In [1]:

```python
# the Python program that loads the VGG19 model

from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.vgg16 import preprocess_input
from keras.applications.imagenet_utils import decode_predictions
import numpy as np
model = VGG16(weights='imagenet')
img_path = 'elephant.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)
# prediction
predictions = model.predict(x)
results = decode_predictions(predictions)
print(results)
```

```
1/1 [==============================] - 1s 1s/step
Downloading data from https://storage.googleapis.com/download.tensorflow.or
g/data/imagenet_class_index.json (https://storage.googleapis.com/download.te
nsorflow.org/data/imagenet_class_index.json)
35363/35363 [==============================] - 0s 2us/step
[[('n02504458', 'African_elephant', 0.8523125), ('n01871265', 'tusker', 0.13
733014), ('n02504013', 'Indian_elephant', 0.009101982), ('n01704323', 'trice
ratops', 0.001076614), ('n03743016', 'megalith', 0.00012015543)]]
```

In [3]:

```python
# A Python code that can load a set of deep learning neural
# network models, such as VGG16, ResNet50, MobileNet, and Inception V3,
# display the summary of the models, and use the model to classify an image.

from tensorflow.keras.preprocessing import image
from keras.applications.imagenet_utils import decode_predictions
import numpy as np
from tensorflow.keras.applications import (
 vgg16,
 resnet50,
 mobilenet,
 inception_v3
 )
# init the models
#model = vgg16.VGG16(weights='imagenet')
#model = resnet50.ResNet50(weights='imagenet')
model = mobilenet.MobileNet(weights='imagenet')
#model = inception_v3.InceptionV3(weights='imagenet')
print(model.summary())
img_path = 'Elephant.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
#processed_image = vgg16.preprocess_input(x)
#processed_image = resnet50.preprocess_input(x)
processed_image = mobilenet.preprocess_input(x)
#processed_image = inception_v3.preprocess_input(x)
# prediction
predictions = model.predict(x)
results = decode_predictions(predictions)
print(results)
```

```
 conv_pw_1_relu (ReLU)       (None, 112, 112, 64)      0

 conv_pad_2 (ZeroPadding2D)  (None, 113, 113, 64)      0

 conv_dw_2 (DepthwiseConv2D)  (None, 56, 56, 64)       576

 conv_dw_2_bn (BatchNormaliz  (None, 56, 56, 64)       256
 ation)

 conv_dw_2_relu (ReLU)       (None, 56, 56, 64)        0

 conv_pw_2 (Conv2D)          (None, 56, 56, 128)       8192

 conv_pw_2_bn (BatchNormaliz  (None, 56, 56, 128)      512
 ation)

 conv_pw_2_relu (ReLU)       (None, 56, 56, 128)       0

 conv_dw_3 (DepthwiseConv2D)  (None, 56, 56, 128)      1152
```

In [5]:

```
# A web camera, or webcam, is a useful tool to capture images to provide data
# resource for image classification using deep learning. A Python code for
# image classification based on the image array from a web
# camera (webcam), using VGG16 deep learning neural network

import cv2
from tensorflow.keras.preprocessing.image import img_to_array
from keras.applications.imagenet_utils import decode_predictions
from tensorflow.keras.applications import vgg16
import numpy as np
image_size = 224
model = vgg16.VGG16(weights='imagenet')
print(model.summary())
camera = cv2.VideoCapture(0)
while camera.isOpened():
    ok, cam_frame = camera.read()
    frame= cv2.resize(cam_frame, (image_size, image_size))
    numpy_image = img_to_array(frame)
    image_batch = np.expand_dims(numpy_image, axis=0)
    processed_image = vgg16.preprocess_input(image_batch.copy())
    # get the predicted probabilities for each class
    predictions = model.predict(processed_image)
    label = decode_predictions(predictions)
 # format final image visualization to display the results of experiments
    cv2.putText(cam_frame, "VGG16: {}, {:.1f}".format(label[0][0][1],
                                                      label[0][0][2]) , (10, 30), cv2.FONT_
    cv2.imshow('video image', cam_frame)
    key = cv2.waitKey(30)
    if key == 27: # press 'ESC' to quit
        break
camera.release()
cv2.destroyAllWindows()
```

```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_4 (InputLayer)        [(None, 224, 224, 3)]     0         
                                                                 
 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792      
                                                                 
 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928     
                                                                 
 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0         
                                                                 
 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856     
                                                                 
 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584    
                                                                 
 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0         
                                                                 
 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168    
```