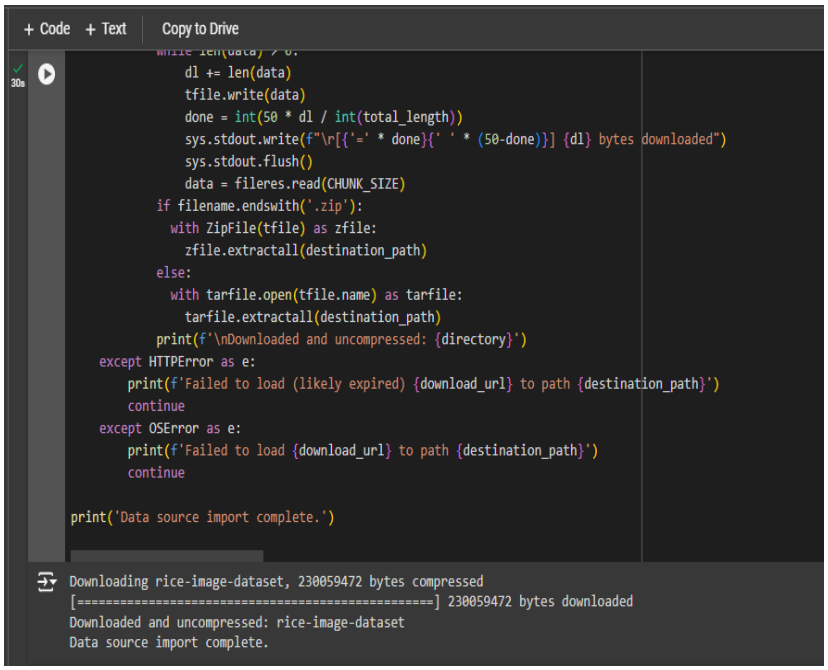
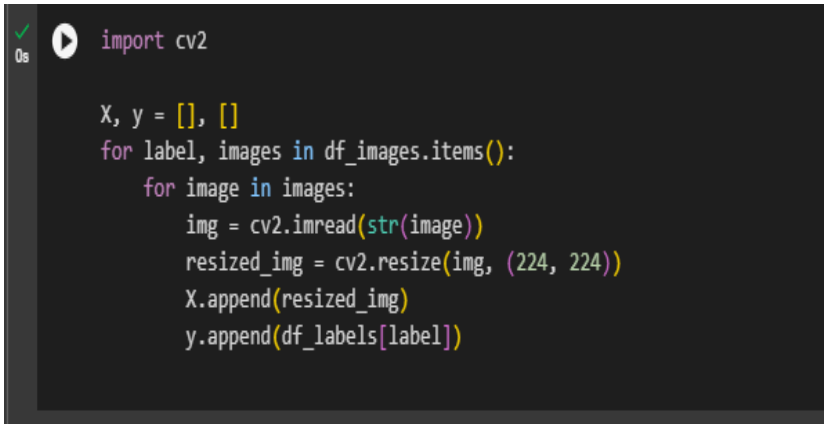
Data Collection and Preprocessing Phase

Date	12 July 2024
Team ID	SWTID1720085076
Project Title	Rice Type Classification using CNN
Maximum Marks	6 Marks

Preprocessing

The images will be preprocessed by resizing, normalizing, augmenting, 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 consists of images of different rice varieties, including Arborio, Basmati, Ipsala, Jasmine, and Karacadag. Each class contains 600 images. The images are stored in directories named after their respective classes. This dataset will be used for training a convolutional neural network (CNN) to classify rice varieties.
Resizing	All images are resized to a target size of 224x224 pixels using OpenCV's `resize` function. This resizing is necessary to ensure that the images can be fed into the MobileNetV2 model, which requires a fixed input size
Normalization	Pixel values of the images are normalized to the range [0, 1] by dividing by 255. This helps in speeding up the training process and improving model performance.
Data Augmentation	Data augmentation techniques such as flipping, rotation, shifting, zooming, or shearing were not explicitly mentioned in the provided code. However, these techniques can be applied using TensorFlow's `ImageDataGenerator` or other libraries to increase the diversity of the training dataset and prevent overfitting.

