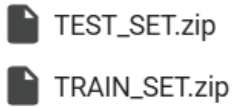
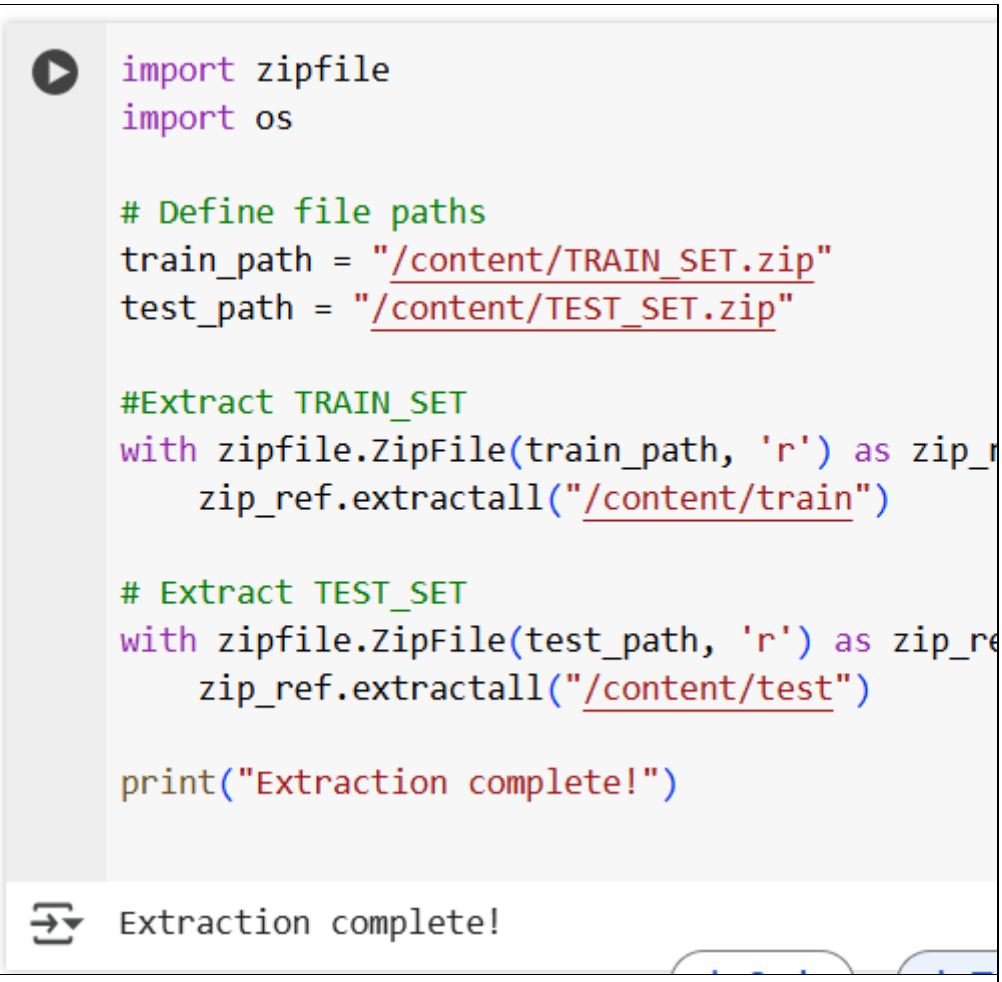
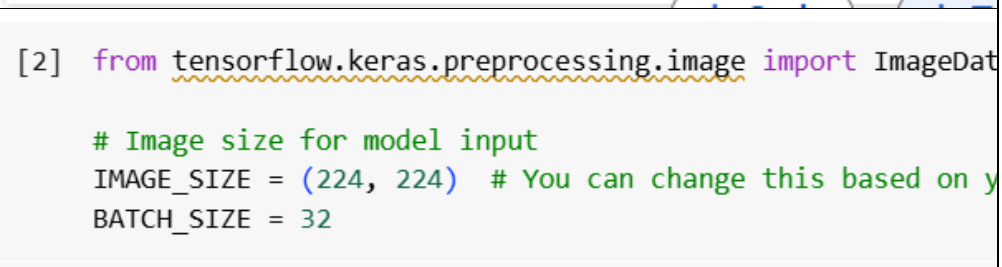
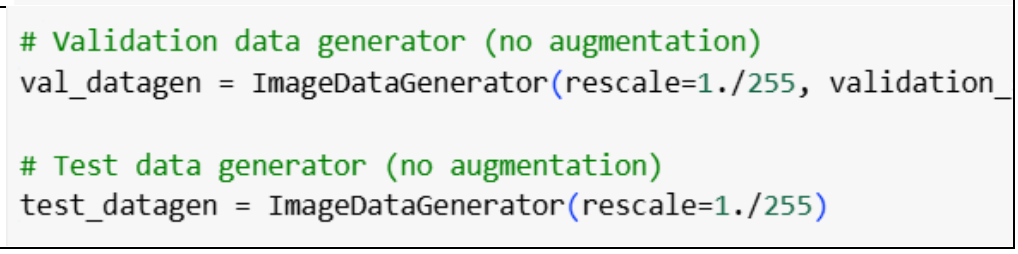
AI-Powered Nutrition Analyzer For Fitness Enthusiasts

