Project Name:

**Fruit Detection Using Object Detection Techniques & CNN**

|  |  |  |
| --- | --- | --- |
|  | Team Member ID | Team member name |
| 1 | 20210265 | حازم احمد عبدالعال شاذلي |
| 2 | 20210565 | عبد الواحد رجب عبد الواحد |
| 3 | 20210531 | عبد الرحمن مصطفي حامد |
| 4 | 20210272 | حافظ عادل حافظ علي عميش |
| 5 | 20210807 | محمد عبد الرزاق فؤاد جعفر |
| 6 | 20210841 | محمد ناصر رفعت علي |
| 7 | 20210666 | فاطمة محمد أبو الحجاج سعيد |

# **Project Objective**

This project aims to develop a complete object detection system that can accurately detect and classify fruits in static images using computer vision techniques. The system includes a Graphical User Interface (GUI) and supports the following functionalities:

* Accepting a static image as input.
* Detecting one or more fruits in the image.
* Drawing bounding boxes around each detected fruit.
* Labeling each fruit with its predicted class name.

# **Chosen Field & Motivation**

The field selected for this project is **Agricultural Product Identification**, focusing on fruit detection.  
This project is motivated by the need for smart systems in agriculture, food quality control, and inventory automation. Object detection in fruits can aid in:

* **Improving accuracy and speed in identifying fruit types.**
* **Enhancing smart applications in grocery stores or farms.**

# **Dataset Selection**

* **Name:** [Fruit Images for Object Detection](https://www.kaggle.com/datasets/mbkinaci/fruit-images-for-object-detection)
* **Source:** Kaggle Dataset - Fruit Images for Object Detection
* **Classes:**
  + Apple
  + Banana
  + Orange
  + Background Label

Having a label for the background is essential for the following reasons:

* **Distinction Between Classes**:
  + In object detection tasks, it is crucial to distinguish between the objects of interest (e.g., fruits) and the background. The background label helps the model understand what does not belong to any of the target classes.
* **Training Data Balance**:
  + Including background samples ensures that the model is trained on a balanced dataset. This helps prevent the model from becoming biased toward detecting only the objects of interest while neglecting to recognize non-object areas.
* **Reduction of False Positives**:
  + By explicitly labeling background areas, the model learns to avoid false positives—incorrectly identifying background regions as objects. This is particularly important in scenarios where the background may contain similar colors or shapes to the objects being detected.
* **Improved Generalization**:
  + Training with both positive samples (objects) and negative samples (background) allows the model to generalize better. It learns to recognize objects in various contexts and backgrounds, making it more effective in real-world applications.
* **Multi-Class Classification**:
  + In multi-class classification problems, having a background class allows the model to output probabilities for multiple classes, including the absence of an object. This is useful for applications where you need to know not just what is present, but also when nothing of interest is detected.
* **Number of Images:** Includes several labeled images for each class, and some images contain mixed fruit types. (260)

**Why this dataset?**

* Contains annotated images ideal for training and testing object detection models. (xml files)
* Lightweight and well-organized.
* Focused on fruits, matching the domain of the project.

# **Exploratory Data Analysis (EDA)**

**1 - We want to check**

* **Reveals whether the dataset is balanced or imbalanced.**
* **Helps in deciding whether to apply class weighting, oversampling, or data augmentation.**
* **Ensure the model doesn't become biased toward more frequent classes.**

**A graph of a class distribution

AI-generated content may be incorrect.**

* **2 – We want insight into the scale of objects per class to Helps anticipate whether small or large objects may cause detection difficulties.**

A graph of blue and orange bars

AI-generated content may be incorrect.

