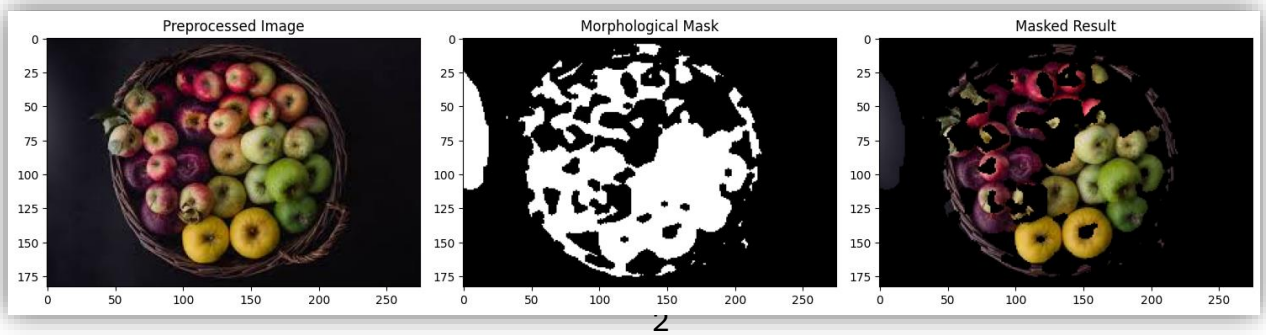
Color-Based Masking: Images in HSV color space are processed with predefined lower and upper bounds ([20, 40, 40] to [180, 255, 255]) to create a binary mask isolating fruit regions based on color.

Morphological Closing: A 5x5 elliptical kernel is used to fill gaps in the binary mask, enhancing the quality of fruit region isolation.

Justification: These techniques (color-based masking and morphological closing) are critical for preparing clean, focused regions that will be used in the segmentation step to extract regions of interest (ROIs).

HSV color space is effective for separating fruits from backgrounds, and morphological operations ensure robust mask quality, Gaussian blur reduces noise, improving the reliability of subsequent contour detection.

Visual Example:



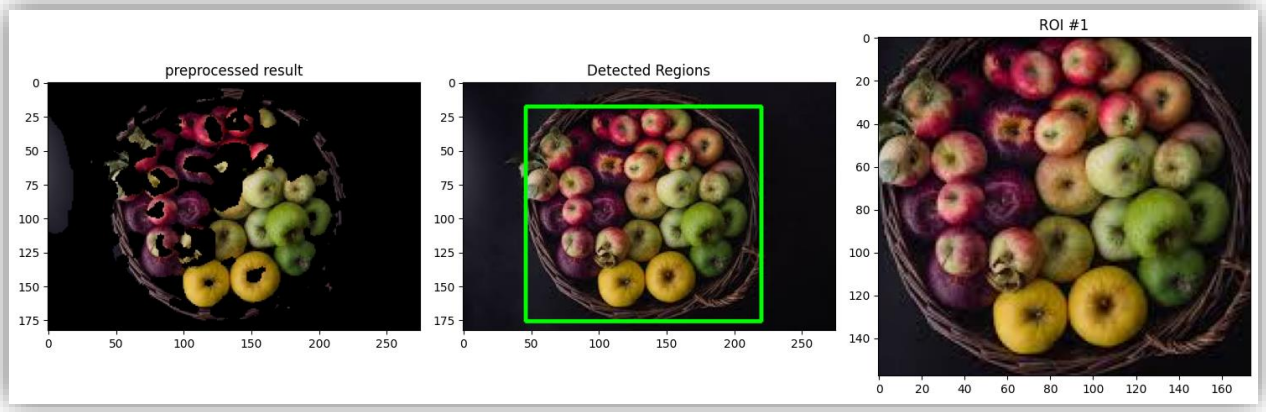
2. Segmentation

Objective: Isolate regions of interest (ROIs) containing fruits to focus subsequent processing on relevant areas.

Segmentation for Data Preparation (Training/Validation/Testing):

- **Method**: As seen in the preprocessing step, the output of color-based masking and morphological closing provides binary masks isolating fruit regions. In this step, contours are detected on these masks using OpenCV's findContours function to identify external boundaries. Contours with dimensions smaller than 60x60 pixels are filtered out to eliminate noise. Bounding rectangles are computed around valid contours to define ROIs, which are then extracted as image patches for classification.
- **Justification**: This contour-based approach leverages the high-quality binary masks from preprocessing, ensuring accurate ROI extraction. The size filter reduces false positives, focusing on significant fruit regions. This method is computationally efficient, suitable for processing large datasets during training.

