Data Collection and Preprocessing Phase

Date	6 July 2024
Team ID	SWTID1720076593
Project Title	Visual Diagnostics: Detecting Tomato Plant Diseases through Leaf Image Analysis
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	The Kaggle Tomato Diseases Dataset provides a comprehensive collection of images depicting various conditions of tomato leaves, including healthy leaves and those affected by different diseases. This dataset is instrumental in developing and training deep learning models for disease detection and classification.
Resizing	Resizing images to a specified target size standardizes the input dimensions for neural networks. It improves model performance and reduces computational load.
Normalization	Normalizing pixel values ensures that all image inputs have a consistent range of values, typically between 0 and 1 or -1 and 1. This process helps improve the convergence rate of neural networks during training.
Data Augmentation	Data augmentation generates new training samples by applying random transformations to the original images, enhancing model robustness and reducing overfitting.
Denoising	Denoising filters reduce unwanted noise in images, enhances the quality and improves the performance of image processing and analysis tasks.

Edge Detection	Edge detection algorithms highlight prominent edges in images, which are crucial for identifying boundaries and features, enhancing image analysis and object detection tasks.
Color Space Conversion	Color space conversion is used to transform images from one color model to another, which can be useful for various image processing tasks like segmentation, filtering, and feature extraction.
Image Cropping	Cropping images to focus on regions containing objects of interest is a common pre-processing technique in computer vision, especially for tasks like object detection and classification.
Batch Normalization	Batch normalization is a powerful technique used in deep neural networks to improve training stability and speed.

Data Preprocessing Code Screenshots

Loading Data

```
training_set = tf.keras.utils.image_dataset_from_directory(
    'train',
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=32,
    image_size=(128, 128),
    shuffle=True,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False
)

validation_set = tf.keras.utils.image_dataset_from_directory(
    'valid',
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=32,
    image_size=(128, 128),
    shuffle=True,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False
)
```

Resizing

```
def resize_images(image, label):
    image = tf.image.resize(image, [128, 128])
    return image, label

training_set = training_set.map(resize_images)
validation_set = validation_set.map(resize_images)
```

Normalization	`training_set = training_set.map(lambda x, y: (tf.cast(x, tf.float32) / 255.0, y))` `validation_set = validation_set.map(lambda x, y: (tf.cast(x, tf.float32) / 255.0, y))`

Data Augmentation	`from tensorflow.keras.preprocessing.image import ImageDataGenerator`
	
	`datagen = ImageDataGenerator(`
	`  rotation_range=40,`
	`  width_shift_range=0.2,`
	`  height_shift_range=0.2,`
	`  shear_range=0.2,`
	`  zoom_range=0.2,`
	`  horizontal_flip=True,`
	`  fill_mode='nearest'`
	`)`
	
	`train_generator = datagen.flow_from_directory(`
	`  'train',`
	`  target_size=(128, 128),`
	`  batch_size=32,`
	`  class_mode='categorical'`
	`)`

Denoising	`import cv2`
	
	`def denoise_image(image):`
	`  return cv2.fastNlMeansDenoisingColored(image, None, 10, 10, 7, 21)`
	
	`training_set = training_set.map(lambda x, y: (tf.numpy_function(denoise_image, [x], tf.float32), y))`
	`validation_set = validation_set.map(lambda x, y: (tf.numpy_function(denoise_image, [x], tf.float32), y))`

Edge Detection	`def edge_detection(image):`
	`  return cv2.Canny(image, 100, 200)`
	
	`training_set = training_set.map(lambda x, y: (tf.numpy_function(edge_detection, [x], tf.float32), y))`
	`validation_set = validation_set.map(lambda x, y: (tf.numpy_function(edge_detection, [x], tf.float32), y))`

Color Space Conversion	`def convert_to_grayscale(image):`
	`  return cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)`
	
	`training_set = training_set.map(lambda x, y: (tf.numpy_function(convert_to_grayscale, [x], tf.float32), y))`
	`validation_set = validation_set.map(lambda x, y: (tf.numpy_function(convert_to_grayscale, [x], tf.float32), y))`

Image Cropping	`def crop_image(image):`
	`  return image[16:112, 16:112]`
	
	`training_set = training_set.map(lambda x, y: (tf.numpy_function(crop_image, [x], tf.float32), y))`
	`validation_set = validation_set.map(lambda x, y: (tf.numpy_function(crop_image, [x], tf.float32), y))`

Batch Normalization	`from keras.layers import BatchNormalization`
	
	`model = Sequential()`
	`model.add(Conv2D(filters=32, kernel_size=3, padding='same', activation='relu', input_shape=[128, 128, 3]))`
	`model.add(BatchNormalization())`
	`model.add(Conv2D(filters=32, kernel_size=3, activation='relu'))`
	`model.add(BatchNormalization())`
	`model.add(MaxPooling2D(pool_size=2, strides=2))`
	`model.add(Conv2D(filters=64, kernel_size=3, padding='same', activation='relu'))`
	`model.add(BatchNormalization())`
	`model.add(Conv2D(filters=64, kernel_size=3, activation='relu'))`
	`model.add(BatchNormalization())`
	`model.add(MaxPooling2D(pool_size=2, strides=2))`
	`model.add(Conv2D(filters=128, kernel_size=3, padding='same', activation='relu'))`
	`model.add(BatchNormalization())`
	`model.add(Conv2D(filters=128, kernel_size=3, activation='relu'))`
	`model.add(BatchNormalization())`
	`model.add(MaxPooling2D(pool_size=2, strides=2))`
	`model.add(Conv2D(filters=256, kernel_size=3, padding='same', activation='relu'))`
	`model.add(BatchNormalization())`
	`model.add(Conv2D(filters=256, kernel_size=3, activation='relu'))`
	`model.add(BatchNormalization())`
	`model.add(MaxPooling2D(pool_size=2, strides=2))`
	`model.add(Conv2D(filters=512, kernel_size=3, padding='same', activation='relu'))`
	`model.add(BatchNormalization())`
	`model.add(Conv2D(filters=512, kernel_size=3, activation='relu'))`
	`model.add(BatchNormalization())`
	`model.add(MaxPooling2D(pool_size=2, strides=2))`
	`model.add(Dropout(0.25))`
	`model.add(Flatten())`
	`model.add(Dense(units=1500, activation='relu'))`
	`model.add(Dropout(0.4))`
	`model.add(Dense(units=10, activation='softmax'))`