



Data Collection and Preprocessing Phase

Date	27 June 2024
Team ID	SWTID1720428909
Project Title	Vitamin Vision: Unveiling the Spectrum of Nutrient Detection
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	Dataset to be used is going to contain multiple different pictures of different types of edibles: In total 8968 images and 224 test images in .jpg format.
Resizing	rescale=1./255
Normalization	All images were convert to YUV format to achieve best accuracy: enhance_and_convert_to_yuv: image = tf.image.adjust_brightness(image, 0.5) image = tf.image.adjust_contrast(image, 1.5) image = tf.image.rgb_to_yuv(image)
Data Augmentation	zoom_range=0.2, horizontal_flip=True, shear_range=0.2
Denoising	Apply Gaussian blur to reduce noise: blurred = cv2.GaussianBlur(gray, (5, 5), 0)





Edge Detection	Apply edge detection algorithms to highlight prominent edges in the images.	
Color Space Conversion	Convert the image to YUV: image = tf.image.rgb_to_yuv(image)	
Image Cropping		
Batch Normalization		
Data Preprocessing Code Screenshots		
Loading Data	[] tfversion_ \$\frac{1}{2} \frac{1}{2}	
Resizing	[11] train_datagen = ImageDataGenerator(rescale=1./255,zoom_range=0.2,horizontal_flip=True,shear_range=0.2) test_datagen = ImageDataGenerator(rescale=1./255, preprocessing_function=enhance_and_convert_to_yuv)	
Normalization	### prompt: write the code to enchance piture quality for better accuracy and change it into best image quality type def enhance_and_convert_to_yuv(image): # Enhance the image image = tt.image_adjust_prightness(image, 0.2) image = tt.image_adjust_contrast(image, 1.2) image = tt.image_adjust_to_adjust_not(image, 1.2) image = tt.image_adjust_to_image, 0.1) image = tt.image_random_illo_left_right(image) image = tt.image_random_illo_pdom(image) #### Convert the image to YVV #### image = tt.image_righ_to_yov(image) return image ###################################	
Data Augmentation	[11] train_datagen = ImageOutaGenerator(rescale=1./255,zoom_range=0.2,horizontal_flip=True,shear_range=0.2) test_datagen = ImageOutaGenerator(rescale=1./255)	





Denoising	<pre>def edge_detect(image): # Convert the image to grayscale gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY) # Apply Gaussian blur to reduce noise blurred = cv2.GaussianBlur(gray, (5, 5), 0)</pre>
Edge Detection	Supperted code may be subject to a license malmocolbuhan/756/moice information entract husseominesylice-glains-segmentation at prompt; write a code for edge detection for given dataset for best accuracy import cv2 import roughy as np def edge_detect(lange): ### Convert the lange to grayscale gray = cv2.cvtclor(lange, vv2.ctocom_Robzoduv) ### Apply Gaussian blue to reduce noise blurred = cv2.caussianBlue(gray, (5, 5), 0) ### Use Canny edge_detection edges = cv2.cvtcolor(blurred, 50, 150) ### Convert the edges to a 3-channel lange edges_2 c = cv2.cvtcolor(edges, cv2.cotom_Gav2xcos) return edges_3c ### Noidify the TangeDataGenerator to apply edge_detection train_datagen = TangeDataGenerator to apply edge_detect(enhance_and_convert_to_ywv(img)) test_datagen = ImageDataGenerator(rescale=1./7255 preprocessing_function=lambda img: edge_detect(enhance_and_convert_to_ywv(img)) test_datagen = ImageDataGenerator(rescale=1./7256 preprocessing_function=lambda img: edge_detect(enhance_and_convert_to_ywv(img)) Convert to edge Convert to edge Convert_to_ywv(img)
Color Space Conversion	<pre># Convert the image to YUV image = tf.image.rgb_to_yuv(image)</pre>
Image Cropping	
Batch Normalization	