

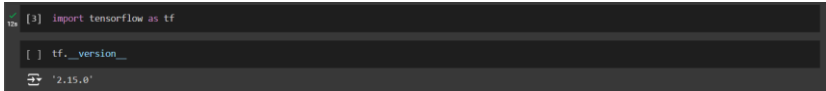
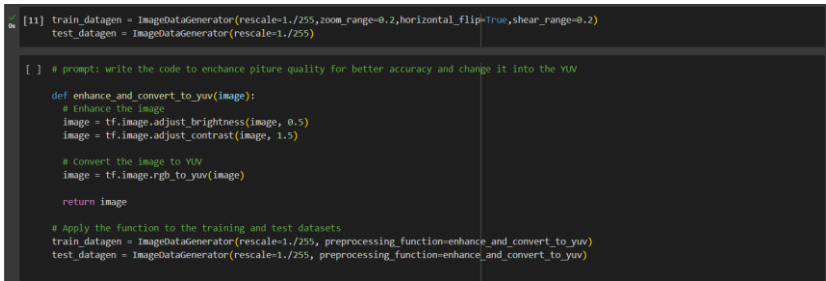
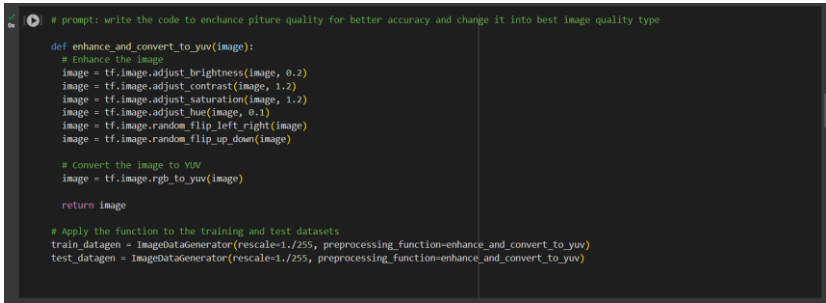
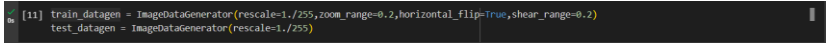
## Data Collection and Preprocessing Phase

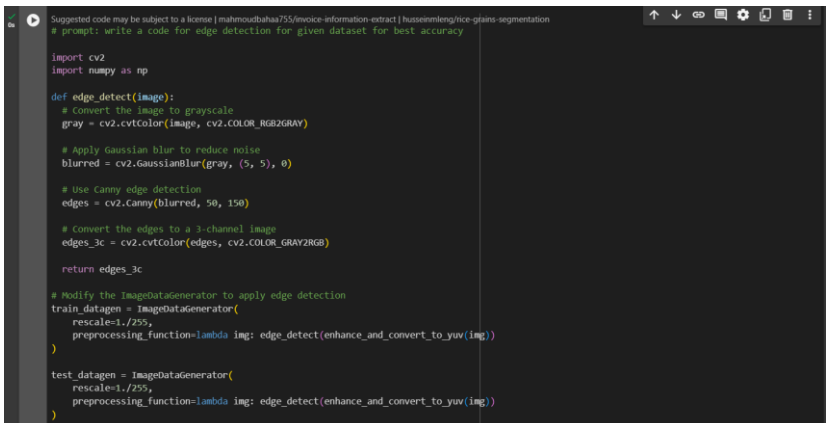
Date	27 June 2024
Team ID	SWTID1720428909
Project Title	Vitamin Vision: Unveiling the Spectrum of Nutrient Detection
Maximum Marks	6 Marks

### 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	----
<b>Data Preprocessing Code Screenshots</b>	
Loading Data	
Resizing	
Normalization	
Data Augmentation	

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	 <pre>Suggested code may be subject to a license [mahmoudbahar75/invoice-information-extract   husseinmg/rice-plants-segmentation] # prompt: write a code for edge detection for given dataset for best accuracy  import cv2 import numpy as np  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)      # Use Canny edge detection     edges = cv2.Canny(blurred, 50, 150)      # Convert the edges to a 3-channel image     edges_3c = cv2.cvtColor(edges, cv2.COLOR_GRAY2RGB)      return edges_3c  # Modify the ImageDataGenerator to apply edge detection train_datagen = ImageDataGenerator(     rescale=1./255,     preprocessing_function=lambda img: edge_detect(enhance_and_convert_to_yuv(img)) )  test_datagen = ImageDataGenerator(     rescale=1./255,     preprocessing_function=lambda img: edge_detect(enhance_and_convert_to_yuv(img)) )</pre>
Color Space Conversion	<pre># Convert the image to YUV image = tf.image.rgb_to_yuv(image)</pre>
Image Cropping	----
Batch Normalization	----