Image Cropping	Image cropping to focus on regions containing objects of interest was not explicitly done in the provided code. This can be achieved using OpenCV's cropping capabilities if specific regions of the images need to be focused on.
Batch Normalization	Batch normalization can be applied using TensorFlow/Keras layers to normalize the input of each layer in the neural network, improving training stability and convergence.
Data Preprocessing Code Screenshots	
Loading Data	
Resizing	

```
+ Code  + Text  Copy to Drive
30s
            dl += len(data)
            tfile.write(data)
            done = int(50 * dl / int(total_length))
            sys.stdout.write(f"\r[{'=' * done}{' ' * (50-done)}] {dl} bytes downloaded")
            sys.stdout.flush()
            data = fileres.read(CHUNK_SIZE)
          if filename.endswith('.zip'):
            with ZipFile(tfile) as zfile:
              zfile.extractall(destination_path)
          else:
            with tarfile.open(tfile.name) as tarfile:
              tarfile.extractall(destination_path)
          print(f'\nDownloaded and uncompressed: {directory}')
    except HTTPError as e:
        print(f'Failed to load (likely expired) {download_url} to path {destination_path}')
        continue
    except OSError as e:
        print(f'Failed to load {download_url} to path {destination_path}')
        continue

print('Data source import complete.')

Downloading rice-image-dataset, 230059472 bytes compressed
[==================================================] 230059472 bytes downloaded
Downloaded and uncompressed: rice-image-dataset
Data source import complete.
```

```
0s
import cv2

X, y = [], []
for label, images in df_images.items():
    for image in images:
        img = cv2.imread(str(image))
        resized_img = cv2.resize(img, (224, 224))
        X.append(resized_img)
        y.append(df_labels[label])
```

Normalization	

```python
# Load images and labels
X, y = [], []
for label, images in df_images.items():
    for image in images:
        img = cv2.imread(str(image))
        resized_img = cv2.resize(img, (224, 224))
        X.append(resized_img)
        y.append(df_labels[label])

# Standardizing the images
X = np.array(X) / 255.0
y = np.array(y)
```

Data Augmentation	

```python
from keras.preprocessing.image import ImageDataGenerator
# Set the image size and batch size
image_size = (224, 224)
batch_size = 32

# Create an ImageDataGenerator object with data augmentation options for image preprocessing
datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=45,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)


# Create a generator for the training data
train_generator = datagen.flow_from_dataframe(
    df_train,
    x_col='image',
    y_col='label',
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=True
)

# Create a generator for the test data
test_generator = datagen.flow_from_dataframe(
    df_test,
    x_col='image',
    y_col='label',
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)
```

```
Found 60000 validated image filenames belonging to 5 classes.
Found 15000 validated image filenames belonging to 5 classes.
```

Image Cropping	3s
	``python
	# Load images and labels with cropping
	X, y = [], []
	for label, images in df_images.items():
	    for image in images:
	        img = cv2.imread(str(image))
	        # Crop the image (e.g., central crop)
	        height, width, _ = img.shape
	        start_row, start_col = int(height * 0.1), int(width * 0.1)
	        end_row, end_col = int(height * 0.9), int(width * 0.9)
	        cropped_img = img[start_row:end_row, start_col:end_col]
	        resized_img = cv2.resize(cropped_img, (224, 224))
	        X.append(resized_img)
	        y.append(df_labels[label])
	``
	15s

Batch Normalization	```python
	# Load MobileNetV2 from TensorFlow Hub
	mobile_net_url = 'https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4'
	mobile_net = hub.KerasLayer(mobile_net_url, input_shape=(224, 224, 3), trainable=False)
	
	# Build the model
	num_labels = 5
	model = keras.Sequential([
	    mobile_net,
	    keras.layers.BatchNormalization(), # Add Batch Normalization
	    keras.layers.Dense(num_labels)
	])
	``
	15s