

## Data Collection and Preprocessing Phase

Date	19 June 2025
Team ID	SWTID1750170729
Project Title	<b>Deepfruitveg: Automated Fruit And Veg Identification</b>
Maximum Marks	6 Marks

### Preprocessing Template

The images will be preprocessed by resizing, normalizing, augmenting, denoising, adjusting contrast, detecting edges, converting color space, cropping, batch normalizing, and whitening data. These steps will enhance data quality, promote model generalization, and improve convergence during neural network training, ensuring robust and efficient performance across various computer vision tasks.

Section	Description
Data Overview	This dataset contains images of fruits and vegetables across 30 categories. Each class has more than 150 images for training, and 100 images each for validation and testing. These images are resized and preprocessed before training a EfficientNetB3-based CNN model.
Resizing	Resize images to a fixed target size of (260, 200) or (224, 224) pixels depending on the model's input requirement.
Normalization	Normalize image pixel values using EfficientNetB3's preprocess_input() to scale them into the range expected by the pre-trained model.
Data Augmentation	Using Keras ImageDataGenerator to apply transformations like rotation, flipping, and zoom to increase data variability.
Denoising	Applying OpenCV's Non-Local Means denoising algorithm to reduce image noise before feeding into the model.

Edge Detection	Applying Canny edge detection to highlight object boundaries — useful for feature extraction or visualizations.
Color Space Conversion	Convert image from BGR (OpenCV default) to RGB for compatibility with TensorFlow/Keras.
Image Cropping	Crop images to focus on the regions containing objects of interest.
Batch Normalization	Apply batch normalization to the input of each layer in the neural network.
<b>Data Preprocessing Code Screenshots</b>	
Loading Data	<pre>[TRAIN] Saved: resized\train\aloevera\20250619-085057-069361.jpg [TRAIN] Saved: resized\train\aloevera\20250619-085057-351944.jpg [TRAIN] Saved: resized\train\aloevera\20250619-085057-650062.jpg [TRAIN] Saved: resized\train\aloevera\20250619-085058-407130.jpg [TRAIN] Saved: resized\train\aloevera\20250619-085059-081043.jpg [TRAIN] Saved: resized\train\aloevera\20250619-085059-365170.jpg [TRAIN] Saved: resized\train\aloevera\20250619-085059-657034.jpg [TRAIN] Saved: resized\train\aloevera\20250619-085059-941451.jpg [TRAIN] Saved: resized\train\aloevera\20250619-085100-254267.jpg [TRAIN] Saved: resized\train\aloevera\20250619-085100-546014.jpg [TRAIN] Saved: resized\train\aloevera\20250619-085101-449667.jpg [TRAIN] Saved: resized\train\aloevera\20250619-085102-177268.jpg [TRAIN] Saved: resized\train\aloevera\20250619-085102-454637.jpg [TRAIN] Saved: resized\train\aloevera\20250619-085102-738072.jpg [TRAIN] Saved: resized\train\aloevera\20250619-085103-031117.jpg [TRAIN] Saved: resized\train\aloevera\20250619-085103-321347.jpg [TRAIN] Saved: resized\train\aloevera\20250619-085103-579228.jpg [TRAIN] Saved: resized\train\aloevera\20250619-085103-863240.jpg [TRAIN] Saved: resized\train\aloevera\20250619-085104-127757.jpg [TRAIN] Saved: resized\train\aloevera\20250619-085104-410383.jpg [TRAIN] Saved: resized\train\aloevera\20250619-085105-098785.jpg [TRAIN] Saved: resized\train\aloevera\20250619-085105-798903.jpg [TRAIN] Saved: resized\train\aloevera\20250619-085106-071716.jpg [TRAIN] Saved: resized\train\aloevera\20250619-085106-352922.jpg [TRAIN] Saved: resized\train\aloevera\20250619-085107-542332.jpg ... [TEST] Saved: resized\test\watermelon\20250619-095102-235309.jpg [TEST] Saved: resized\test\watermelon\20250619-095102-831708.jpg [TEST] Saved: resized\test\watermelon\20250619-095103-101143.jpg [TEST] Saved: resized\test\watermelon\20250619-095103-364991.jpg</pre>

Resizing	<pre> 1  import os 2  import random 3  from datetime import datetime 4  import cv2 5  from PIL import Image 6  import numpy as np 7 8  # Folder where your images are stored (already split into train, validation) 9  input_root = 'data/merged' 10 11 # Folder where resized images will be saved 12 output_root = 'resized' 13 14 # Final size of each image: width=260, height=200 15 resize_size = (260, 200) 16 17 # Maximum number of images to keep per class in each set 18 max_images_per_set = { 19     'train': 120,      # Keep only 120 images per class 20     'validation': 15,  # Keep only 15 images per class 21     'test': 15        # Keep only 15 images per class 22 } 23 24 # Resize image to 260x200 25 img = cv2.resize(img, resize_size) 26 27 # Save the image using PIL 28 img_pil = Image.fromarray(img) 29 img_pil.save(output_path, optimize=True, quality=100) 30 31 print(f"[{dataset_type.upper()}] Saved: {output_path}") </pre>
Normalization	<p>Prepare Inputs</p> <pre> train_x = np.array([item[0] for item in train_data], dtype=np.float32) train_y = np.array([item[1] for item in train_data], dtype=np.float32)  val_x = np.array([x[0] for x in val_data], dtype=np.float32) val_y = np.array([x[1] for x in val_data], dtype=np.float32)  test_x = np.array([item[0] for item in test_data], dtype=np.float32) test_y = np.array([item[1] for item in test_data], dtype=np.float32) </pre> <p>Normalize using EfficientNetB3 PreProcess</p> <pre> training_x = preprocess_input(train_x) validation_x = preprocess_input(val_x) testing_x = preprocess_input(test_x) </pre>
Data Augmentation	<pre> from keras.preprocessing.image import ImageDataGenerator datagen = ImageDataGenerator(rotation_range=20, zoom_range=0.2, horizontal_flip=True) </pre>

Denoising	<pre>try:     # Read the image using OpenCV     img = cv2.imread(input_path)      # If image is not readable, skip it     if img is None:         raise Exception("Image could not be read")      # Convert image from BGR (OpenCV default) to RGB     img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)      # Apply noise reduction     img = cv2.fastNlMeansDenoisingColored(img, None, 10, 10, 7, 21)      # Resize image to 260x200     img = cv2.resize(img, resize_size)      # Save the image using PIL     img_pil = Image.fromarray(img)     img_pil.save(output_path, optimize=True, quality=100)      print(f"[{dataset_type.upper()}] Saved: {output_path}")</pre>
Edge Detection	<pre>edges = cv2.Canny(img, 100, 200)</pre>
Color Space Conversion	<pre># Convert image from BGR (OpenCV default) to RGB img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)</pre>
Image Cropping	Not Needed As resized already Done
Batch Normalization	Not used explicitly, but EfficientNetB3 uses it internally.