[2]:

```python
# example of using a pre-trained model as a classifier
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from keras.applications.vgg16 import preprocess_input
from keras.applications.vgg16 import decode_predictions
from keras.applications.vgg16 import VGG16
# load an image from file
image = load_img('img.jpg', target_size=(224, 224))
# convert the image pixels to a numpy array
image = img_to_array(image)
# reshape data for the model
image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
# prepare the image for the VGG model
image = preprocess_input(image)
# load the model
model = VGG16()
# predict the probability across all output classes
yhat = model.predict(image)
# convert the probabilities to class labels
label = decode_predictions(yhat)
# retrieve the most likely result, e.g. highest probability
label = label[0][0]
# print the classification
print('%s (%.2f%%)' % (label[1], label[2]*100))
```

```
WARNING:tensorflow:From C:\Users\hp\anaconda3\Lib\site-
packages\keras\src\backend.py:1398: The name
tf.executing_eagerly_outside_functions is deprecated. Please use
tf.compat.v1.executing_eagerly_outside_functions instead.

WARNING:tensorflow:From C:\Users\hp\anaconda3\Lib\site-
packages\keras\src\layers\pooling\max_pooling2d.py:161: The name tf.nn.max_pool
is deprecated. Please use tf.nn.max_pool2d instead.

Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
374554624/553467096 [===================>…] - ETA: 7:14
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[2], line 16
     14 image = preprocess_input(image)
     15 # load the model
---> 16 model = VGG16()
     17 # predict the probability across all output classes
     18 yhat = model.predict(image)

File ~\anaconda3\Lib\site-packages\keras\src\applications\vgg16.py:235, in␣
 ↪VGG16(include_top, weights, input_tensor, input_shape, pooling, classes,␣
 ↪classifier_activation)
    233 if weights == "imagenet":
    234     if include_top:
--> 235         weights_path = data_utils.get_file(
    236             "vgg16_weights_tf_dim_ordering_tf_kernels.h5",
    237             WEIGHTS_PATH,
    238             cache_subdir="models",
    239             file_hash="64373286793e3c8b2b4e3219cbf3544b",
    240         )
    241     else:
    242         weights_path = data_utils.get_file(
    243             "vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5",
    244             WEIGHTS_PATH_NO_TOP,
    245             cache_subdir="models",
    246             file_hash="6d6bbae143d832006294945121d1f1fc",
    247         )

File ~\anaconda3\Lib\site-packages\keras\src\utils\data_utils.py:362, in␣
 ↪get_file(fname, origin, untar, md5_hash, file_hash, cache_subdir,␣
 ↪hash_algorithm, extract, archive_format, cache_dir)
    360     if os.path.exists(fpath) and file_hash is not None:
    361         if not validate_file(fpath, file_hash, algorithm=hash_algorithm):
--> 362             raise ValueError(
    363                 "Incomplete or corrupted file detected. "
    364                 f"The {hash_algorithm} "
    365                 "file hash does not match the provided value "
    366                 f"of {file_hash}."
    367             )
    369 if untar:
    370     if not os.path.exists(untar_fpath):

ValueError: Incomplete or corrupted file detected. The auto file hash does not␣
 ↪match the provided value of 64373286793e3c8b2b4e3219cbf3544b.
```

[5]:

```
# load an image from file
image = load_img('download2.png', target_size=(224, 224))
# convert the image pixels to a numpy array
image = img_to_array(image)
# reshape data for the model
image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
# prepare the image for the VGG model
image = preprocess_input(image)
# load the model
model = VGG16()
# predict the probability across all output classes
yhat = model.predict(image)
# convert the probabilities to class labels
label = decode_predictions(yhat)
# retrieve the most likely result, e.g. highest probability
label = label[0][0]
# print the classification
print('%s (%.2f%%)' % (label[1], label[2]*100))
```

```
valley (44.85%)
```

[1]:

```
from keras.applications.vgg16 import VGG16
```

```
WARNING:tensorflow:From C:\Users\hp\anaconda3\Lib\site-
packages\keras\src\losses.py:2976: The name
tf.losses.sparse_softmax_cross_entropy is deprecated. Please use
tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

[2]:

```
model = VGG16()
```

```
WARNING:tensorflow:From C:\Users\hp\anaconda3\Lib\site-
packages\keras\src\backend.py:1398: The name
tf.executing_eagerly_outside_functions is deprecated. Please use
tf.compat.v1.executing_eagerly_outside_functions instead.

