```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as pllt
import cv2
```

Start coding or generate with AI.

```
vgg=tf.keras.applications.VGG16()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16
553467096/553467096 ━━━━━━━━━━━━━━━━━━━━ 4s 0us/step
```

```
vgg.summary()
```

Model: "vgg16"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1,792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36,928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73,856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147,584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295,168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590,080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590,080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102,764,544
fc2 (Dense)	(None, 4096)	16,781,312
predictions (Dense)	(None, 1000)	4,097,000

```
vgg.to_json
```

```
keras.src.models.model.Model.to_json
def to_json(**kwargs)
```

```
Args:
    **kwargs: Additional keyword arguments to be passed to
        `json.dumps()`.

Returns:
    A JSON string.
```

```
tf.keras.applications.vgg16.decode_predictions(np.ones((1,1000)),top=1000)
```

```
('n03887697', 'paper_towel', 1.0),
('n03884397', 'panpipe', 1.0),
('n03877845', 'palace', 1.0),
('n03877472', 'pajama', 1.0),
('n03876231', 'paintbrush', 1.0),
('n03874599', 'padlock', 1.0),
('n03874293', 'paddlewheel', 1.0),
('n03873416', 'paddle', 1.0),
('n03871628', 'packet', 1.0),
('n03868863', 'oxygen_mask', 1.0),
('n03868242', 'oxcart', 1.0),
('n03866082', 'overskirt', 1.0),
('n03930630', 'pickup', 1.0),
('n03933933', 'pier', 1.0),
('n03935335', 'piggy_bank', 1.0),
```

```
!wget /content/7A68FE16-3F13-40AD-A54B-C2C87194FFC8-scaled.jpeg
```

```
/content/7A68FE16-3F13-40AD-A54B-C2C87194FFC8-scaled.jpeg: Scheme missing.
```

