1、自己选择5类，每类100种的图片，上传到github或其他能够被下载的地方；

图片来自http://www.robots.ox.ac.uk/~vgg/data/pets/ (http://www.robots.ox.ac.uk/~vgg/data/pets/)

是一组宠物的图片，原数据集有37个分类，每类有约200张图片。命名规则为classname_数字序号.jpg

这里去前5个分类，每类取100张图片

{1: 'Abyssinian', 2: 'Bengal', 3: 'Birman', 4: 'Bombay', 5: 'British'}

上传的GitHub地址：
https://github.com/mayi140611/LSCJcourses/blob/master/dl4cv/lesson7/littleCBIRdatasets.npz (https://github.com/mayi140611/LSCJcourses/blob/master/dl4cv/lesson7/littleCBIRdatasets.npz)

In [ ]:

```
import os
import cv2
import math
import numpy as np
```

In [ ]:

```
def get_files(trainDir='D:/Desktop/pets', ratio=0.7):
    topdir = trainDir
    imgpathlist = list()
    labellist = list()
    labeldict = dict()
    num = 0
    count = 0
    for dirpath, dirs, files in os.walk(topdir):
        for f in sorted(files):
            # print(os.path.join(dirpath, f))
            labelname = f.split('_')[0]
            if labelname not in labeldict.values():
                if num > 4: break
                labeldict[num] = labelname
                num += 1
                count = 0
            if count < 100:
                imgpathlist.append(os.path.join(dirpath, f))
                labellist.append(num)
                count += 1

    print(len(labellist), len(imgpathlist), labeldict)
    seq = list(range(len(imgpathlist)))
    np.random.shuffle(seq)
    n_train = math.floor(len(imgpathlist)*ratio)
    n_val = len(imgpathlist) - n_train
    # print(seq[:5])
    train_images = list()
    val_images = list()
    train_labels = list()
    val_labels = list()
    for t in seq[:n_train]:
        i = imgpathlist[0]
        img = cv2.imread(i, cv2.IMREAD_GRAYSCALE)
        train_images.append(img)
        train_labels.append(labellist[t])
    for t in seq[n_train:]:
        i = imgpathlist[t]
        img = cv2.imread(i, cv2.IMREAD_GRAYSCALE)
        val_images.append(img)
        val_labels.append(labellist[t])
    return np.array(train_images), np.array(train_labels), np.array(val_images), np
```

In [ ]:

```
def get_files(trainDir='D:/Desktop/pets', ratio=0.3):
    topdir = trainDir
    imgpathlist = list()
    labellist = list()
    labeldict = dict()
    num = 0
    count = 0
    for dirpath, dirs, files in os.walk(topdir):
        for f in sorted(files):
            # print(os.path.join(dirpath, f))
            labelname = f.split('_')[0]
            if labelname not in labeldict.values():
                if num > 4: break
                num += 1
                labeldict[num] = labelname
                count = 0
            if count < 100:
                imgpathlist.append(os.path.join(dirpath, f))
                labellist.append(num)
                count += 1

    print(len(labellist), len(imgpathlist), labeldict)
    # seq = [1:len(imgpathlist)]
    seq = list(range(len(imgpathlist)))
    np.random.shuffle(seq)
    n_train = math.floor(len(imgpathlist)*ratio)
    n_val = len(imgpathlist) - n_train
    print(seq[:5])
    train_images = list()
    val_images = list()
    train_labels = list()
    val_labels = list()
    for t in seq[:n_train]:
        i = imgpathlist[t]
        img = cv2.imread(i, cv2.IMREAD_GRAYSCALE)
        # print(img.shape)
        #把图片的大小统一为512*512
        if img.shape[0] < 512:
            arr = np.zeros((512-img.shape[0], img.shape[1]))
            img = np.vstack((img, arr))
        elif img.shape[0] > 512:
            img = img[:512, :]
        if img.shape[1] < 512:
            arr = np.zeros((img.shape[0], 512-img.shape[1]))
            img = np.hstack((img, arr))
        elif img.shape[1] > 512:
            img = img[:, :512]
        train_images.append(img)
        train_labels.append(labellist[t])
    for t in seq[n_train:]:
        i = imgpathlist[t]
        img = cv2.imread(i, cv2.IMREAD_GRAYSCALE)
        # print(img.shape)
        #把图片的大小统一为512*512
        if img.shape[0] < 512:
            arr = np.zeros((512-img.shape[0], img.shape[1]))
            img = np.vstack((img, arr))
        elif img.shape[0] > 512:
            img = img[:512, :]
```

```
        if img.shape[1] < 512:

            arr = np.zeros((img.shape[0], 512-img.shape[1]))
            img = np.hstack((img, arr))
        elif img.shape[1] > 512:
            img = img[:, :512]
        val_images.append(img)
        val_labels.append(labellist[t])
    return np.array(train_images), np.array(train_labels), np.array(val_images), np
```

