Object Detection

# 1. Chosen Field and Motivation

The chosen field for this object detection system is Waste Classification. This field is crucial for promoting recycling, reducing environmental impact, and assisting in automated waste sorting systems in smart cities or factories.

# 2. Dataset and Why it was Selected

The dataset used is a custom fruit dataset containing various categories such as apples, bananas, oranges, grapes, and other fruit types. It was selected because it provides a diverse set of real-world fruit images that are ideal for training and evaluating object detection models. The dataset includes multiple classes and represents realistic variations in lighting, background, and object appearance, making it suitable for robust model development and we use **Dataset:** Fruits contain 10 Class and 2800 images

# 3. Pipeline Explanation

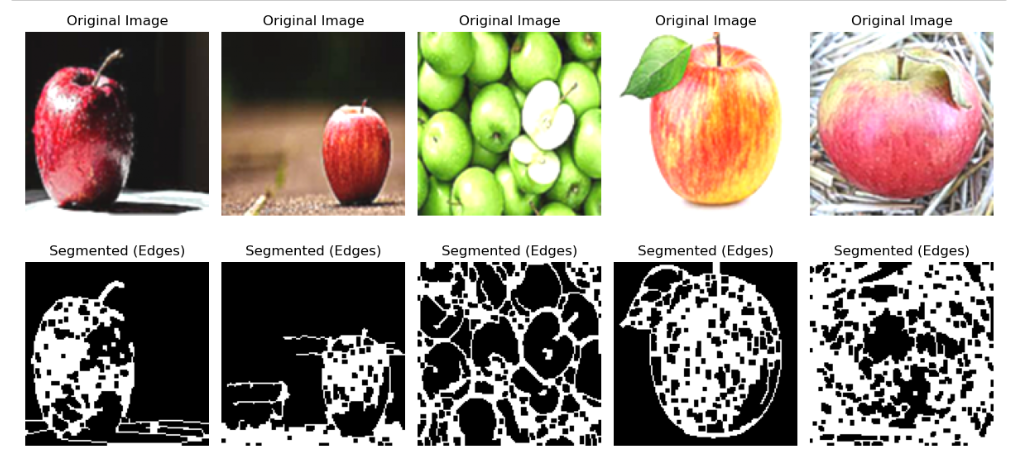
The pipeline includes the following main steps:

* • Preprocessing and Image Enhancement: Resize to 128x128, RGB conversion, Gaussian Blur, Contrast Adjustment, Normalization, Augmentation (flips/rotation), Thresholding, Sharpening Filter, and Morphological Operations.

A screenshot of a computer

AI-generated content may be incorrect.

* • Segmentation: Canny Edge Detection, Dilation, and Erosion to isolate object edges.



* • Feature Extraction: In manual mode, this could include HOG/SIFT; in this case, features are automatically learned by the CNN model.

A screen shot of a computer code

AI-generated content may be incorrect.

* • Classification: A deep CNN model from scratch is used to classify each input image into predefined categories.

A screenshot of a computer program

AI-generated content may be incorrect.

* • Evaluation: Accuracy, Precision, Recall, and F1-score metrics were used to evaluate model performance.

