In [ ]:

```python
from tensorflow.keras.applications import VGG16
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import preprocess_input
import numpy as np
import matplotlib.pyplot as plt

model = VGG16(weights='imagenet', include_top=False)
model.summary()

img_path = 'sample.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

layer_outputs = [layer.output for layer in model.layers[:5]]
from tensorflow.keras.models import Model
activation_model = Model(inputs=model.input, outputs=layer_outputs)
activations = activation_model.predict(x)

first_layer_activation = activations[0]
plt.matshow(first_layer_activation[0, :, :, 0], cmap='viridis')
plt.show()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applicati
ons/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 ━━━━━━━━━━━━━━━━━━━━ 2s 0us/step
```

Model: "vgg16"

Layer (type)	Output Shape	Param #
input_layer_11 (InputLayer)	(None, None, None, 3)	0
block1_conv1 (Conv2D)	(None, None, None, 64)	1,792
block1_conv2 (Conv2D)	(None, None, None, 64)	36,928
block1_pool (MaxPooling2D)	(None, None, None, 64)	0
block2_conv1 (Conv2D)	(None, None, None, 128)	73,856
block2_conv2 (Conv2D)	(None, None, None, 128)	147,584
block2_pool (MaxPooling2D)	(None, None, None, 128)	0
block3_conv1 (Conv2D)	(None, None, None, 256)	295,168
block3_conv2 (Conv2D)	(None, None, None, 256)	590,080
block3_conv3 (Conv2D)	(None, None, None, 256)	590,080
block3_pool (MaxPooling2D)	(None, None, None, 256)	0
block4_conv1 (Conv2D)	(None, None, None, 512)	1,180,160
block4_conv2 (Conv2D)	(None, None, None, 512)	2,359,808
block4_conv3 (Conv2D)	(None, None, None, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, None, None, 512)	0
block5_conv1 (Conv2D)	(None, None, None, 512)	2,359,808
block5_conv2 (Conv2D)	(None, None, None, 512)	2,359,808
block5_conv3 (Conv2D)	(None, None, None, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, None, None, 512)	0

Total params: 14,714,688 (56.13 MB)

Trainable params: 14,714,688 (56.13 MB)

Non-trainable params: 0 (0.00 B)

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
/tmp/ipython-input-382938378.py in <cell line: 0>()
      9 
     10 img_path = 'sample.jpg'
---> 11 img = image.load_img(img_path, target_size=(224, 224))
     12 x = image.img_to_array(img)
     13 x = np.expand_dims(x, axis=0)

/usr/local/lib/python3.12/dist-packages/keras/src/utils/image_utils.py in load_img(path, color_mode, target_size, interpolation, keep_aspect_ratio)
    233         if isinstance(path, pathlib.Path):
    234             path = str(path.resolve())
--> 235         with open(path, "rb") as f:
    236             img = pil_image.open(io.BytesIO(f.read()))
    237     else:

FileNotFoundError: [Errno 2] No such file or directory: 'sample.jpg'
```

Exp-13 : Understanding the Architecture of a Pre-Trained Model.

Aim:

To study and analyze the structure of pre-trained CNN model such as VGG16 or ResNet50.

Description

Pre-trained models are trained on large datasets. and can be reused for feature extraction. or fine-tuning.

They contain multiple convolutional and fully connected layers designed for hierarchical feature learning.

Procedure:

1.) Import a pre-trained model (e.g., keras.applications.VGG16).

2.) Display layer names, types, and output shapes

3.) Analyze the number of parameters and filter sizes.

4.) Visualize intermediate feature maps for a sample image.

Pseudocode:

```
Load VGG16
Print model summary
select an input image.
Extract outputs from selected layers
Visualize feature maps.
```

Observation :

Early layers capture edges and colors, while deeper layers detect shapes and ~~of~~ objects. The networks show hierarchical representation learning.

Result :

The pre-trained CNN architecture was successfully analyzed, revealing how deep networks learn and organize visual features.

Output:-

Model Architecture:-

Resnet (

conv(): Conv2d(3, 64, kernel-size = (7,7), stride = (2,2) padding = (3,3), bias = False

(bn1): - - - -

:

)

Model sumary:-

Layer (type)	Output shape	Para
Lay~~er (type)~~	[-1, 64, 118, 112]	9,408
Conv 2d-1	[-1, 64, 112, 112]	128
batch Norm2d-2	[-1, 64, 56, 56]	0
MaxPool 2D-4	[-1, 1000]	513,000
Linear-68		

Total paras: 11,689,512

Trainable paras: 11689,512

Non-trainable paras: 0