In [ ]:

```
X_train, y_train, X_test, y_test = get_files()
np.savez_compressed('littleCBIRdatasets.npz', X_train=X_train, y_train=y_train, X_t
```

2、修改本课代码，进行训练。看能否得到类似的结果；

In [2]:

```
"""
基础_模型迁移_CBIR_augmentation
by jsxyhelu
"""
#!apt-get -qq install -y graphviz && pip install -q pydot

import numpy as np
import cv2
import os
import math
import h5py as h5py

import pydot
import matplotlib.pyplot as plt
from keras.utils.vis_utils import plot_model
from IPython.display import Image

from keras.utils.data_utils import get_file
from keras.models import *
from keras.layers import *
from keras.applications.vgg16 import VGG16
from keras.optimizers import SGD
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing import image
```

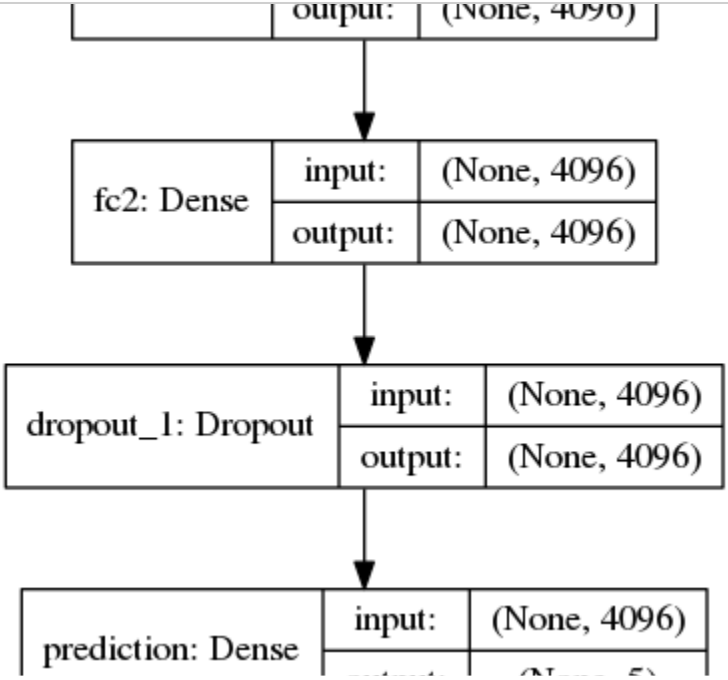
In [13]:

```
#训练集和验证集比率
RATIO = 0.2
#根据分类总数确定one-hot总类
NUM_DENSE = 5
#训练总数
epochs = 10
#默认图片大小:512*512
ishape=48
#one hot TODO给改掉
def tran_y(y):
    y_ohe = np.zeros(NUM_DENSE)
    y_ohe[y] = 1
    return y_ohe
```

In [4]:

```
#导入vgg模型
model_vgg = VGG16(include_top = False, weights = 'imagenet', input_shape = (ishape,
#将fc层失活，并且重新迁移训练
for layer in model_vgg.layers:
        layer.trainable = False
model = Flatten()(model_vgg.output)
model = Dense(4096, activation='relu', name='fc1')(model)
model = Dense(4096, activation='relu', name='fc2')(model)
model = Dropout(0.5)(model)
model = Dense(NUM_DENSE, activation = 'softmax', name='prediction')(model)
model_vgg_pretrain = Model(model_vgg.input, model, name = 'vgg16_pretrain')
#模型编译
sgd = SGD(lr = 0.05, decay = 1e-5)
model_vgg_pretrain.compile(loss = 'categorical_crossentropy', optimizer = sgd, metr

#模型结构打印
plot_model(model_vgg_pretrain, to_file="model.png", show_shapes=True)
Image('model.png')
```



In [40]:

```
#下载已经打包好的数据集,本例先验{1: 'Abyssinian', 2: 'Bengal', 3: 'Birman', 4: 'Bombay',
path='littleCBIRdatasets.npz'
f = np.load(path)
X_train, y_train = f['X_train'], f['y_train']
X_test, y_test = f['X_test'], f['y_test']
```