# **Preprocessing and Image Enhancement**

**In this project, several preprocessing techniques were applied to the input images to improve the performance of the object detection system. The following methods were utilized:**

1. **Gaussian Blur**

* A computer screen shot of text

  AI-generated content may be incorrect.**is applied to reduce noise and smooth the image. It involves convolving the image with a Gaussian kernel.**
* **Benefits:**
  + **Noise Reduction: Helps in minimizing high-frequency noise, making it easier for the detection algorithms to identify relevant features.**
  + **Edge Preservation: While smoothing the image, it preserves edges better than other blurring techniques, which is crucial for object detection.**

1. **Contrast Limited Adaptive Histogram Equalization (CLAHE)**

* **A computer screen shot of a program

  AI-generated content may be incorrect.CLAHE enhances the contrast of the image by applying histogram equalization to small tiles of the image, thereby improving local contrast.**
* **Benefits:**
  + **Local Contrast Enhancement: Unlike standard histogram equalization, which can lead to over-amplification of noise in homogeneous areas, CLAHE enhances contrast locally, making features more distinguishable.**
  + **Prevention of Over-Contrast: The clipLimit parameter ensures that the contrast does not exceed a certain threshold, preventing the introduction of artifacts.**

1. **Color Space Conversion (BGR to LAB)**

* The image is converted from BGR (Blue, Green, Red) to LAB color space before applying CLAHE.

**A screen shot of a computer

AI-generated content may be incorrect.**

* **Benefits:**
  + **Luminance and Color Separation**: LAB separates luminance (L) from color information (A and B), allowing for more effective contrast enhancement on the luminance channel without affecting color fidelity.
  + **Improved Feature Detection**: Enhancing the luminance channel helps in better feature detection, as it focuses on brightness variations rather than color variations.

1. **Image Resizing**

**The images are resized to a standard dimension (e.g., 128x128 pixels) to ensure uniformity across the dataset.**

1. **Normalization**
2. **Improved Convergence**: Normalizing pixel values helps the model converge faster during training by ensuring that the gradients are not too large or too small.

# **Segmentation**

**This section encompasses various methods for generating Region of Interest (ROI) proposals. Each method is designed to detect and isolate objects within images.**

**Thresholding Method**

**This function generates Region of Interest (ROI) proposals using Otsu's thresholding method, which is effective for segmenting images based on pixel intensity.**

1. **Extract Image Dimensions**
   * **Retrieves the height (img\_h) and width (img\_w) of the input image, which will be used for filtering proposals later.**
2. **Convert to Grayscale:** 
   * **Converts the input image from BGR (Blue, Green, Red) color space to grayscale. This simplifies the image data and is often sufficient for segmentation tasks.**
3. **Apply Gaussian Blur:**
   * **Applies Gaussian blur to the grayscale image to reduce noise and smooth out the image. The kernel size of (7, 7) indicates the size of the blurring filter, and a value of 0 means that the standard deviation is calculated from the kernel size.**
4. **Otsu's Thresholding:**
   * Otsu's method automatically determines the optimal threshold value to separate the foreground from the background.
   * The parameters used here are:
     + 0: This is a placeholder since Otsu's method calculates the threshold automatically.
     + 255: This is the maximum value to use with the binary thresholding.
     + cv2.THRESH\_BINARY\_INV + cv2.THRESH\_OTSU: This combination specifies that the thresholding should be binary inverted (i.e., pixels above the threshold become black and those below become white) and that Otsu's method should be used to calculate the threshold value.
5. **Morphological Operations:**
   * **Morphological Closing**: The first operation (MORPH\_CLOSE) is applied to close small holes in the foreground. This is done using a kernel of ones with size (5, 5) and is applied for 2 iterations.
   * **Morphological Opening**: The second operation (MORPH\_OPEN) is applied to remove small noise points from the background. This uses the same kernel and is applied for 1 iteration.
6. **Find Contours:**
   * This function retrieves the contours of the cleaned threshold image. cv2.RETR\_EXTERNAL retrieves only the extreme outer contours, while cv2.CHAIN\_APPROX\_SIMPLE compresses horizontal, vertical, and diagonal segments and leaves only their endpoints.
7. **Bounding Rectangles:**
   * For each contour found, the bounding rectangle is calculated, and its coordinates (x, y) along with its width w and height h are stored in a list called proposals\_raw.
8. **Filtering Proposals:**
   * The raw proposals are then passed to the filter\_proposals function, which filters them based on specified criteria, such as minimum area, aspect ratio, and maximum area ratio. The filtered proposals are returned.

A screen shot of a computer code

AI-generated content may be incorrect.

