Data Collection and Preprocessing Phase

Date	2nd July 2024
Team ID	SWTID1719992739
Project Title	Visual Diagnostics: Detecting Tomato Plant Diseases through Leaf Image Analysis
Maximum Marks	6 Marks

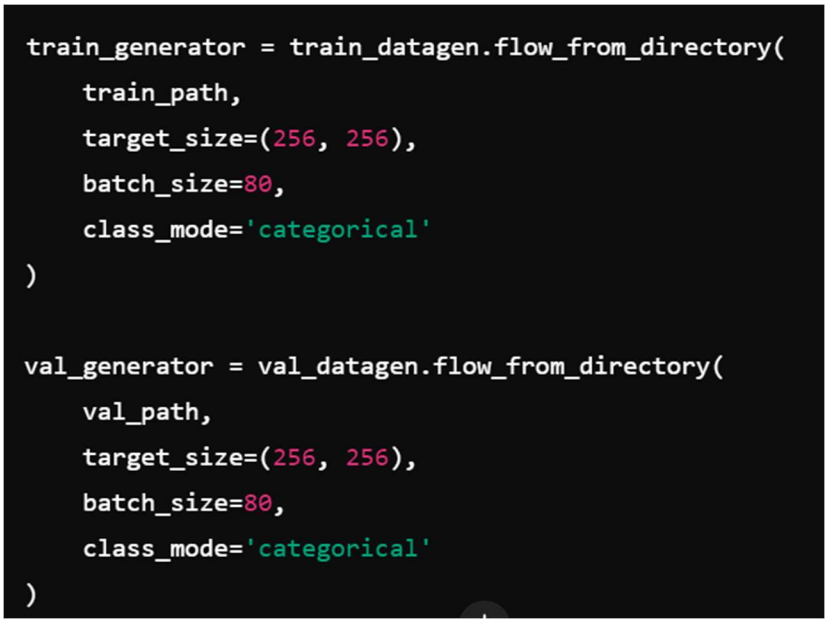
Preprocessing Template

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 dataset used for this project consists of images of tomato leaves, categorized into healthy leaves and leaves affected by various diseases. The dataset is split into training and validation sets.
Resizing	Resize images to a target size of 256x256 pixels.
Normalization	Normalize pixel values to the range [0, 1] by rescaling with a factor of 1./255.
Data Augmentation	Apply augmentation techniques such as flipping, rotation, shifting, zooming, or shearing.
Denoising	denoising is not applied in this project.
Edge Detection	Edge detection is not explicitly used in this project.

Color Space Conversion	Color space conversion is not explicitly used in this project.
Image Cropping	Image cropping is not explicitly used in this project.
Batch Normalization	Batch normalization is applied through the use of pre-trained models which include normalization layers.
Data Preprocessing Code Screenshots	

Loading Data	Python
	`from tensorflow.keras.preprocessing.image import ImageDataGenerator`
	
	`# Create an instance of ImageDataGenerator for training with augmentati`
	`train_datagen = ImageDataGenerator(`
	`rescale=1./255, # Normalize pixel values to [0, 1]`
	`rotation_range=40, # Randomly rotate images by 40 degrees`
	`width_shift_range=0.2, # Randomly shift images horizontally by 2`
	`height_shift_range=0.2, # Randomly shift images vertically by 20%`
	`shear_range=0.2, # Apply random shear transformations`
	`zoom_range=0.2, # Apply random zoom transformations`
	`horizontal_flip=True, # Randomly flip images horizontally`
	`fill_mode='nearest' # Fill mode for new pixels created during`
	`)`
	
	`# Create an instance of ImageDataGenerator for validation without augme`
	`val_datagen = ImageDataGenerator(rescale=1./255) # Only normalize pixe`

Resizing	
Normalization	

```
train_generator = train_datagen.flow_from_directory(
    train_path,
    target_size=(256, 256),
    batch_size=80,
    class_mode='categorical'
)

val_generator = val_datagen.flow_from_directory(
    val_path,
    target_size=(256, 256),
    batch_size=80,
    class_mode='categorical'
)
```

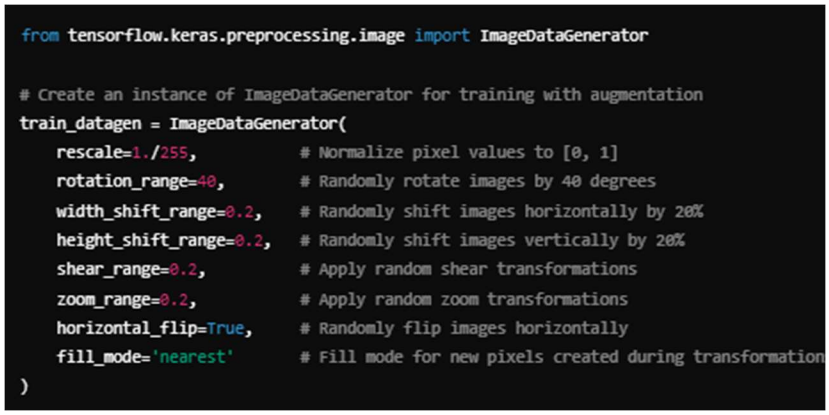
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Create an instance of ImageDataGenerator for training with augmentation
train_datagen = ImageDataGenerator(
    rescale=1./255,           # Normalize pixel values to [0, 1]
    rotation_range=40,        # Randomly rotate images by 40 degrees
    width_shift_range=0.2,    # Randomly shift images horizontally by 20%
    height_shift_range=0.2,   # Randomly shift images vertically by 20%
    shear_range=0.2,          # Apply random shear transformations
    zoom_range=0.2,           # Apply random zoom transformations
    horizontal_flip=True,     # Randomly flip images horizontally
    fill_mode='nearest'       # Fill mode for new pixels created during transformations
)

# Create an instance of ImageDataGenerator for validation without augmentation
val_datagen = ImageDataGenerator(rescale=1./255)  # Only normalize pixel values to [0,

# Apply ImageDataGenerator functionality to training set
train_generator = train_datagen.flow_from_directory(
    train_path,               # Directory containing training images
    target_size=(256, 256),   # Resize images to 256x256 pixels
    batch_size=80,            # Number of images to yield per batch
    class_mode='categorical'  # Type of classification label array to return
)

# Apply ImageDataGenerator functionality to validation set
val_generator = val_datagen.flow_from_directory(
    val_path,                 # Directory containing validation images
    target_size=(256, 256),   # Resize images to 256x256 pixels
    batch_size=80,            # Number of images to yield per batch
    class_mode='categorical'  # Type of classification label array to return
)
```

Data Augmentation	
Denoising	-
Edge Detection	-
Color Space Conversion	-
Image Cropping	
Batch Normalization	-