```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import classification_report, confusion_matrix
```

These lines import the necessary libraries:

tensorflow for building and training neural networks.

Sequential, Conv2D, MaxPooling2D, Flatten, and Dense from tensorflow.keras.layers for building the CNN model.

ImageDataGenerator from tensorflow.keras.preprocessing.image for image data augmentation and preprocessing.

matplotlib.pyplot as plt for data visualization.

numpy as np for numerical computations.

classification_report and confusion_matrix from sklearn.metrics for evaluating the model.

```
# Define paths to the directories
train_dir = 'C:\\Users\\Rajabi\\Desktop\\NASA\\RSSCN7-master\\New\\train'
validation_dir = 'C:\\Users\\Rajabi\\Desktop\\NASA\\RSSCN7-master\\New\\validation'
test_dir = 'C:\\Users\\Rajabi\\Desktop\\NASA\\RSSCN7-master\\New\\test'
```

These lines define the paths to the directories containing the training, validation, and test images.

```
# Define image data generators
train_datagen = ImageDataGenerator(rescale=1./255)
validation_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)
```

These lines create ImageDataGenerator objects for training, validation, and test sets. The rescale=1./255 parameter normalizes pixel values to the range [0,1].

```
# Create generators for training, validation, and test sets
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=target_size,
    batch_size=batch_size,
    class_mode='categorical')

validation_generator = validation_datagen.flow_from_directory(
    validation_dir,
```

```
    target_size=target_size,
    batch_size=batch_size,
    class_mode='categorical')

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=target_size,
    batch_size=batch_size,
    class_mode='categorical')
```

These lines create generator objects for generating batches of augmented images for training, validation, and test sets using the specified directories, target size, batch size, and class mode (categorical).

```
# Define CNN model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(400, 400, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(7, activation='softmax')
])
```

This block defines the CNN model architecture using a Sequential model with convolutional, max pooling, flatten, and dense layers. The input shape is set to (400, 400, 3) for RGB images.

```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

This line compiles the model, specifying the optimizer, loss function, and evaluation metrics for training.

```
# Train the model
epochs = 20

history = model.fit(
    train_generator,
    epochs=epochs,
    validation_data=validation_generator
)
```

This code trains the model using the fit method, specifying the training generator, number of epochs, and validation data for monitoring model performance during training.

```
# Evaluate the model on test set
test_loss, test_acc = model.evaluate(test_generator)
print("Test Accuracy:", test_acc)
```
These lines evaluate the trained model on the test set and print the test accuracy.

```
# Confusion Matrix
test_predictions = model.predict(test_generator)
test_predictions = np.argmax(test_predictions, axis=1)

true_classes = test_generator.classes
class_labels = list(test_generator.class_indices.keys())

conf_matrix = confusion_matrix(true_classes, test_predictions)
print("Confusion Matrix:")
print(conf_matrix)

plt.imshow(conf_matrix, cmap='binary')
plt.xticks(ticks=range(len(class_labels)), labels=class_labels, rotation=45)
plt.yticks(ticks=range(len(class_labels)), labels=class_labels)
plt.colorbar()
plt.show()
```
This code computes and displays the confusion matrix based on the model's predictions on the test set, providing insights into the model's performance across different classes.

```
# Plot Training History
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```
This block plots the training and validation accuracy over epochs, visualizing the model's learning progress during training.

A test accuracy of approximately **54.52%** indicates the performance of your convolutional neural network (CNN) model on the test set. This means that around 54.52% of the images in the test set were classified correctly by the model.

Confusion Matrix:

[[9 5 1 9 15 10 11]

 [16 3 5 11 10 7 8]

 [15 4 8 12 9 5 7]

 [12 4 7 10 16 6 5]

 [10 4 2 9 13 11 11]

 [11 2 9 9 11 10 8]