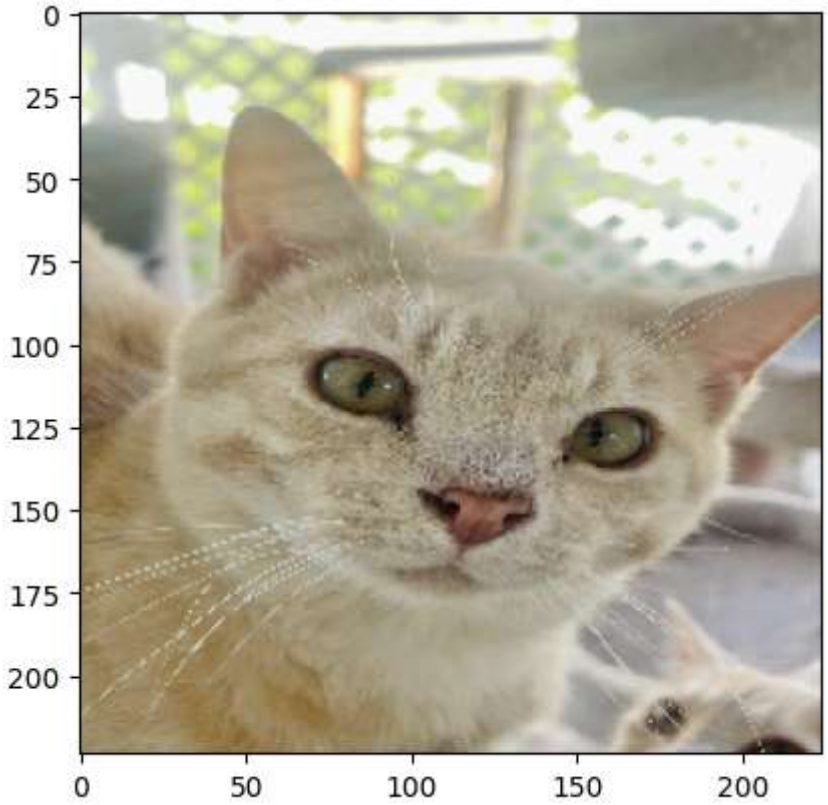
```
test_image=cv2.imread("/content/7A68FE16-3F13-40AD-A54B-C2C87194FFC8-scaled.jpeg")
```

```
test_image=cv2.cvtColor(test_image,cv2.COLOR_BGR2RGB)
```

```
import matplotlib.pyplot as plt
```

```
test_image=cv2.resize(test_image,(224,224))
plt.imshow(test_image)
```

```
<matplotlib.image.AxesImage at 0x7bba33734250>
```




```
test_image=np.expand_dims(test_image,axis=0)
```

```
#vgg preprocessing
image=tf.keras.applications.vgg16.preprocess_input(test_image)
```

```
image.shape
```

```
(1, 224, 224, 3)
```

```
plt.imshow(image[0])
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data
<matplotlib.image.AxesImage at 0x7bba2ec10c10>
```


```
result=vgg.predict(image)
result
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 700ms/step
array([[9.51911474e-08, 9.49917819e-07, 7.23380595e-07, 1.81828034e-06,
        1.25485276e-05, 2.57029369e-05, 2.40916802e-06, 6.82948439e-07,
        3.02313379e-06, 3.66768222e-06, 1.32719990e-07, 3.86068493e-08,
        7.70350539e-07, 4.07382316e-07, 4.85696212e-08, 3.55915603e-07,
        1.38720225e-06, 4.06204708e-06, 6.12845895e-07, 1.05910217e-07,
        4.29570918e-07, 2.22160020e-06, 1.01636189e-07, 1.00175475e-06,
        3.78804799e-07, 1.47594051e-07, 2.82587791e-07, 4.55793810e-07,
        2.03291862e-07, 5.87952286e-07, 1.63967655e-07, 2.07741778e-07,
        3.83789796e-07, 8.27809927e-08, 8.97060275e-08, 1.01302632e-07,
        2.32334699e-07, 1.05490777e-07, 8.29413329e-06, 3.01681354e-07,
        5.52687607e-06, 6.92509559e-07, 1.56849785e-06, 1.61646108e-06,
        5.42478074e-06, 4.88532521e-07, 4.09940321e-06, 1.99538545e-06,
        1.39129938e-07, 1.52963850e-07, 4.19196340e-07, 2.86776904e-05,
        2.42076158e-06, 1.27393253e-06, 9.35157277e-06, 8.89204159e-07,
        1.69129089e-06, 1.30000373e-07, 1.13387523e-06, 4.76534206e-06,
        4.00797444e-06, 2.49157392e-06, 1.85367423e-06, 4.98288614e-07,
        1.94205791e-06, 2.49343401e-07, 3.73876392e-05, 1.52671009e-06,
        5.10104519e-06, 1.29788100e-06, 5.39047278e-07, 3.81168604e-07,
        2.02781848e-07, 3.18905620e-07, 1.43539552e-07, 1.78913990e-06,
        7.80535800e-07, 1.09147663e-06, 1.17324475e-06, 3.29460386e-06,
        1.74567631e-07, 4.38467431e-08, 2.94869596e-06, 7.02138891e-07,
        5.85828002e-06, 4.54681071e-07, 1.57007082e-06, 4.25539127e-07,
        4.39576354e-07, 7.69647315e-08, 1.06553330e-07, 1.13125225e-07,
        1.59485737e-07, 2.25510259e-08, 4.60854244e-06, 5.35620863e-08,
        1.89489214e-07, 2.08203303e-07, 5.18933405e-07, 2.99181397e-06,
        1.83629822e-07, 6.80185330e-09, 3.92329468e-07, 1.33289700e-06,
        1.22032252e-04, 1.59692281e-05, 2.86311806e-05, 3.16202147e-07,
        1.04390665e-06, 1.32609216e-06, 3.93704653e-07, 7.78023093e-07,
        3.97776648e-06, 1.66700511e-06, 5.52221854e-06, 1.01819899e-06,
        2.11079467e-07, 4.43324495e-07, 2.06656154e-07, 1.93699137e-08,
        2.86536999e-08, 4.85378102e-08, 1.47093374e-07, 8.88316265e-08,
        9.05444267e-07, 3.96075308e-07, 1.63535702e-07, 4.09609271e-08,
        3.92864656e-08, 8.10586371e-08, 3.49897391e-07, 1.26223895e-07,
        2.76388334e-07, 6.44181853e-07, 1.11712268e-06, 2.28724630e-07,
        2.37098604e-07, 2.81496852e-07, 3.63797790e-06, 4.62396770e-08,
```