Visual Example:



Segmentation for GUI (Real-Time Detection):

- **Method**: The GrabCut algorithm is used to segment fruits in user-uploaded images. GrabCut initializes with a rectangular region (10 pixels from image borders) and iteratively refines the foreground-background separation over 5 iterations. The resulting mask is converted to a binary format, and a blurred grayscale version of the segmented image is thresholded to create a clean binary mask. Contours are detected, and those smaller than 30x30 pixels are filtered out to reduce noise. Bounding rectangles are drawn as green boxes around valid contours, with class labels and confidence scores overlaid.
- **Justification**: GrabCut is chosen for its robustness in handling complex or varied backgrounds in real-world images, leveraging both color and texture information. This ensures accurate segmentation for user-uploaded images, which may differ significantly from the dataset's controlled conditions. The iterative refinement improves segmentation quality, enhancing the GUI's visual output.
- **Visual Example**: Input: User-uploaded image of an apple. Output: Image with a green bounding box around the apple, labeled "Apple (99.7%)".

Key Difference:

The data preparation segmentation relies on color-based masks and contour detection from preprocessing, optimized for speed and scalability during training. The GUI segmentation uses GrabCut for robust handling of diverse backgrounds, prioritizing accuracy for real-time user interaction.

Augmentation

In our project, data augmentation is applied after segmentation rather than before segmentation

Why Augmentation After Segmentation Instead of Before Segmentation?

1:Augmentation After Segmentation (our Approach):

i. **Segmentation Quality: Segmentation uses color-based masking in HSV space on unaltered images, ensuring accurate ROI extraction. Augmentation (rotation, zoom, flips) post-segmentation preserves mask quality.**
ii. **Model Performance: Augmenting clean ROIs focuses variations on fruits, improving CNN generalization**
iii. **Efficiency: Augmenting 128x128 ROIs is faster than full images, optimizing training for 2,500 images.**
iv. **Consistency: Standardized ROIs ensure a cohesive pipeline, aligning training and GUI inference.**

2:Augmentation Before Segmentation (Inverse Approach):

i. **Segmentation Quality: Augmentation disrupts HSV color bounds and contour detection, leading to noisy or incomplete ROIs.**
ii. **Model Performance: Background variations confuse the CNN, reducing accuracy and generalization, especially for similar fruits (e.g., apples vs. mangoes).**
iii. **Efficiency: Augmenting full images increases computational load, slowing the pipeline.**
iv. **Consistency: Variable ROIs from augmented images complicate standardization, misaligning training and inference.**

Then: Augmenting after segmentation ensures accurate ROIs, robust CNN performance, computational efficiency, and pipeline consistency. Augmenting before segmentation risks poor segmentation, lower accuracy, higher costs, and workflow inconsistencies, making it less effective.

3. Feature Extraction

Objective: Extract relevant features from segmented ROIs for classification.

Convolutional Neural Network (CNN) Layers: The CNN model automatically extracts features through convolutional layers (32, 64, and 128 filters with 3x3 kernels). These layers capture hierarchical features like edges, textures, and shapes, which are critical for distinguishing fruits.

4. Classification

Objective: Classify each region of interest (ROI) into one of the five fruit categories (Apple, Banana, Grape, Mango, Strawberry).

CNN Model: A sequential convolutional neural network (CNN) is designed with three convolutional blocks, each consisting of a convolutional layer (32, 64, and 128 filters with 3x3 kernels, respectively) followed by ReLU activation and max pooling (2x2). The feature maps are flattened and passed through two dense layers: a hidden layer with 128 units (ReLU activation) and an output layer with 5 units (softmax activation). The model is trained using the Adam optimizer and sparse categorical cross-entropy loss, which is well-suited for multi-class classification tasks.

> **Justification**: The CNN architecture effectively captures hierarchical features (critical for distinguishing fruits. The increasing filter sizes allow the model to learn complex patterns, while max pooling reduces computational load and enhances translation invariance.