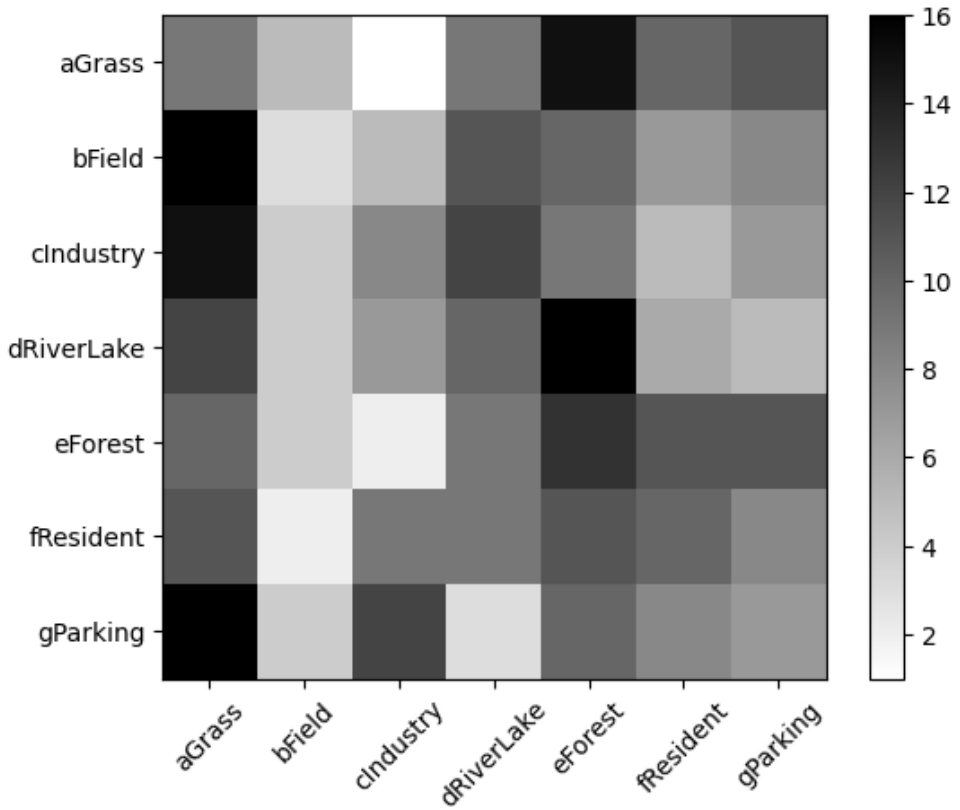
 [16 4 12 3 10 8 7]]

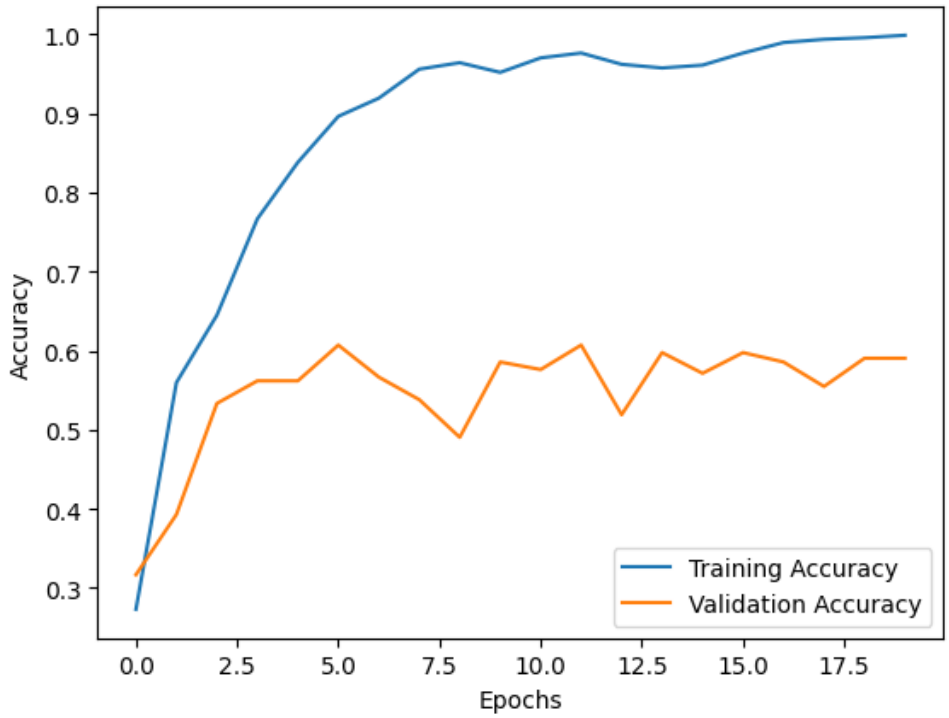
Rows represent the actual classes, while columns represent the predicted classes.

The diagonal elements (top-left to bottom-right) represent correct predictions, where the actual class matches the predicted class.

Off-diagonal elements represent misclassifications. For example, the element in row 1, column 2 (5) indicates that 5 images from class 1 were predicted as class 2.

The sum of each row gives the total number of instances for each actual class, and the sum of each column gives the total number of predictions for each predicted class.




1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
Accuracy
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epochs
Training Accuracy
Validation Accuracy