一 專題摘要

浣熊與袋鼠辨識模型

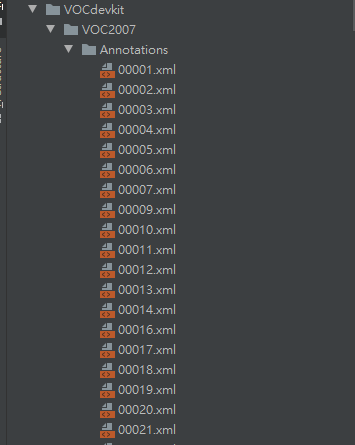
本次期末專題主題建立一個模型同時能辨識出浣(raccoon)

與袋鼠(kangaroo)的類別與位置

二實作方法介紹

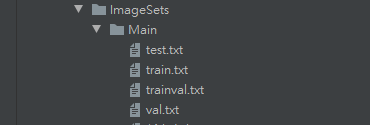
Yolov3\_karas

首先利用lableImg工具標注自己的圖片，生成對應的xml檔，具體標注方法這裡不再贅述，然後按照以下的格式創建VOC資料集, 其中，把剛才標注好的JPG圖片放入JPEGImages資料夾中，將xml標注文件放在Annotations資料夾中



接下來要在Main中生成r個txt檔，其中test.txt是測試集，train.txt是訓練集，val.txt是驗證集，trainval.txt是訓練和驗證集。為了操作方便，我們直接在VOC2007下新建一個name.py的檔，內容如下

import os  
import random  
import sys  
'''  
if len(sys.argv) < 2:  
 print("no directory specified, please input target directory")  
 exit()  
'''  
#root\_path = sys.argv[1]  
root\_path = "C:\\Users\\reyna\\Yolo\\VOCdevkit\\VOC2007"  
  
  
xmlfilepath = root\_path + '[\\Annotations](file://\\Annotations\)'  
  
txtsavepath = root\_path + '[\\ImageSets\\Main](file://\\ImageSets\\Main)'  
  
if not os.path.exists(root\_path):  
 print("cannot find such directory: " + root\_path)  
 exit()  
  
if not os.path.exists(txtsavepath):  
 os.makedirs(txtsavepath)  
  
trainval\_percent = 0.9  
train\_percent = 0.8  
total\_xml = os.listdir(xmlfilepath)  
num = len(total\_xml)  
list = range(num)  
tv = int(num \* trainval\_percent)  
tr = int(tv \* train\_percent)  
trainval = random.sample(list, tv)  
train = random.sample(trainval, tr)  
  
print("train and val size:", tv)  
print("train size:", tr)  
  
ftrainval = open(txtsavepath + '/trainval.txt', 'w')  
ftest = open(txtsavepath + '/test.txt', 'w')  
ftrain = open(txtsavepath + '/train.txt', 'w')  
fval = open(txtsavepath + '/val.txt', 'w')  
  
for i in list:  
 name = total\_xml[i][:-4] + '\n'  
 if i in trainval:  
 ftrainval.write(name)  
 if i in train:  
 ftrain.write(name)  
 else:  
 fval.write(name)  
 else:  
 ftest.write(name)  
  
ftrainval.close()  
ftrain.close()  
fval.close()  
ftest.close()



接下來修改yolov3資料夾下的voc\_annotation.py檔，將classes修改為自己的類別，然後運行生成yolo格式的標注檔

python voc\_annotation.py

這樣在主目錄下生成以下三個檔

2007\_test.txt

2007\_train.txt

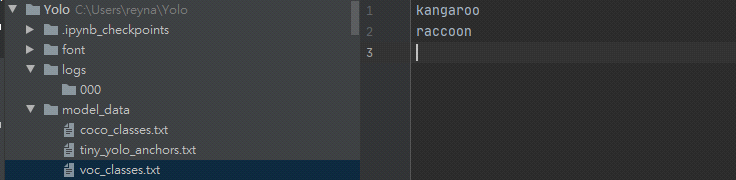
2007\_val.txt

yolo預訓練模型

* wget <https://pjreddie.com/media/files/yolov3.weights>
* python convert.py yolov3.cfg yolov3.weights model\_data/yolo.h5

