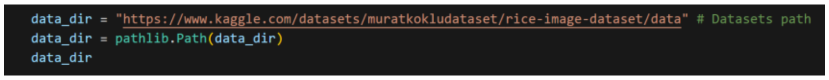
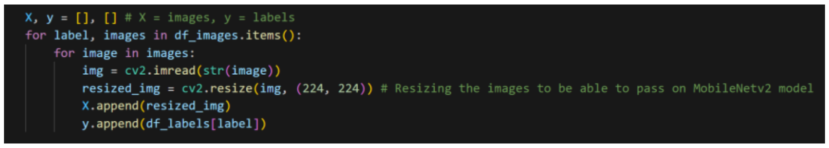
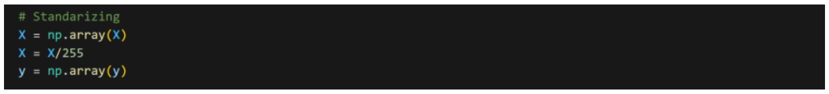
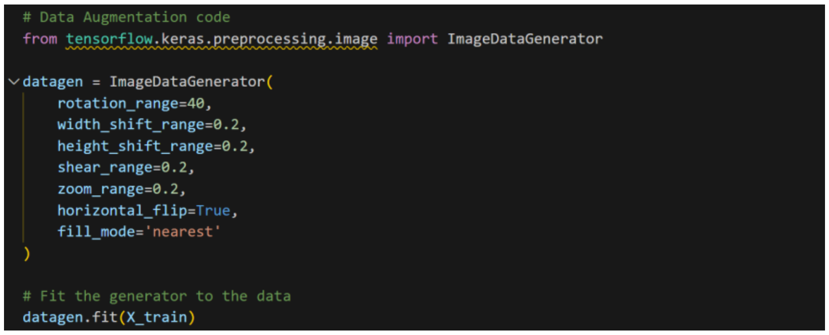
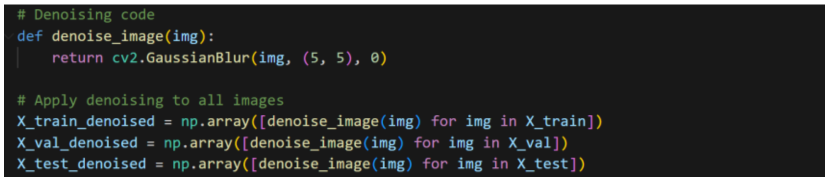
# Data Collection and Preprocessing Phase

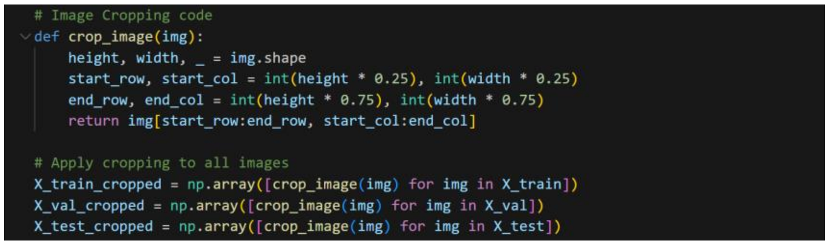
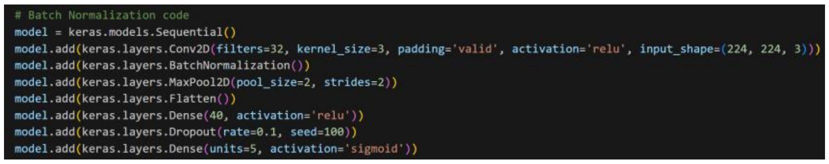
| | |
|---|---|
| Date | 8 July 2024 |
| Team ID | SWTID1720078167 |
| Project Title | Rice Type Classification using CNN |
| Maximum Marks | 6 Marks |

**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 consists of rice images from five different categories: Arborio, Basmati, Ipsala, Jasmine, and Karacadag. Each category contains 15000 images, and the images are stored in directories named after these categories. |
| Resizing | Images are resized to 224x224 pixels to be compatible with the MobileNetV2 model |
| Normalization | Normalization is performed by dividing the pixel values by 255, bringing them to the range [0, 1]. |
| Data Augmentation | Data augmentation techniques such as flipping, rotation, and zooming can be applied using the ImageDataGenerator from Keras. |
| Denoising | Denoising filters like Gaussian Blur can be applied to reduce noise in images. |
| Edge Detection | - |

| | |
|---|---|
| Color Space Conversion | - |
| Image Cropping | Images can be cropped to focus on regions containing objects of interest. |
| Batch Normalization | Batch normalization is applied to the inputs of each layer in the neural network. |
| **Data Preprocessing Code Screenshots** | |
| Loading Data | <br>`data_dir = "https://www.kaggle.com/datasets/muratkokludataset/rice-image-dataset/data" # Datasets path`<br>`data_dir = pathlib.Path(data_dir)`<br>`data_dir` |
| Resizing | <br>`X, y = [], [] # X = images, y = labels`<br>`for label, images in df_images.items():`<br>`    for image in images:`<br>`        img = cv2.imread(str(image))`<br>`        resized_img = cv2.resize(img, (224, 224)) # Resizing the images to be able to pass on MobileNetv2 model`<br>`        X.append(resized_img)`<br>`        y.append(df_labels[label])` |
| Normalization | <br>`# Standarizing`<br>`X = np.array(X)`<br>`X = X/255`<br>`y = np.array(y)` |
| Data Augmentation | <br>`# Data Augmentation code`<br>`from tensorflow.keras.preprocessing.image import ImageDataGenerator`<br><br>`datagen = ImageDataGenerator(`<br>`    rotation_range=40,`<br>`    width_shift_range=0.2,`<br>`    height_shift_range=0.2,`<br>`    shear_range=0.2,`<br>`    zoom_range=0.2,`<br>`    horizontal_flip=True,`<br>`    fill_mode='nearest'`<br>`)`<br><br>`# Fit the generator to the data`<br>`datagen.fit(X_train)` |
| Denoising | <br>`# Denoising code`<br>`def denoise_image(img):`<br>`    return cv2.GaussianBlur(img, (5, 5), 0)`<br><br>`# Apply denoising to all images`<br>`X_train_denoised = np.array([denoise_image(img) for img in X_train])`<br>`X_val_denoised = np.array([denoise_image(img) for img in X_val])`<br>`X_test_denoised = np.array([denoise_image(img) for img in X_test])` |
| Edge Detection | - |
| Color Space Conversion | - |

| | |
|---|---|
| Image Cropping |  |
| Batch Normalization |  |

```python
# Image Cropping code
def crop_image(img):
    height, width, _ = img.shape
    start_row, start_col = int(height * 0.25), int(width * 0.25)
    end_row, end_col = int(height * 0.75), int(width * 0.75)
    return img[start_row:end_row, start_col:end_col]

# Apply cropping to all images
X_train_cropped = np.array([crop_image(img) for img in X_train])
X_val_cropped = np.array([crop_image(img) for img in X_val])
X_test_cropped = np.array([crop_image(img) for img in X_test])
```

```python
# Batch Normalization code
model = keras.models.Sequential()
model.add(keras.layers.Conv2D(filters=32, kernel_size=3, padding='valid', activation='relu', input_shape=(224, 224, 3)))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.MaxPool2D(pool_size=2, strides=2))
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(40, activation='relu'))
model.add(keras.layers.Dropout(rate=0.1, seed=100))
model.add(keras.layers.Dense(units=5, activation='sigmoid'))
```