Date	21-06-2025
Team ID	SWTID1749893823
Project Title	AI-Powered Nutrition Analyzer For Fitness Enthusiasts
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	 Small dataset given in the guided project (test set, train set) for 5 fruits around 150 images each.
Resizing	Resize images to a specified target size.
Normalization	Normalize pixel values to a specific range.
Data Augmentation	Apply augmentation techniques such as flipping, rotation, shifting, zooming, or shearing.
Denoising	Apply denoising filters to reduce noise in the images.
Edge Detection	Apply edge detection algorithms to highlight prominent edges in the images.
Color Space Conversion	Convert images from one color space to another
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.
Data Preprocessing Code Screenshots	

Loading Data	
Resizing	
Normalization	

Data Augmentation	```# Train data generator with augmentation train_datagen = ImageDataGenerator(rescale=1./255, rotation_range=20, zoom_range=0.2, horizontal_flip=True, validation_split=0.2 # 80% train, 20% validatio)```
Denoising	```import cv2 import matplotlib.pyplot as plt # Load an image img = cv2.imread('/content/train/TRAIN_SET/BANANA/104_100.jpg') img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) # Denoise the image denoised = cv2.fastNlMeansDenoisingColored(img, None, h=10, hColor=10, templateWindowSize=7, s # Plot original and denoised images plt.subplot(1, 2, 1) plt.imshow(img) plt.title("Original Image") plt.subplot(1, 2, 2) plt.imshow(denoised) plt.title("Denoised Image") plt.show()```
Edge Detection	```import cv2 import matplotlib.pyplot as plt # Load the image in grayscale img = cv2.imread('/content/train/TRAIN_SET/BANANA/104_100.jpg', cv2.IMR # Apply Canny edge detection edges = cv2.Canny(img, threshold1=100, threshold2=200) # Show original and edge-detected images plt.figure(figsize=(10,5)) plt.subplot(1, 2, 1) plt.imshow(img, cmap='gray') plt.title("Original Image") plt.axis('off') plt.subplot(1, 2, 2) plt.imshow(edges, cmap='gray') plt.title("Edge Detection (Canny)") plt.axis('off') plt.show()```

Color Space Conversion	``` import cv2 import matplotlib.pyplot as plt # Load an image img = cv2.imread('/content/train/TRAIN_SET/BANANA/104_100 img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) ```
Image Cropping	``` import cv2 import matplotlib.pyplot as plt # Load the image img = cv2.imread('/content/train/TRAIN_SET/BANANA/104_100.jpg img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) # Print shape to guide cropping print("Image shape:", img.shape) # Safe crop (adjust based on shape) cropped_img = img[30:120, 50:180] # Display plt.figure(figsize=(10,5)) plt.subplot(1, 2, 1) plt.imshow(img) plt.title("Original Image") plt.axis('off') plt.subplot(1, 2, 2) plt.imshow(cropped_img) plt.title("Cropped Image") plt.axis('off') plt.show() ```

Batch Normalization	[16] ``` from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D, Batch from tensorflow.keras.models import Sequential from tensorflow.keras.applications import MobileNetV2 # Number of classes NUM_CLASSES = 5 # Base model base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(224, base_model.trainable = False # Build the model model = Sequential([base_model, GlobalAveragePooling2D(), Dense(128), BatchNormalization(), tf.keras.layers.Activation('relu'), Dropout(0.3), Dense(NUM_CLASSES, activation='softmax')]) ```