**修改類別檔**

找到model\_data/voc\_classes.txt檔，將裡邊的類別修改為自己的類別



修改yolov3.cfg

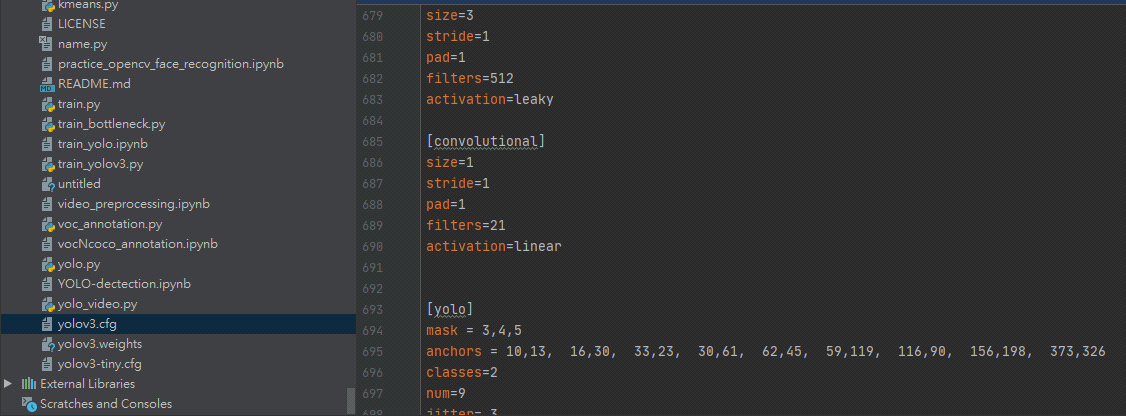
yolov3的filters計算方式:

filters=(classes+5)\*3

#(1+5)\*3=18

decay=0.005

[convolutional]  
size=1  
stride=1  
pad=1  
filters=21  
activation=linear  
  
  
[yolo]  
mask = 6,7,8  
anchors = 10,13, 16,30, 33,23, 30,61, 62,45, 59,119, 116,90, 156,198, 373,326  
classes=2  
num=9  
jitter=.3  
ignore\_thresh = .5  
truth\_thresh = 1  
random=0



開始訓練

import numpy as np

import keras.backend as K

from keras.layers import Input, Lambda

from keras.models import Model

from keras.optimizers import Adam

from keras.callbacks import TensorBoard, ModelCheckpoint, ReduceLROnPlateau, EarlyStopping

from train import get\_classes, get\_anchors, create\_model, create\_tiny\_model, data\_generator, data\_generator\_wrapper

from PIL import ImageFile

ImageFile.LOAD\_TRUNCATED\_IMAGES = True

annotation\_path = '2007\_train.txt' # 轉換好格式的標註檔案

log\_dir = 'logs/000/' # 訓練好的模型儲存的路徑

classes\_path = 'model\_data/voc\_classes.txt'

anchors\_path = 'model\_data/yolo\_anchors.txt'

class\_names = get\_classes(classes\_path)

num\_classes = len(class\_names)

anchors = get\_anchors(anchors\_path)

input\_shape = (416,416) # multiple of 32, hw

is\_tiny\_version = len(anchors)==6 # default setting

if is\_tiny\_version:

    model = create\_tiny\_model(input\_shape, anchors, num\_classes,

        freeze\_body=2, weights\_path='model\_data/tiny\_yolo\_weights.h5')

else:

    model = create\_model(input\_shape, anchors, num\_classes,

        freeze\_body=2, weights\_path='model\_data/yolo\_weights.h5') # make sure you know what you freeze

logging = TensorBoard(log\_dir=log\_dir)

checkpoint = ModelCheckpoint(log\_dir + 'ep{epoch:03d}-loss{loss:.3f}-val\_loss{val\_loss:.3f}.h5',

    monitor='val\_loss', save\_weights\_only=True, save\_best\_only=True, period=3)

reduce\_lr = ReduceLROnPlateau(monitor='val\_loss', factor=0.1, patience=3, verbose=1)

early\_stopping = EarlyStopping(monitor='val\_loss', min\_delta=0, patience=10, verbose=1)

# 分為 training 以及 validation

val\_split = 0.1

with open(annotation\_path) as f:

    lines = f.readlines()

np.random.seed(10101)

np.random.shuffle(lines)

np.random.seed(None)

num\_val = int(len(lines)\*val\_split)

num\_train = len(lines) - num\_val

# Train with frozen layers first, to get a stable loss.

# Adjust num epochs to your dataset. This step is enough to obtain a not bad model.

# 一開始先 freeze YOLO 除了 output layer 以外的 darknet53 backbone 來 train

if True:

    model.compile(optimizer=Adam(lr=1e-3), loss={

        # use custom yolo\_loss Lambda layer.

        'yolo\_loss': lambda y\_true, y\_pred: y\_pred})

    batch\_size = 16

    print('Train on {} samples, val on {} samples, with batch size {}.'.format(num\_train, num\_val, batch\_size))

    # 模型利用 generator 產生的資料做訓練，強烈建議大家去閱讀及理解 data\_generator\_wrapper 在 train.py 中的實現

    model.fit\_generator(data\_generator\_wrapper(lines[:num\_train], batch\_size, input\_shape, anchors, num\_classes),

            steps\_per\_epoch=max(1, num\_train//batch\_size),

            validation\_data=data\_generator\_wrapper(lines[num\_train:], batch\_size, input\_shape, anchors, num\_classes),

            validation\_steps=max(1, num\_val//batch\_size),

            epochs=50,

            initial\_epoch=0,

            callbacks=[logging, checkpoint])

    model.save\_weights(log\_dir + 'trained\_weights\_stage\_1.h5')

