IMAGE RECOGNITION WITH IBM CLOUD VISUAL RECOGNITION

NAME:Saranya M

REG NO:822721104039

PHASE 3

PROGRAM FOR IMAGE RECOGNITION

GOAL:

1.The main goal and purpose of the program is to perform image classification on a user-provided image using a pre-trained InceptionV3 model.

2.The program combines image processing, a pre-trained deep learning model, and visualization techniques to classify an image provided by the user through a URL. It then displays the top predictions and the image itself.

Here are the steps and objectives :

1. **User Interaction:**

The program prompts the user to enter a URL of an image. This is done using the `input()` function which waits for user input.

2. Image Retrieval:

The ‘requests.get(url)’ function sends a GET request to the provided URL. It retrieves the image data from the internet.

3.Image Processing:

'BytesIO(response.content)' creates a byte stream from the image content obtained in the response. This stream-like object allows us to treat the binary image data as a file.

'Image.open(...)' opens the image from the byte stream.

'.convert("RGB")' converts the image to RGB mode, which is a standard color mode for images.

4. Image Resizing:

img.resize((299, 299))' resizes the image to a square of 299x299 pixels. This is a requirement for the InceptionV3 model.

5. Image to Array:

'img_to_array(img)' converts the image to a numpy array. This format is compatible with the InceptionV3 model.

6. Preprocessing for Model Input:

'preprocess_input(img)' applies necessary preprocessing to the image array,like mean subtraction, scaling, etc. This prepares the image for input into the neural network.

7. Tensor Conversion:

'tf.convert_to_tensor(img)' converts the numpy array to a TensorFlow tensor. TensorFlow uses tensors as the fundamental data structure for computations.

8. Resizing for Model Input:

‘tf.image.resize(img, (299, 299))’ resizes the image tensor to match the input size expected by the InceptionV3 model.

9. Adding Batch Dimension:

‘img[tf.newaxis, ...]’ adds an extra dimension at the beginning to represent a batch of images. This is needed for compatibility with the model.

10. Loading Pre-trained Model:

‘InceptionV3(weights='imagenet')’ loads the InceptionV3 model pre-trained on the ImageNet dataset.

11. Model Prediction:

‘model.predict(img)’ passes the preprocessed image through the neural network to get predictions.

12. Prediction Decoding:

‘decode_predictions(predictions)’ translates the raw class probabilities into human-readable labels.

13. Display Top Predictions:

The program prints out the top 5 predicted labels along with their associated probabilities.

14. Display Image:

‘plt.imshow(...)’ displays the downloaded image using matplotlib. The image is shown without axis information.

15. Display the Result:

The program outputs the predictions and displays the image.

PROGRAM :

```
from PIL import Image
import requests
from io import BytesIO
import tensorflow as tf
from tensorflow.keras.applications.inception_v3 import InceptionV3,
preprocess_input, decode_predictions
import matplotlib.pyplot as plt
url = input("Please enter the URL of the image: ")
response = requests.get(url)
img = Image.open(BytesIO(response.content)).convert("RGB")
img = img.resize((299, 299))
img = tf.keras.preprocessing.image.img_to_array(img)
img = tf.keras.applications.inception_v3.preprocess_input(img)
img = tf.convert_to_tensor(img)
img = tf.image.resize(img, (299, 299))
img = img[tf.newaxis, ...]
model = InceptionV3(weights='imagenet')
predictions = model.predict(img)
decoded_predictions = decode_predictions(predictions)[0]
print("Top predictions:")
for i, (imagenet_id, label, score) in enumerate(decoded_predictions[:5]):
print(f"{i + 1}: {label} ({score:.2f})")
plt.imshow(Image.open(BytesIO(response.content)))
plt.axis('off')
plt.show()
```