# Import necessary libraries

from tensorflow.compat.v1 import ConfigProto, InteractiveSession

from tensorflow.keras.layers import Input, Lambda, Dense, Flatten

from tensorflow.keras.models import Model

from tensorflow.keras.applications.inception\_v3 import InceptionV3

from tensorflow.keras.preprocessing.image import ImageDataGenerator

import numpy as np

from glob import glob

import os  # Import os module

# GPU configuration (if applicable)

config = ConfigProto()

config.gpu\_options.per\_process\_gpu\_memory\_fraction = 0.5

config.gpu\_options.allow\_growth = True

session = InteractiveSession(config=config)

# Image size for InceptionV3 model

IMAGE\_SIZE = [224, 224]

# Paths for training and validation datasets

train\_path = '/content/drive/MyDrive/College Project/Dataset Rice Plant/Rice leaf disease 1.7gb/Training data'

valid\_path = '/content/drive/MyDrive/College Project/Dataset Rice Plant/Rice leaf disease 1.7gb/Testing data'

# Load the pre-trained InceptionV3 model without the top classification layer

inception = InceptionV3(input\_shape=IMAGE\_SIZE + [3], weights='imagenet', include\_top=False)

# Freeze the layers of the pre-trained model

for layer in inception.layers:

    layer.trainable = False

# Count the number of output classes (folders)

# Use os.listdir to get the subfolders within the training directory

folders = os.listdir(train\_path)  # Get subfolders from training data

# Modify the Dense layer to match the number of classes

x = Flatten()(inception.output)

prediction = Dense(len(folders), activation='softmax')(x)  # Ensure the output layer matches the number of classes

# Rebuild the model

model = Model(inputs=inception.input, outputs=prediction)

# Compile the model with categorical\_crossentropy for multi-class classification

model.compile(

  loss='categorical\_crossentropy',

  optimizer='adam',

  metrics=['accuracy']

)

# Image data generators for training and testing data

# Make sure the image size matches the input size expected by InceptionV3 (224x224)

training\_set = ImageDataGenerator(rescale=1./255).flow\_from\_directory(

    '/content/drive/MyDrive/College Project/Dataset Rice Plant/Rice leaf disease 1.7gb/Training data',

    target\_size=(224, 224),

    batch\_size=32,

    class\_mode='categorical'  # Ensures multi-class labels

)

# \*\*Change here: Set class\_mode to 'categorical' for validation data\*\*

test\_set = ImageDataGenerator(rescale=1./255).flow\_from\_directory(

    '/content/drive/MyDrive/College Project/Dataset Rice Plant/Rice leaf disease 1.7gb/Testing data',

    target\_size=(224, 224),

    batch\_size=32,  # Add batch\_size for validation data

    class\_mode='categorical'

)

# Check if the test\_set contains data

if test\_set.samples == 0:

    print("Error: Validation data directory is empty or does not contain valid images.")

else:

    # Train the model

    r = model.fit(

      training\_set,

      validation\_data=test\_set,

      epochs=5,

      steps\_per\_epoch=len(training\_set),

      validation\_steps=len(test\_set)

    )