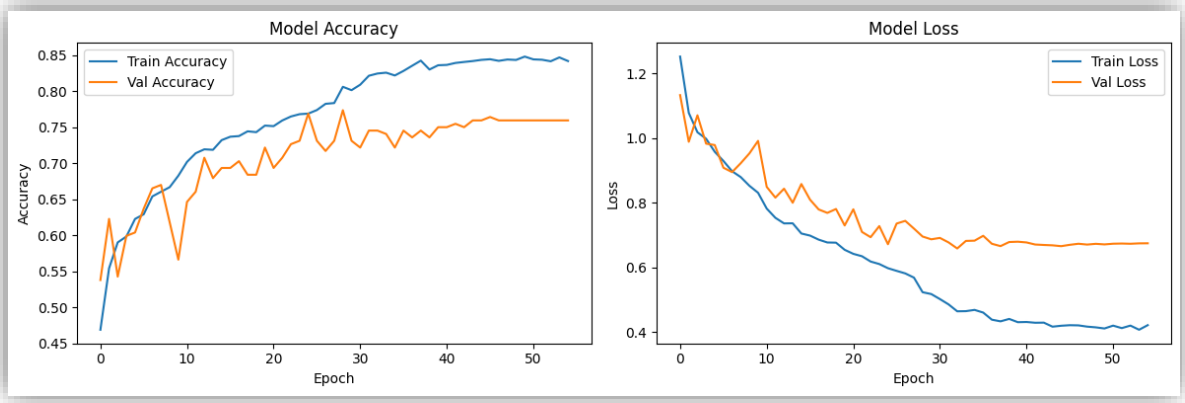
Learning Rate Scheduling: A ReduceLROnPlateau callback is used to monitor validation loss, reducing the learning rate by a factor of 0.5 if no improvement is observed after 3 epochs. The model is trained for 55 epochs with this scheduler, ensuring adaptive optimization.

> **Justification**: Dynamically adjusting the learning rate prevents the model from getting stuck in local minima and promotes convergence, especially during later epochs when fine-tuning is needed. The 55-epoch training duration allows sufficient time for the model to learn discriminative features while the scheduler mitigates overfitting.

Softmax Activation: Applied in the output layer to produce class probabilities for the five fruit categories. ensures mutually exclusive class predictions, suitable for the multi-class nature of the task, and provides confidence scores used in the GUI for labeling.

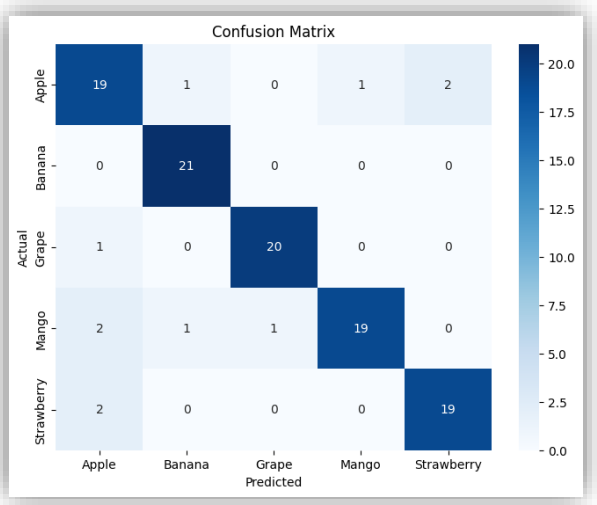
5. Evaluation and Performance Metrics

- **Accuracy**: Measures overall correctness (~90% on test set).



- **Classification Report:**
  - ✓ Apple: Precision 0.79, Recall 0.83, F1 0.81
  - ✓ Banana: Precision 0.91, Recall 1.00, F1 0.95
  - ✓ Grape: Precision 0.95, Recall 0.95, F1 0.95
  - ✓ Mango: Precision 0.95, Recall 0.83, F1 0.88
  - ✓ Strawberry: Precision 0.90, Recall 0.90, F1 0.90