Note

**Threshold Value Used**

In the context of Otsu's thresholding:

* **Automatic Threshold Calculation**: The specific threshold value used is not explicitly defined in the code because Otsu's method computes it dynamically based on the image histogram. This means that the threshold value is determined based on the distribution of pixel intensities in the blurred grayscale image.

Image With and without Preprocessing

A screenshot of a computer

AI-generated content may be incorrect.

**2- K-Means Clustering**

**function generates Region of Interest (ROI) proposals using K-Means clustering based on the color information of the pixels in the image.**

1. **Extract Image Dimensions**
   * **Retrieves the height (img\_h) and width (img\_w) of the input image, which will be used for filtering proposals later.**

**A black background with white text

AI-generated content may be incorrect.**

1. **Reshape Image for Clustering:**
   * **The input image is reshaped from a 2D array (height x width) to a 1D array of pixel values, where each pixel is represented by its RGB color values.**
   * **The pixel values are converted to float32 type to prepare them for the K-Means algorithm**

**A screen shot of a computer

AI-generated content may be incorrect.**

1. **Define K-Means Criteria:**
   * **The termination criteria for the K-Means algorithm are defined:**
     + **cv2.TERM\_CRITERIA\_EPS + cv2.TERM\_CRITERIA\_MAX\_ITER: The algorithm will stop either after a set number of iterations or when the specified accuracy is reached.**
     + **100: Maximum number of iterations.**
     + **0.2: Desired accuracy (the algorithm stops when the change in cluster centers is less than this value).**

****

1. **Apply K-Means Clustering:**
   * The K-Means algorithm is applied to the pixel values:
     + n\_clusters: The number of clusters to form (in this case, 4).
     + None: Initial labels (none are provided, so the algorithm initializes them).
     + criteria: The previously defined termination criteria.
     + 10: Number of attempts to find the best clustering.
     + cv2.KMEANS\_RANDOM\_CENTERS: The initial centers are chosen randomly.



1. **Reconstruct Segmented Image**:
   * The cluster centers are converted to unsigned 8-bit integers.
   * A segmented image is created by mapping each pixel to its corresponding cluster center based on the labels assigned by K-Means.
   * The segmented image is reshaped back to the original image dimensions.

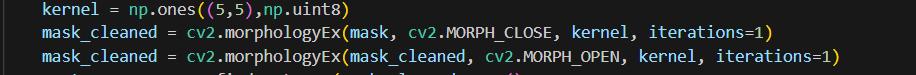
A black screen with white text

AI-generated content may be incorrect.

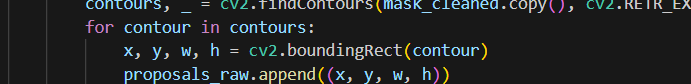
1. **Create Masks for Each Cluster**
   * For each cluster, a binary mask is created using cv2.inRange, which identifies pixels that belong to the current cluster center.



1. **Morphological Operations**:
   * Morphological operations are applied to clean the mask:
     + **Morphological Closing**: Closes small holes in the detected regions.
     + **Morphological Opening**: Removes small noise points from the background.



1. **Find Contours**:



1. **Bounding Rectangles**:
2. **Filtering Proposals**:
3. **Unique Proposals**:
   * Duplicate proposals are removed by converting them to a set and then back to a list format, ensuring that only unique proposals are returned.

A screen shot of a computer code

AI-generated content may be incorrect.

3 - **Enhanced Contour**

**This method focuses on contour detection with additional enhancements to improve the accuracy and quality of the detected contours.**