In [41]:

```
#下载的图片进行格式转换
X_train = [cv2.resize(i, (ishape, ishape)) for i in X_train]
X_train = [cv2.cvtColor(i, cv2.COLOR_GRAY2BGR) for i in X_train]
# X_train = X_train[:, :, np.newaxis]
X_train = np.concatenate([arr[np.newaxis] for arr in X_train]).astype('float32')
X_train /= 255.0
# print(X_train[:1])
X_test = [cv2.cvtColor(cv2.resize(i, (ishape, ishape)), cv2.COLOR_GRAY2BGR) for i i
# X_test = X_test[:, :, np.newaxis]
X_test = np.concatenate([arr[np.newaxis] for arr in X_test]).astype('float32')
X_test /= 255.0

y_train_ohe = np.array([tran_y(y_train[i]-1) for i in range(len(y_train))])
y_test_ohe = np.array([tran_y(y_test[i]-1) for i in range(len(y_test))])
y_train_ohe = y_train_ohe.astype('float32')
y_test_ohe = y_test_ohe.astype('float32')
```

In [42]:

```
#agumentation
# 设置生成参数
img_generator = ImageDataGenerator(
    rotation_range = 20,
    width_shift_range = 0.2,
    height_shift_range = 0.2,
    zoom_range = 0.2
    )

#模型训练
#log = model_vgg_pretrain.fit(X_train, y_train_ohe, validation_data = (X_test, y_te
img_generator.fit(X_train)
# fits the model_2 on batches with real-time data augmentation:
log = model_vgg_pretrain.fit_generator(img_generator.flow(X_train,y_train_ohe, batc
                    steps_per_epoch=len(X_train), epochs=epochs)
score = model_vgg_pretrain.evaluate(X_test, y_test_ohe, verbose=0)
#打印显示结果
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Epoch 1/10
350/350 [==============================] - 103s 294ms/step - loss: 1.
6191 - acc: 0.2007
Epoch 2/10
350/350 [==============================] - 103s 294ms/step - loss: 1.
6132 - acc: 0.2057
Epoch 3/10
350/350 [==============================] - 102s 293ms/step - loss: 1.
6107 - acc: 0.2047
Epoch 4/10
350/350 [==============================] - 104s 297ms/step - loss: 1.
6097 - acc: 0.2081
Epoch 5/10
350/350 [==============================] - 104s 298ms/step - loss: 1.
6094 - acc: 0.2105
Epoch 6/10
350/350 [==============================] - 103s 295ms/step - loss: 1.
6086 - acc: 0.2122
Epoch 7/10
350/350 [==============================] - 104s 299ms/step - loss: 1.
6083 - acc: 0.2098
Epoch 8/10
350/350 [==============================] - 104s 296ms/step - loss: 1.
6075 - acc: 0.2136
Epoch 9/10
350/350 [==============================] - 103s 295ms/step - loss: 1.
6075 - acc: 0.2115
Epoch 10/10
350/350 [==============================] - 104s 297ms/step - loss: 1.
6071 - acc: 0.2180
Test loss: 1.632811458905538
Test accuracy: 0.15333333412806194
```

In [52]:

```
log.history
```

Out[52]:

```
{'loss': [1.618825703608399,
  1.612831182773505,
  1.6106433109803633,
  1.6093502351045843,
  1.6089499529406635,
  1.6084347085718012,
  1.608274695561407,
  1.6076055709947363,
  1.6074840914470092,
  1.6071617506081188],
 'acc': [0.20143920163997284,
  0.20667711037807626,
  0.20510934634453656,
  0.21025063687235565,
  0.21152339978878493,
  0.21334706337784007,
  0.21044644654535352,
  0.21377521096443344,
  0.21226831464553753,
  0.21769140445429774]}
```

In [58]:

```
plt.figure('acc')
plt.subplot(2, 1, 1)
plt.plot(range(epochs), log.history['acc'],'r--',label='Training Accuracy')
plt.subplot(2, 1, 2)
plt.plot(range(epochs), log.history['loss'],'r--',label='Training Accuracy')
# plt.plot(log.history['val_acc'],'r-',label='Validation Accuracy')
# plt.legend(loc='best')
# plt.xlabel('Epochs')
# plt.axis([0, epochs, 0.9, 1])
# plt.figure('loss')
# plt.plot(log.history['loss'],'b--',label='Training Loss')
# # plt.plot(log.history['val_loss'],'b-',label='Validation Loss')
# plt.legend(loc='best')
# plt.xlabel('Epochs')
# plt.axis([0, epochs, 0, 1])
```

Out[58]:

```
[<matplotlib.lines.Line2D at 0x7fd7482e5e48>]
```