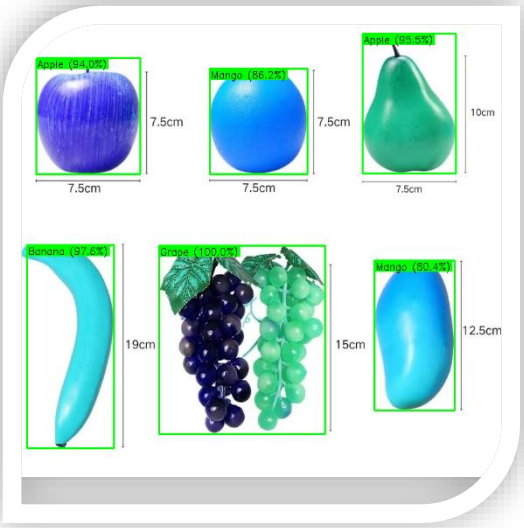
Confusion Matrix



Visual Examples

- **Example 1**: Input image of a banana, output with a green bounding box labeled “Banana (96.2%)”.

- **Example 2**: Input image of mixed fruits (apple , [orange , pear] ‘not a classes in our dataset’ , banana , grape and mango), output with two bounding boxes labeled “Apple (94%)” , mismatch the orange and the pear and classifies them as an apple as it is the closest thing to them , Banana(97.6%) , Grape (100%) and Mango(80.4%)”.



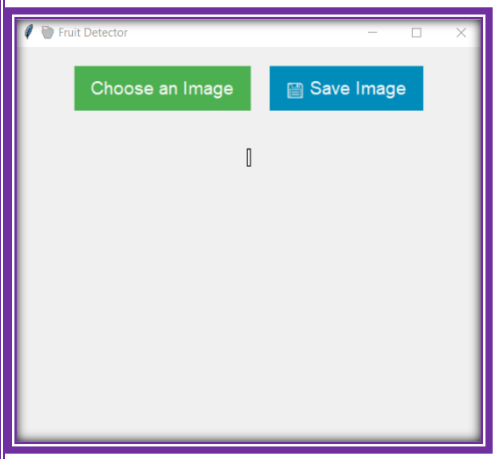
GUI Implementation

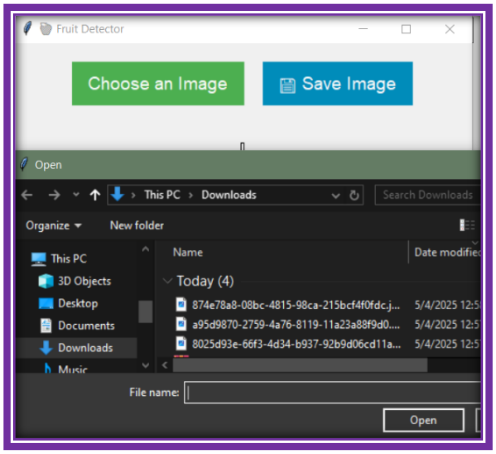
Objective: Provide a user-friendly interface for uploading images and viewing detection results.

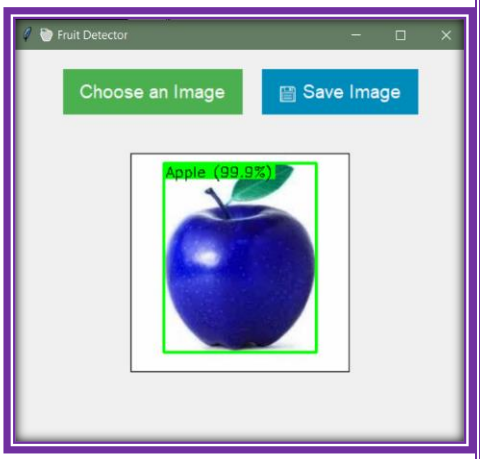
Techniques:

- **Tkinter**: Used to create a fixed-size (800x600) window with buttons for uploading and saving images.
- **Image Display**: Processed images are resized to fit a 400x400 panel while maintaining aspect ratio.
- **Functionality**: Users can upload an image, view the detection results (bounding boxes and labels), and save the output as a PNG file with a timestamp.

