Data Collection and Preprocessing Phase

Date	20 November 2025
Team ID	740018
Project Title	Deepfruitveg:Automated Fruit And Vegetables Identification
Maximum Marks	6 Marks

Preprocessing Template

Preprocessing in Deepfruitveg involves tasks like image resizing, normalization, and noise reduction to improve the quality of input images. Additionally, data augmentation techniques such as rotation, flipping, and cropping are applied to increase dataset diversity, ensuring the deep learning model can generalize well and handle varying environmental conditions.

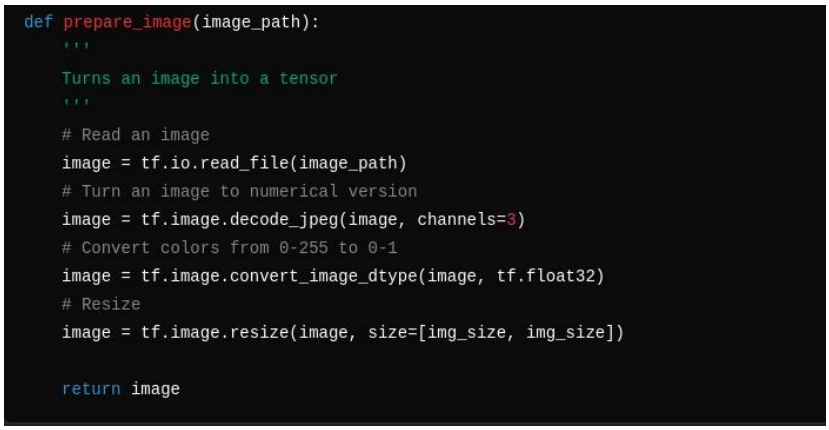
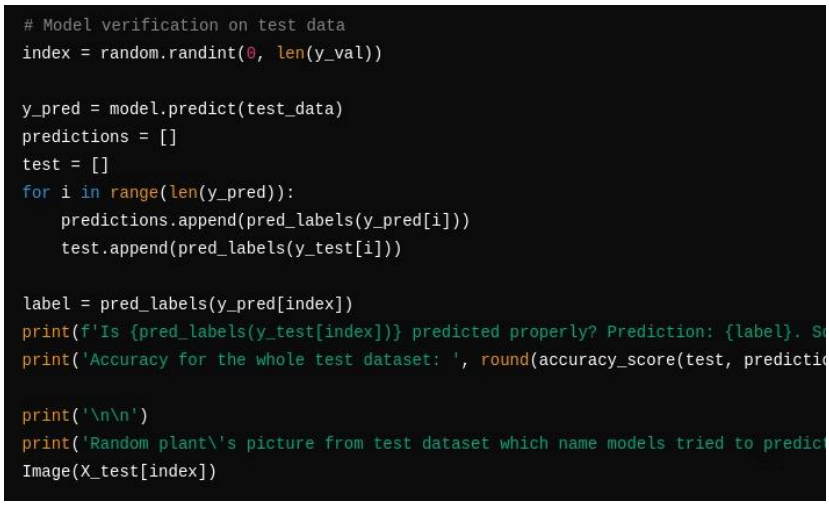
Section	**Description**
Resizing	Resizing images to a uniform size to ensure consistency and to make them compatible with the model's input requirements, improving computational efficiency.
Normalization	Scaling pixel values to a standard range (e.g., 0 to 1 or -1 to 1) to ensure uniformity across the dataset and improve model convergence during training.
Data Augmentation	Applying techniques like rotation, flipping, cropping, and scaling to artificially increase dataset size and variability, helping the model generalize better
Color Space Conversion	Converting images to different color spaces (e.g., RGB to HSV or Grayscale) to emphasize specific features like color or brightness, aiding in better identification and classification.
Image Cropping	Cropping regions of interest from images to focus on specific parts of the produce, removing unnecessary background and reducing computational load for model training.
Batch Normalization	Normalizing the output of each layer within the network during training to accelerate convergence, reduce overfitting, and

	stabilize the learning process.
Data Preprocessing Code Screenshots	

Resizing	`def prepare_image(image_path):`
	`  '''`
	`  Turns an image into a tensor`
	`  '''`
	`  # Read an image`
	`  image = tf.io.read_file(image_path)`
	`  # Turn an image to numerical version`
	`  image = tf.image.decode_jpeg(image, channels=3)`
	`  # Convert colors from 0-255 to 0-1`
	`  image = tf.image.convert_image_dtype(image, tf.float32)`
	`  # Resize`
	`  image = tf.image.resize(image, size=[img_size, img_size])`
	
	`  return image`

Normalization	`def show_image_samples(gen):`
	`  t_dict = gen.class_indices`
	`  classes = list(t_dict.keys())`
	`  images, labels = next(gen)  # Get a sample batch from the generator`
	`  plt.figure(figsize=(20, 20))`
	`  length = len(labels)`
	`  if length < 25:  # Show maximum of 25 images`
	`  r = length`
	`  else:`
	`  r = 25`
	`  for i in range(r):`
	`  plt.subplot(5, 5, i + 1)`
	`  image = images[i] / 255  # Normalize`
	`  plt.imshow(image)`
	`  index = np.argmax(labels[i])`
	`  class_name = classes[index]`
	`  plt.title(class_name, color='blue', fontsize=14)`
	`  plt.axis('off')`
	`  plt.show()`

Data Augmentation	`from tensorflow.keras.preprocessing.image import ImageDataGenerator`
	
	`datagen = ImageDataGenerator(`
	`  rotation_range=30,`
	`  width_shift_range=0.2,`
	`  height_shift_range=0.2,`
	`  shear_range=0.2,`
	`  zoom_range=0.2,`
	`  horizontal_flip=True,`
	`  fill_mode='nearest'`
	`)`

Color Space Conversion	
Image Cropping	
Batch Normalization