WARNING:tensorflow:From C:\Users\hp\anaconda3\Lib\site-
packages\keras\src\layers\pooling\max_pooling2d.py:161: The name tf.nn.max_pool
is deprecated. Please use tf.nn.max_pool2d instead.

A local file was found, but it seems to be incomplete or outdated because the
auto file hash does not match the original value of
64373286793e3c8b2b4e3219cbf3544b so we will re-download the data.
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
553467096/553467096 [==============================] - 349s 1us/step
```

[3]:

```
model.summary()
```

```
Model: "vgg16"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (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

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

 fc1 (Dense)                 (None, 4096)              102764544

 fc2 (Dense)                 (None, 4096)              16781312
 predictions (Dense)         (None, 1000)              4097000

=================================================================
Total params: 138357544 (527.79 MB)
Trainable params: 138357544 (527.79 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[4]:

```python
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from keras.applications.vgg16 import preprocess_input
from keras.applications.vgg16 import decode_predictions
```

[5]:

```python
# load an image from file
image = load_img('img.jpg', target_size=(224, 224))
# convert the image pixels to a numpy array
image = img_to_array(image)
# reshape data for the model
image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
# prepare the image for the VGG model
image = preprocess_input(image)
```

[6]:

```python
image
```

[6]:

```
array([[[[-52.939003 , -57.779    , -65.68     ],
         [-49.939003 , -54.779    , -62.68     ],
         [-53.939003 , -58.779    , -66.68     ],
         …,
         [ 34.060997 , 111.221    ,  69.32     ],
         [ 28.060997 , 110.221    ,  59.32     ],
         [-62.939003 , -66.779    , -39.68     ]],

        [[-51.939003 , -56.779    , -64.68     ],
         [-49.939003 , -54.779    , -62.68     ],
         [-51.939003 , -56.779    , -64.68     ],
         …,
         [ 10.060997 , 103.221    ,  37.32     ],
         [ 28.060997 , 111.221    ,  67.32     ],
         [-46.939003 , -54.779    , -46.68     ]],

        [[-50.939003 , -55.779    , -63.68     ],
         [-48.939003 , -53.779    , -61.68     ],
         [-51.939003 , -56.779    , -64.68     ],
         …,
         [-10.939003 ,  61.221    ,  22.32     ],
         [ 38.060997 , 120.221    ,  63.32     ],
         [-54.939003 , -55.779    , -58.68     ]],
        …,

        [[ 76.061    ,  52.221    ,  37.32     ],
         [ 63.060997 ,  38.221    ,  27.32     ],
         [ 73.061    ,  49.221    ,  38.32     ],
         …,
         [ 18.060997 ,  11.221001 ,   3.3199997],
         [ 16.060997 ,   8.221001 ,   2.3199997],
         [-18.939003 , -26.779    , -24.68     ]],

        [[ 72.061    ,  48.221    ,  33.32     ],
         [ 67.061    ,  45.221    ,  29.32     ],
         [ 72.061    ,  48.221    ,  33.32     ],
         …,
         [ 17.060997 ,  10.221001 ,   2.3199997],
         [ 17.060997 ,   9.221001 ,   5.3199997],
         [-54.939003 , -64.779    , -57.68     ]],

        [[ 75.061    ,  51.221    ,  36.32     ],
         [ 70.061    ,  48.221    ,  31.32     ],
         [ 75.061    ,  52.221    ,  35.32     ],
         …,
         [ 16.060997 ,   9.221001 ,   1.3199997],
         [ 14.060997 ,   6.2210007,   2.3199997],
         [-66.939    , -76.779    , -69.68     ]]]], dtype=float32)
```

[7]:

```
yhat = model.predict(image)
# convert the probabilities to class labels
label = decode_predictions(yhat)
```

```
1/1 [==============================] - 1s 980ms/step
Downloading data from https://storage.googleapis.com/download.tensorflow.org/dat
a/imagenet_class_index.json
35363/35363 [==============================] - 0s 0us/step
```

[8]:

```
label
```

[8]:

```
[[('n02110063', 'malamute', 0.32372573),
  ('n02110185', 'Siberian_husky', 0.21747147),
  ('n02109961', 'Eskimo_dog', 0.1527094),
  ('n03218198', 'dogsled', 0.053158097),
  ('n02106166', 'Border_collie', 0.04220756)]]
```

[9]:

```
label = label[0][0]
# print the classification
print('%s (%.2f%%)' % (label[1], label[2]*100))
```

```
malamute (32.37%)
```

[ ]:

```
```