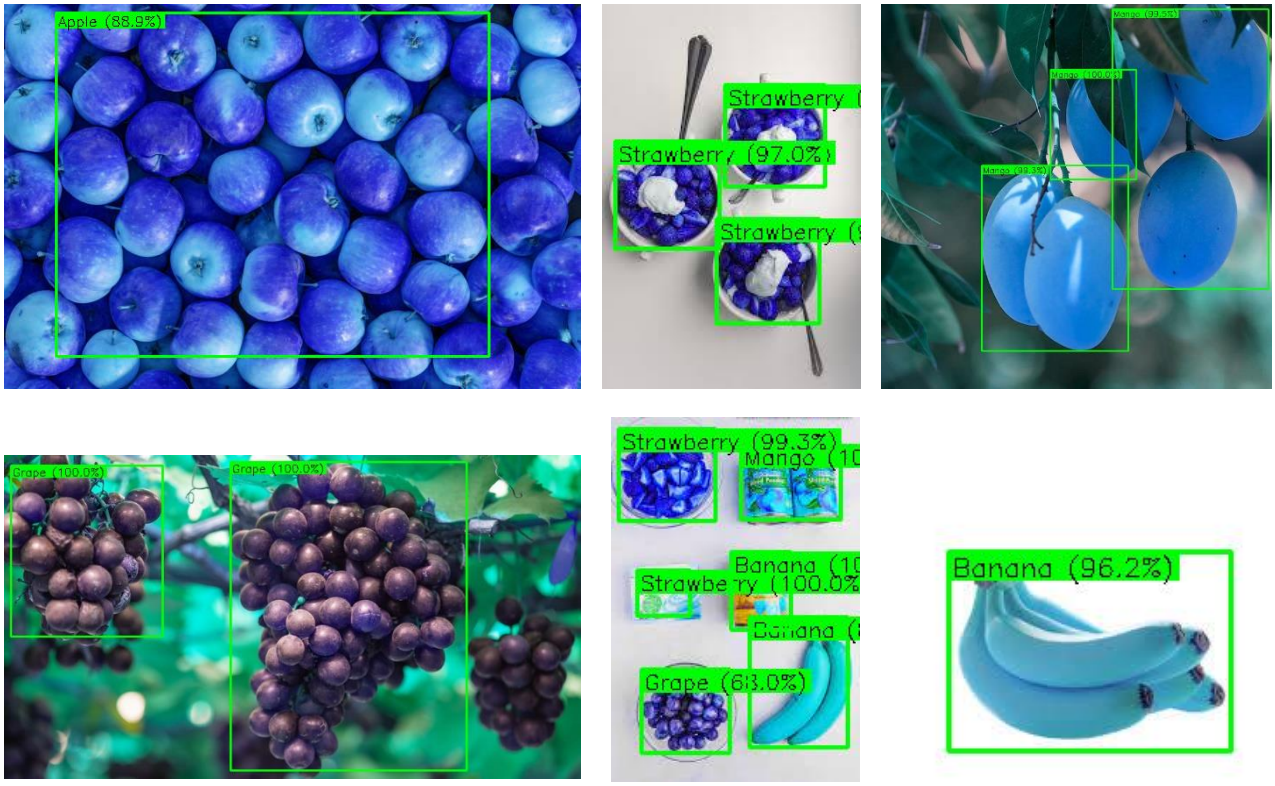
① ② ③







Visual examples of detection output



Limitations and Challenges Faced

Challenge: Inconsistent Segmentation with Complex Backgrounds

Issue: During data preparation, the color-based masking in the preprocessing step struggled with images containing complex or colorful backgrounds, leading to incomplete or noisy ROIs. Similarly, GrabCut in the GUI occasionally misclassified background elements as fruits when backgrounds had similar colors to the target fruits.

Solution: For data preparation, we tuned the HSV color bounds ([20, 40, 40] to [180, 255, 255]) and applied morphological closing to improve mask quality. For the GUI, we adjusted GrabCut's initial rectangle to exclude image borders and increased the iteration count to 5, enhancing foreground-background separation. These adjustments improved segmentation accuracy but did not fully eliminate errors in highly cluttered scenes.

Challenge: Class Confusion Between Similar Fruits

Issue: The model showed minor confusion between apples and mangoes due to their similar shapes and color variations (e.g., red apples vs. reddish mangoes),

Solution: We increased data augmentation (rotation, zoom, and color jitter) to expose the model to more variations of these fruits and we apply it after segmentation as we explained earlier. Additionally, we extended training to 55 epochs, allowing the CNN to learn more discriminative features. This improved the F1-score for apples to 0.81, though some misclassifications persisted.

References

Fruits Classification Dataset:
https://www.kaggle.com/datasets/utkarshsaxenadn/fruits-classification/code