In [ ]:

```
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.models import Model
from tensorflow.keras.applications.vgg16 import decode_predictions
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import Input, Flatten, Dense

from PIL import Image
import numpy as np
import pandas as pd
import os
import warnings
```

In [ ]:

```
path = '/Users/utkarsh/Desktop/study/iitj/sem2/ml2/assignment2/101_ObjectCate
valid_exts = [".jpg", ".gif", ".png", ".jpeg"]
print ("[%d] CATEGORIES ARE IN \n %s" % (len(os.listdir(path)), path))
```

```
[10] CATEGORIES ARE IN
 /Users/utkarsh/Desktop/study/iitj/sem2/ml2/assignment2/101_ObjectCategories
```

In [ ]:

```
categories = sorted(os.listdir(path))
ncategories = len(categories)
imgs = []
labels = []
cat_dict = {}
# LOAD ALL IMAGES
for i, category in enumerate(categories):
    iter = 0
    for f in os.listdir(path + "/" + category):
        if iter == 0:
            ext = os.path.splitext(f)[1]
            if ext.lower() not in valid_exts:
                continue
            fullpath = os.path.join(path + "/" + category, f)
            img = load_img(fullpath, target_size=(200, 300))
            img = img_to_array(img)
            img = img.reshape((1, img.shape[0], img.shape[1], img.shape[2]))
            imgs.append(img) # NORMALIZE IMAGE
            label_curr = i
            labels.append(label_curr)
            cat_dict[category] = i
        #iter = (iter+1)%10;
print ("Num imgs: %d" % (len(imgs)))
print ("Num labels: %d" % (len(labels)) )
print (ncategories)
print(cat_dict)
```

```
Num imgs: 60
Num labels: 60
10
{'accordion': 0, 'airplanes': 1, 'anchor': 2, 'ant': 3, 'bass': 4, 'beaver':
5, 'binocular': 6, 'bonsai': 7, 'buddha': 8, 'butterfly': 9}
```

In [ ]:

```
seed = 7
np.random.seed(seed)
X_train, X_test, y_train, y_test = train_test_split(imgs, labels, test_size =
```

In [ ]:

```
model_vgg16_conv = VGG16(weights='imagenet', include_top=False)
#model_vgg16_conv.summary()

#Create your own input format (here 3x200x200)
ip = Input(shape=(200,300,3),name = 'image_input')

#Use the generated model
output_vgg16_conv = model_vgg16_conv(ip)

#Add the fully-connected layers
x = Flatten(name='flatten')(output_vgg16_conv)
x = Dense(4096, activation='relu', name='fc1')(x)
x = Dense(4096, activation='relu', name='fc2')(x)
x = Dense(1000, activation='softmax', name='predictions')(x)

#Create your own model
my_model = Model(inputs=ip, outputs=x)

#In the summary, weights and layers from VGG part will be hidden, but they wi
my_model.summary()
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
image_input (InputLayer)     [(None, 200, 300, 3)]     0         
_________________________________________________________________
vgg16 (Model)                multiple                  14714688  
_________________________________________________________________
flatten (Flatten)            (None, 27648)             0         
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              113250304 
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312  
_________________________________________________________________
predictions (Dense)          (None, 1000)              4097000   
=================================================================
Total params: 148,843,304
Trainable params: 148,843,304
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:

```

temp_labels = []
temp_features = []

for index in range(0, len(X_train)):
    img = X_train[index]
    img = preprocess_input(img)
    # Extracting features
    yhat = my_model.predict(img)
    # 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))
```

```
hard_disc (78.46%)
kuvasz (29.98%)
mask (99.20%)
hard_disc (100.00%)
gyromitra (88.94%)
hard_disc (99.95%)
hard_disc (84.33%)
hard_disc (100.00%)
hard_disc (93.08%)
hard_disc (95.52%)
hard_disc (94.11%)
muzzle (48.69%)
envelope (99.36%)
mask (87.51%)
hard_disc (93.04%)
gyromitra (57.80%)
golfcart (65.01%)
mask (61.02%)
hard_disc (99.94%)
jinrikisha (28.36%)
hard_disc (85.03%)
piggy_bank (79.35%)
gyromitra (99.83%)
hard_disc (96.63%)
spider_monkey (74.97%)
ballpoint (53.75%)
mask (97.96%)
mask (93.35%)
mask (99.99%)
mask (85.09%)
hard_disc (99.86%)
mask (84.28%)
hard_disc (99.98%)
radiator (38.21%)
cougar (52.87%)
Norwich_terrier (99.58%)
hard_disc (94.16%)
book_jacket (34.27%)
hard_disc (62.65%)
gyromitra (97.09%)
miniature_schnauzer (37.69%)
mask (77.01%)
hard_disc (95.61%)
hard_disc (99.99%)
academic_gown (98.97%)
hard_disc (43.63%)
hard_disc (75.06%)
dugong (80.62%)
```