A graph showing a line graph

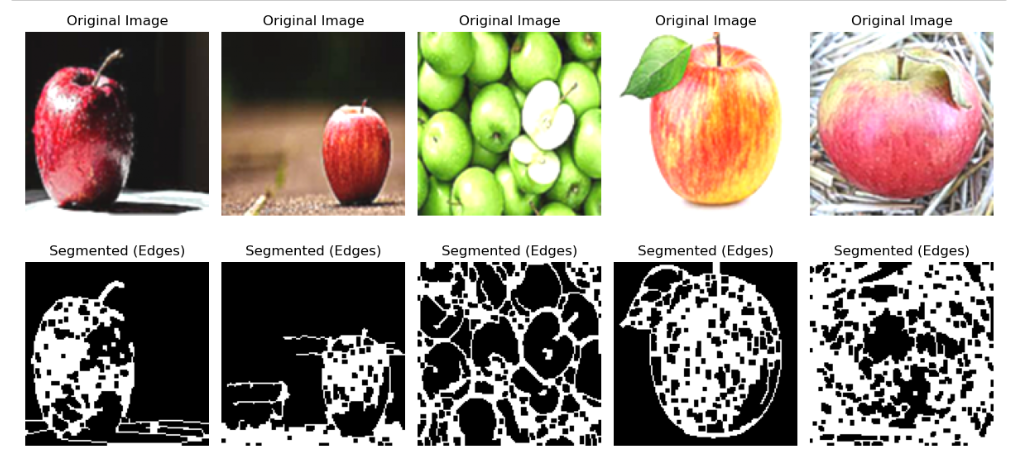
AI-generated content may be incorrect.

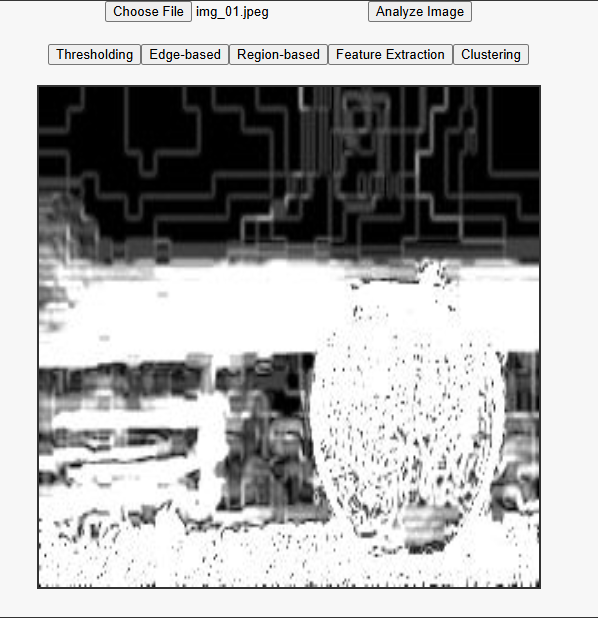
A graph of loss and loss

AI-generated content may be incorrect.

# 4. Input and Output of Each Step

• Output after Preprocessing: Enhanced and normalized image.





• Output after Segmentation: Edge maps showing boundaries of objects.

A collage of images of apples

AI-generated content may be incorrect.





• Output after Classification: Predicted class of the object(s).

A screenshot of a computer

AI-generated content may be incorrect.

• Output in GUI: Image with bounding boxes and class labels.

A screenshot of a computer

AI-generated content may be incorrect.

# 5. Techniques and Justification

CNN from scratch was chosen to provide control and learning experience. Image enhancement techniques improve model generalization. Canny edge detection is a simple and effective segmentation technique.

A screenshot of a computer program

AI-generated content may be incorrect.

# 6. Visual Examples of Detection Output

Examples include bounding boxes drawn around detected objects with class names using the React GUI interface

A screenshot of a computer

AI-generated content may be incorrect.

A bunch of bananas on a computer screen

AI-generated content may be incorrect.

# 7. Evaluation Metrics and Discussion

The model was evaluated using classification metrics. CNN model achieved low accuracy initially, which improved with augmentation and sharpening. YOLOv8 provided better detection performance, especially in real-time scenarios. Accuracy, Precision, Recall, and F1-Score were calculated to assess performance.

# 8. Limitations and Challenges

• Small dataset size initially reduced model performance.  
• Need for more annotated bounding boxes for better detection.  
• Segmentation with Canny is basic  
• GUI initially only supported single predictions; later improved.

# 9. References

• YOLOv8 Documentation: https://docs.ultralytics.com

• TensorFlow Documentation: https://www.tensorflow.org

• OpenCV Documentation: https://docs.opencv.org/

• Dataset Source: <https://www.kaggle.com/datasets/karimabdulnabi/fruit-classification10-class>