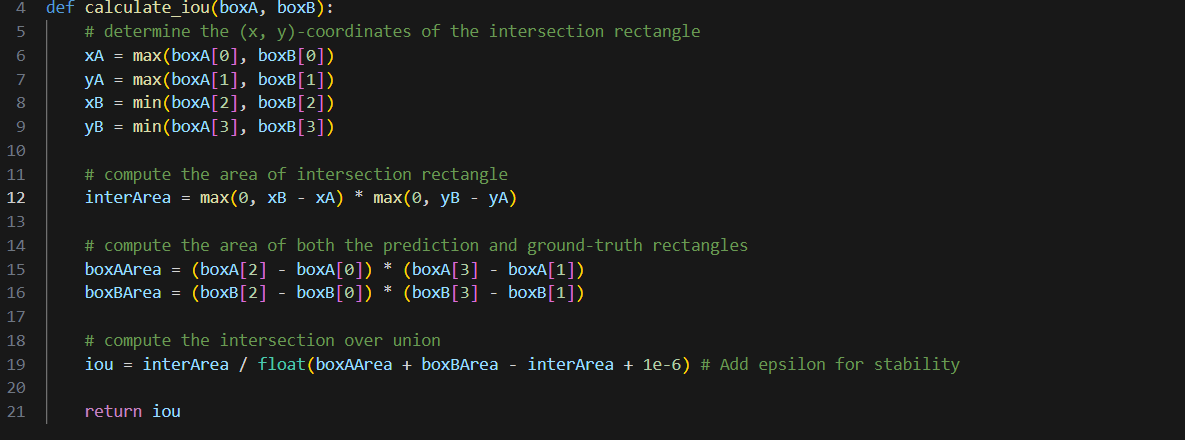
1. **Extract Image Dimensions**
2. **Convert to Grayscale:**
3. **Apply Gaussian Blur**
4. **Edge Detection**
5. **Morphological Closing**
6. **Find Contours**
7. **Bounding Rectangles:**
8. **Filtering Proposals**
9. **Return Final Proposals:**

**Enhanced Contour Method Characteristics**

* **Contour-Based Segmentation: This method relies on detecting the outlines of objects within the image, making it effective for images with well-defined shapes.**
* **Morphological Operations: The use of morphological operations helps to refine the contours, making the detection process more robust against noise and small imperfections in the image.**
* **A screen shot of a computer code

  AI-generated content may be incorrect.Parameter Tuning: The method allows for tuning various parameters such as blur kernel size, Canny thresholds, and morphological iterations, which can be adjusted based on the specific characteristics of the input images.**

# **Feature Extraction**



* **Purpose: This function calculates the Intersection over Union (IoU) between two bounding boxes. IoU is a common metric used to evaluate the overlap between predicted and ground truth bounding boxes.**
* **Parameters:**
  + **boxA: Coordinates of the first bounding box in the format [xmin, ymin, xmax, ymax].**
  + **boxB: Coordinates of the second bounding box in the same format.**
* **Output: Returns the IoU value, which ranges from 0 (no overlap) to 1 (perfect overlap).**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

* **Purpose: These constants help in determining which samples are positive or negative based on their IoU values, as well as controlling the number of samples generated during training.**

# **Machine Learning Model**

A screenshot of a computer program

AI-generated content may be incorrect.

* **simple CNN architecture consisting of convolutional layers, max pooling layers, dropout for regularization, and dense layers for classification.**
* **Architecture:**
  + **Convolutional Layers: Extract features from the input images.**
  + **Max Pooling Layers: Reduce the spatial dimensions of the feature maps.**
  + **Dropout Layers: Prevent overfitting by randomly dropping neurons during training.**
  + **Dense Layers: Final classification layer using softmax activation for multi-class outputs.**

# **Evaluation and Performance Metrics**

**A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

AI-generated content may be incorrect.**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

# **Save Model & GUI**

A screen shot of a computer

AI-generated content may be incorrect.

We have saved joblib encoder and model.keras

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Limitation Faced :**

**2. IoU Threshold Sensitivity**

* **Threshold Selection**: The choice of IoU thresholds for classifying positive and negative samples can significantly impact model performance. Too high a threshold may reject valid detections, while too low a threshold may accept too many false positives.
* **Variability in Object Size and Shape**: Different objects may have varying shapes and sizes, making a fixed IoU threshold less effective across all classes.

# 

# References

<https://builtin.com/machine-learning/image-segmentation>

<https://medium.com/@maahip1304/the-complete-guide-to-image-preprocessing-techniques-in-python-dca30804550c>

https://www.kaggle.com/datasets/mbkinaci/fruit-images-for-object-detection

# Technology used

* + Tensorflow and deeplearning

# 