Data Collection and Preprocessing Phase

Date	13 Feb 2026
Student Name	Pratik Rajendra Pawar
Project Title	GreenClassify: A Vegetable Classifier

Data Preprocessing

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 project uses image datasets of 15 vegetable species. The images are collected from various sources including SmartInternz, custom field-captured images, and platforms like Kaggle and Open Food Facts. This ensures rich visual diversity and robust generalization during training.
Resizing	All images are resized to **224×224 pixels** using OpenCV's cv2.resize() function to ensure uniform input dimensions for CNN-based models.
Normalization	Pixel values are normalized to the range **[0, 1]** by dividing by 255.0, improving convergence during model training.
Data Augmentation	Using ImageDataGenerator, images are augmented with **random rotation**, **shifts**, **zoom**, **horizontal/vertical flips**, and **fill modes** to avoid overfitting.
Denoising	OpenCV's fastNlMeansDenoisingColored() is applied to reduce environmental noise and improve image clarity, especially for field-captured data.

Edge Detection	cv2.Canny() is used for edge detection, helping to emphasize structure, texture, and contour features of different mushroom species.
Color Space Conversion	Images are converted from **BGR to HSV** color space using cv2.cvtColor() to better capture color-based patterns across lighting variations.
Image Cropping	Manual center cropping is done on some images to focus on the mushroom body and reduce irrelevant background noise, enhancing object recognition.
Batch Normalization	BatchNormalization() is applied in the neural network model to stabilize and accelerate the learning process by reducing internal covariate shift.

Data Preprocessing Code Screenshots

Loading Data

```python
import streamlit as st
import torch
import cv2
import numpy as np
from PIL import Image
import plotly.express as px
from model import get_model
from torchvision import transforms

# Nutritional info (calories and vitamins per 100g)
nutrition_info = {
    "Bean": {"Calories": "31 kcal", "Vitamins": "Vitamin C, K"},
    "Bitter Gourd": {"Calories": "17 kcal", "Vitamins": "Vitamin A, C"},
    "Bottle Gourd": {"Calories": "14 kcal", "Vitamins": "Vitamin C"},
    "Brinjal": {"Calories": "25 kcal", "Vitamins": "Vitamin C, K"},
    "Broccoli": {"Calories": "34 kcal", "Vitamins": "Vitamin C, K"},
    "Cabbage": {"Calories": "25 kcal", "Vitamins": "Vitamin C, K"},
    "Capsicum": {"Calories": "26 kcal", "Vitamins": "Vitamin A, C"},
    "Carrot": {"Calories": "41 kcal", "Vitamins": "Vitamin A, K"},
    "Cauliflower": {"Calories": "25 kcal", "Vitamins": "Vitamin C, K"},
    "Cucumber": {"Calories": "16 kcal", "Vitamins": "Vitamin K"},
    "Papaya": {"Calories": "43 kcal", "Vitamins": "Vitamin A, C"},
    "Potato": {"Calories": "77 kcal", "Vitamins": "Vitamin C, B6"},
    "Pumpkin": {"Calories": "26 kcal", "Vitamins": "Vitamin A, C"},
    "Radish": {"Calories": "16 kcal", "Vitamins": "Vitamin C"},
    "Tomato": {"Calories": "18 kcal", "Vitamins": "Vitamin C, K"}
}

def load_model(num_classes, model_path="vegetable_classifier.pth"):
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    model = get_model(num_classes)
    model.load_state_dict(torch.load(model_path, map_location=device))
    model.eval()
    return model, device

def preprocess_image(image):
    transform = transforms.Compose([
        transforms.ToPILImage(),
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
    ])
    return transform(image).unsqueeze(0)
```

Resizing

```python
img = cv2.resize(img, (224, 224))
```

Normalization	`data = np.array(data, dtype=object)`
	`X = np.array([i[0] for i in data]) / 255.0  # Normalize pixel values`
	`y = np.array([i[1] for i in data])`

Data Augmentation	`train_datagen = ImageDataGenerator(`
	`  rotation_range=30,`
	`  width_shift_range=0.1,`
	`  height_shift_range=0.1,`
	`  zoom_range=0.2,`
	`  horizontal_flip=True,`
	`  vertical_flip=True,`
	`  fill_mode='nearest'`
	`)`

Denoising	`denoised_img = cv2.fastNlMeansDenoisingColored(img, None, 10, 10, 7, 21)`
Edge Detection	`edges = cv2.Canny(img, threshold1=100, threshold2=200)`
Color Space Conversion	`hsv_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)`
Image Cropping	`cropped_img = img[30:194, 30:194]  # Manual center crop`

Batch Normalization	`from tensorflow.keras.layers import BatchNormalization`
	
	`model.add(BatchNormalization())`