import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Path to your train, test, and validate folders

train\_dir = '/content/drive/MyDrive/waste project/data/train'

test\_dir= '/content/drive/MyDrive/waste project/data/test'

validation\_dir = '/content/drive/MyDrive/waste project/data/valid'

# Define image dimensions and batch size

img\_width, img\_height = 150, 150

batch\_size = 32

# Create data generators with data augmentation for training and validation

train\_datagen = ImageDataGenerator(

    rescale=1./255,

    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'

)

validation\_datagen = ImageDataGenerator(rescale=1./255)

# Create a generator for test data without data augmentation

test\_datagen = ImageDataGenerator(rescale=1./255)

train\_generator = train\_datagen.flow\_from\_directory(

    train\_dir,

    target\_size=(img\_width, img\_height),

    batch\_size=batch\_size,

    class\_mode='categorical'

)

validation\_generator = validation\_datagen.flow\_from\_directory(

    validation\_dir,

    target\_size=(img\_width, img\_height),

    batch\_size=batch\_size,

    class\_mode='categorical'

)

# Generate test data

test\_generator = test\_datagen.flow\_from\_directory(

    test\_dir,

    target\_size=(img\_width, img\_height),

    batch\_size=batch\_size,

    class\_mode='categorical'

)

# Build the CNN model

model = Sequential([

    Conv2D(32, (3, 3), activation='relu', input\_shape=(img\_width, img\_height, 3)),

    MaxPooling2D(2, 2),

    Conv2D(64, (3, 3), activation='relu'),

    MaxPooling2D(2, 2),

    Conv2D(128, (3, 3), activation='relu'),

    MaxPooling2D(2, 2),

    Conv2D(128, (3, 3), activation='relu'),

    MaxPooling2D(2, 2),

    Flatten(),

    Dropout(0.5),

    Dense(512, activation='relu'),

    Dense(6, activation='softmax')  # Adjust the number of output classes based on your dataset

])

# Compile the model

model.compile(optimizer='adam',

              loss='categorical\_crossentropy',

              metrics=['accuracy'])

# Train the model

history = model.fit(

    train\_generator,

    steps\_per\_epoch=train\_generator.samples // batch\_size,

    epochs=50,  # Adjust the number of epochs based on your dataset and training performance

    validation\_data=validation\_generator,

    validation\_steps=validation\_generator.samples // batch\_size

)

# Evaluate the model on test data

test\_loss, test\_acc = model.evaluate(test\_generator, verbose=2)

print('\nTest accuracy:', test\_acc)

# Save the model

model.save('waste\_classification\_model.h5')