hy-23 / TensorFlow_Tutorials Private

Code Issues Pull requests Actions Projects Security Insights Settings

main

TensorFlow_Tutorials / TensorFlow_2.x / Chapter4_FeatureExtraction_VGG16.ipynb

hy23 Folder restructuring ... History

0 contributors

30.3 MB

Open in Colab

In [1]:
```
import tensorflow as tf
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras import models
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import preprocess_input
from matplotlib import pyplot as plt
from datetime import datetime
import numpy as np
import cv2

%load_ext tensorboard
```

In [2]:
```
basemodel = VGG16(weights='imagenet', include_top=True)
print(basemodel)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf
_dim_ordering_tf_kernels.h5
553467904/553467096 [==============================] - 6s 0us/step
553476096/553467096 [==============================] - 6s 0us/step
<keras.engine.functional.Functional object at 0x7fef1e4b5350>
```

In [3]:
```
for i, layer in enumerate(basemodel.layers):
  print(i, layer.name, layer.output_shape)
```

```
0 input_1 [(None, 224, 224, 3)]
1 block1_conv1 (None, 224, 224, 64)
2 block1_conv2 (None, 224, 224, 64)
3 block1_pool (None, 112, 112, 64)
4 block2_conv1 (None, 112, 112, 128)
5 block2_conv2 (None, 112, 112, 128)
6 block2_pool (None, 56, 56, 128)
7 block3_conv1 (None, 56, 56, 256)
8 block3_conv2 (None, 56, 56, 256)
9 block3_conv3 (None, 56, 56, 256)
10 block3_pool (None, 28, 28, 256)
11 block4_conv1 (None, 28, 28, 512)
12 block4_conv2 (None, 28, 28, 512)
13 block4_conv3 (None, 28, 28, 512)
14 block4_pool (None, 14, 14, 512)
15 block5_conv1 (None, 14, 14, 512)
16 block5_conv2 (None, 14, 14, 512)
17 block5_conv3 (None, 14, 14, 512)
18 block5_pool (None, 7, 7, 512)
19 flatten (None, 25088)
20 fc1 (None, 4096)
21 fc2 (None, 4096)
22 predictions (None, 1000)
```

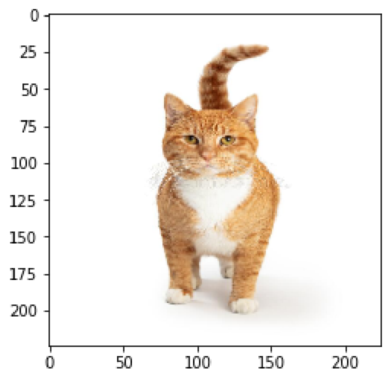
In [4]:
```
# extract features from "block4_pool" block
model = models.Model(inputs=basemodel.input, outputs=basemodel.get_layer('block4_pool').output)
```

In [5]:
```
model.summary()
# seems like everything until block4_pool
```

```
Model: "model"
_________________________________________________________________
 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

=================================================================
Total params: 7,635,264
Trainable params: 7,635,264
Non-trainable params: 0
_________________________________________________________________
```

In [6]:

```
img_path = '/content/drive/MyDrive/ResourceFiles/cat.jpg'
img = image.load_img(img_path, target_size=(224,224))
plt.imshow(img)
plt.show()
```



In [7]:

```
x = image.img_to_array(img)
x = np.expand_dims(x, axis = 0)

# use of preprocess_input ?
x = preprocess_input(x)

# get the features of this block
features = model.predict(x)
features.shape
```

Out[7]:

```
(1, 14, 14, 512)
```

In [13]:

```
# How to use TensorBoard?
# https://www.tensorflow.org/tensorboard/get_started
# https://www.tensorflow.org/tensorboard/image_summaries

# Clear out any prior log data.
!rm -rf logs

# Sets up a timestamped log directory.
logdir = "logs/data/" + datetime.now().strftime("%Y%m%d-%H%M%S")

# Creates a file writer for the log directory.
file_writer = tf.summary.create_file_writer(logdir)

# Using the file writer, log the reshaped image.
with file_writer.as_default():
  tf.summary.image("Training data", x, step=0)

%tensorboard --logdir logs/data
```

```
Reusing TensorBoard on port 6006 (pid 300), started 0:01:11 ago. (Use '!kill 300' to kill it.)
```

Auxiliary

In [14]:

```
print("shape of features is {}".format(features.shape))

#https://www.tensorflow.org/tensorboard/image_summaries#visualizing_a_single_image
```

```
y = np.transpose(features, (3, 1, 2, 0))
print("Requirement of y's shape as per link is {}".format(y.shape))
```

```
shape of features is (1, 14, 14, 512)
Requirement of y's shape as per link is (512, 14, 14, 1)
```

Take care.

'y' needs to be np.array, at least as per the link, https://www.tensorflow.org/tensorboard/image_summaries

You can also try feeding list of np.arrays and give the base address of list. Refer the next tensorboard, after this

In [15]:

```
with file_writer.as_default():
  # Don't forget to reshape.

  tf.summary.image("512 training data examples", y, max_outputs=6, step=0)

%tensorboard --logdir logs/data
```

```
Reusing TensorBoard on port 6006 (pid 300), started 0:01:41 ago. (Use '!kill 300' to kill it.)
```

In [11]:

```
list_images = []
for i in range(features.shape[3]):
  f = features[:,:,:,i]
  f = np.reshape(f, (14, 14, 1))
  list_images.append(f)
```

In [18]:

```
with file_writer.as_default():
  # Don't forget to reshape.

  # 'y' needs to be np.array, at least as per the link,
  # https://www.tensorflow.org/tensorboard/image_summaries

  # you can also try feeding list of np.arrays
  # and give the base address of list.
  tf.summary.image("Exp 512 training data examples", list_images, max_outputs=6, step=0)

%tensorboard --logdir logs/data
```

```
Reusing TensorBoard on port 6006 (pid 300), started 0:02:32 ago. (Use '!kill 300' to kill it.)
```