# Unfreeze and continue training, to fine-tune.

# Train longer if the result is not good.

if True:

    # 把所有 layer 都改為 trainable

    for i in range(len(model.layers)):

        model.layers[i].trainable = True

    model.compile(optimizer=Adam(lr=1e-4), loss={'yolo\_loss': lambda y\_true, y\_pred: y\_pred}) # recompile to apply the change

    print('Unfreeze all of the layers.')

    batch\_size = 16 # note that more GPU memory is required after unfreezing the body

    print('Train on {} samples, val on {} samples, with batch size {}.'.format(num\_train, num\_val, batch\_size))

    model.fit\_generator(data\_generator\_wrapper(lines[:num\_train], batch\_size, input\_shape, anchors, num\_classes),

        steps\_per\_epoch=max(1, num\_train//batch\_size),

        validation\_data=data\_generator\_wrapper(lines[num\_train:], batch\_size, input\_shape, anchors, num\_classes),

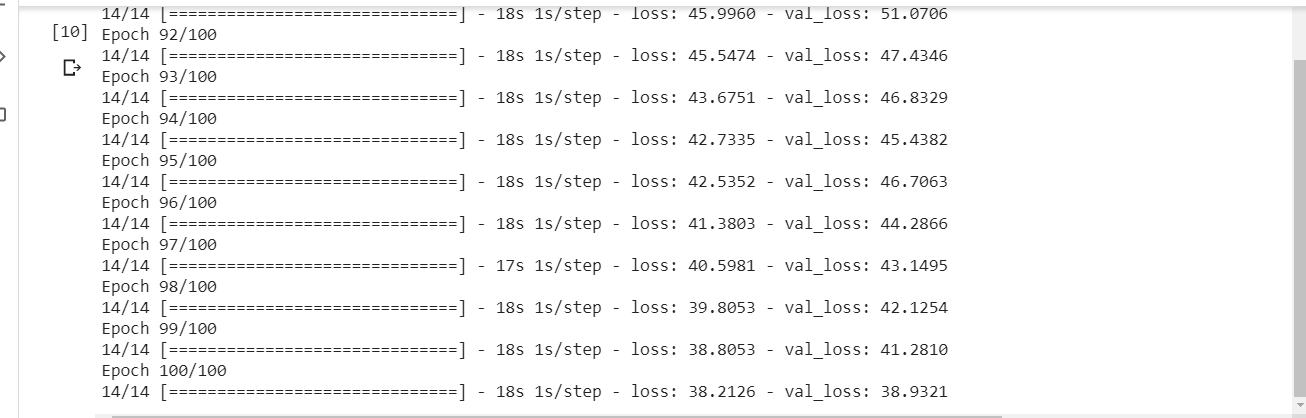
        validation\_steps=max(1, num\_val//batch\_size),

        epochs=100,

        initial\_epoch=50,

        callbacks=[logging, checkpoint, reduce\_lr, early\_stopping])

    model.save\_weights(log\_dir + 'trained\_weights\_final.h5')



1. YOLO會把圖先平均分成S×S格，這邊假設S = 5，圖會被平均分成5×5格(如上左圖)，每一格在英文被稱為grid cell (大小為20×20)。

整體的概念就是如果要「被偵測的物件中心」落在哪一個grid cell，那個grid cell就要負責偵測這個物件。

2. 每個grid cell必須要負責預測「B個bounding boxes」和「屬於每個類別的機率」，每個bounding box會帶有5個預設值(x, y, w, h, and confidence)

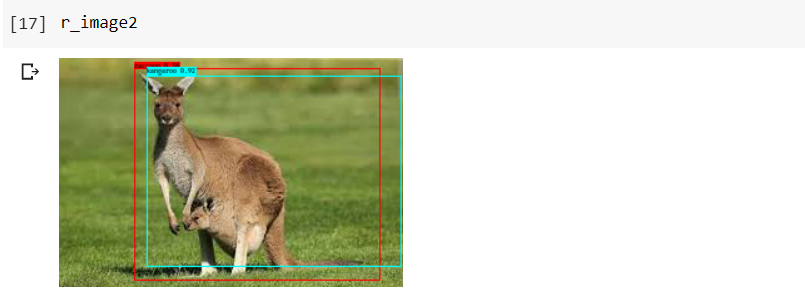
3. 作者舉PASCAL VOC在執行YOLO的例子，他設定S=7，B=2，PASCAL VOC有20個物件的類別，所以C=20。一開始有說YOLO最後的tensor為S×S×(B×5+C)。

Bounding Box正規化

Activation function

三、成果展示





四、結論