```
from keras.models import load_model # pip install tensorflow
from keras.layers import DepthwiseConv2D # pip install tensorflow
import cv2 # pip install opencv-python
import numpy as np # pip install numpy
import os

# Disable scientific notation for clarity
np.set_printoptions(suppress=True)

# Define a custom DepthwiseConv2D class without the groups parameter
class CustomDepthwiseConv2D(DepthwiseConv2D):
    def __init__(self, **kwargs):
        # Remove the 'groups' parameter if it exists
        if 'groups' in kwargs:
            del kwargs['groups']  # Remove the groups parameter
        super().__init__(**kwargs)

# Create a dictionary of custom objects to pass to the load_model function
custom_objects = {
    'DepthwiseConv2D': CustomDepthwiseConv2D,
}


# Load the model
model = load_model("data/converted_keras/keras_model.h5", custom_objects=custom_objects, compile=False)

# Load the labels
class_names = open("data/converted_keras/labels.txt", "r").readlines()

def stillimages(pathname):
    imagenames = os.listdir(pathname)
    images = []
    for name in imagenames:
```

```
        _, extension = os.path.splitext(name)
        if (extension == ".jpg" or extension == ".
png"):
            images.append(cv2.imread(pathname + '/'
 + name))

    return images, imagenames


def predict_still(image_to_predict, imagename):

        # Resize the raw image into (224-height,224
-width) pixels
        image = cv2.resize(image_to_predict, (224,
224), interpolation=cv2.INTER_AREA)
        image = np.asarray(image, dtype=np.float32
).reshape(1, 224, 224, 3)

        # Normalize the image array
        image = (image / 127.5) - 1

        # Predicts the model
        prediction = model.predict(image)
        index = np.argmax(prediction)
        class_name = class_names[index]
        confidence_score = prediction[0][index]

        # Print prediction and confidence score
        print("Class:", class_name[2:], end="")
        print("Filename:", imagename)
        print("Confidence Score:", str(np.round(
confidence_score * 100))[:-2], "%")


# Get all the files in the folder and predict for
each of them.
images, imagenames = stillimages('data/fruits_test'
)
i = 0
for image in images:
         predict_still(image, imagenames[i])
         i = i+1
```