Hw1

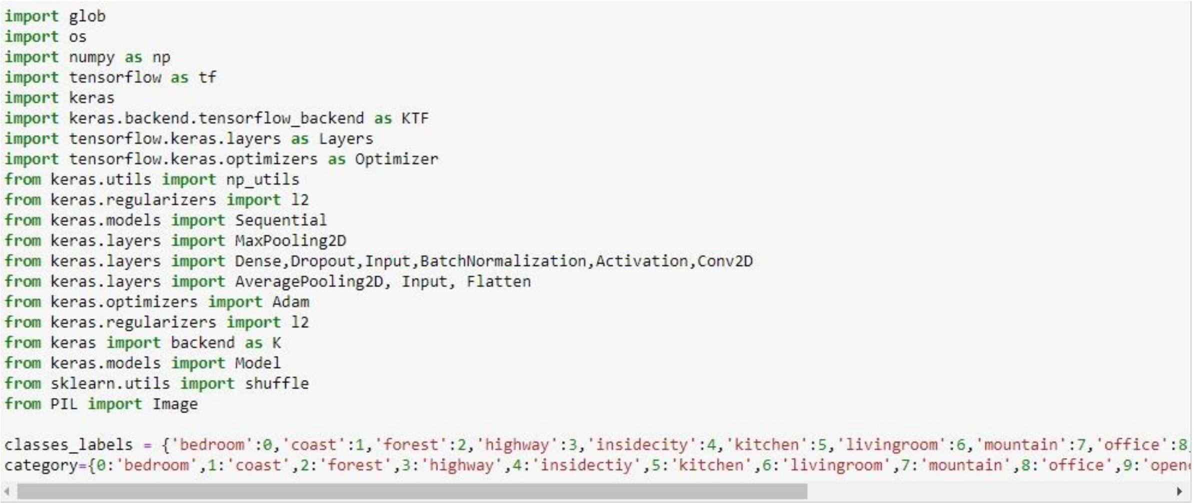
0856703 黃威竣

GitHub link:

Introduction:

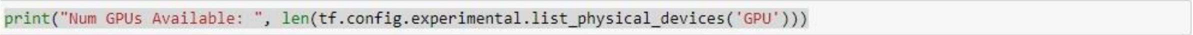
我使用的建置 cnn model 的函式庫是使用 kera，接著為了使用 GPU 去運算，使用了 tensorflow_gpu，雖然我自己使用的 GPU 很不夠力，不過還是稍微比 CPU 快 ，而主要是使用 anaconda 的 jupyter notebook 去寫的。

以下為函式庫以及 GPU 的部分，classes_labels 與 category 主要用來分類使用

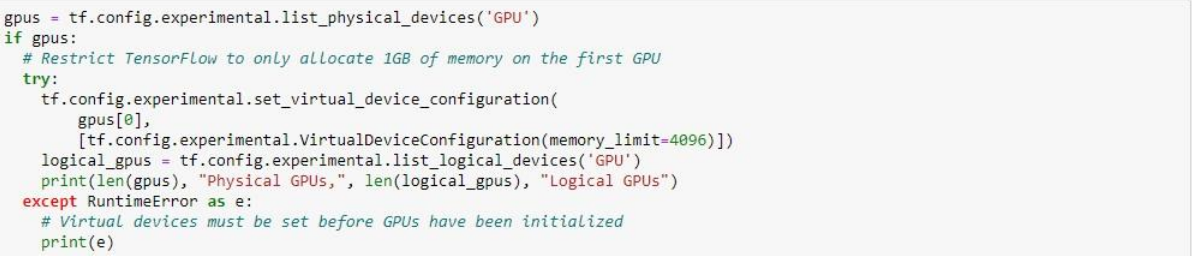


```
import glob
import os
import numpy as np
import tensorflow as tf
import keras
import keras.backend.tensorflow_backend as KTF
import tensorflow.keras.layers as Layers
import tensorflow.keras.optimizers as Optimizer
from keras.utils import np_utils
from keras.regularizers import l2
from keras.models import Sequential
from keras.layers import MaxPooling2D
from keras.layers import Dense,Dropout,Input,BatchNormalization,Activation,Conv2D
from keras.layers import AveragePooling2D, Input, Flatten
from keras.optimizers import Adam
from keras.regularizers import l2
from keras import backend as K
from keras.models import Model
from sklearn.utils import shuffle
from PIL import Image

classes_labels = {'bedroom':0,'coast':1,'forest':2,'highway':3,'insidecity':4,'kitchen':5,'livingroom':6,'mountain':7,'office':8
category={0:'bedroom',1:'coast',2:'forest',3:'highway',4:'insidectiy',5:'kitchen',6:'livingroom',7:'mountain',8:'office',9:'open
```

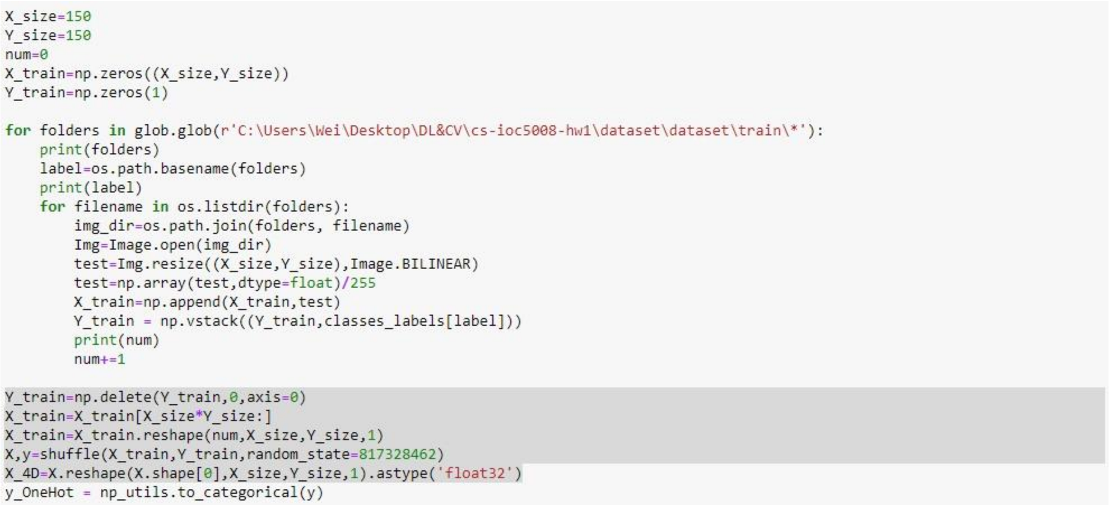


```
print("Num GPUs Available: ", len(tf.config.experimental.list_physical_devices('GPU')))
```



```
gpus = tf.config.experimental.list_physical_devices('GPU')
if gpus:
  # Restrict TensorFlow to only allocate 1GB of memory on the first GPU
  try:
    tf.config.experimental.set_virtual_device_configuration(
        gpus[0],
        [tf.config.experimental.VirtualDeviceConfiguration(memory_limit=4096)])
    logical_gpus = tf.config.experimental.list_logical_devices('GPU')
    print(len(gpus), "Physical GPUs,", len(logical_gpus), "Logical GPUs")
  except RuntimeError as e:
    # Virtual devices must be set before GPUs have been initialized
    print(e)
```

以下為前置讀檔的部分，由路徑讀取測資，接著 size 取為 150*150，這是我訓練出來感覺準度比較好的大小，接著在做一些單純的 reshape 處理。



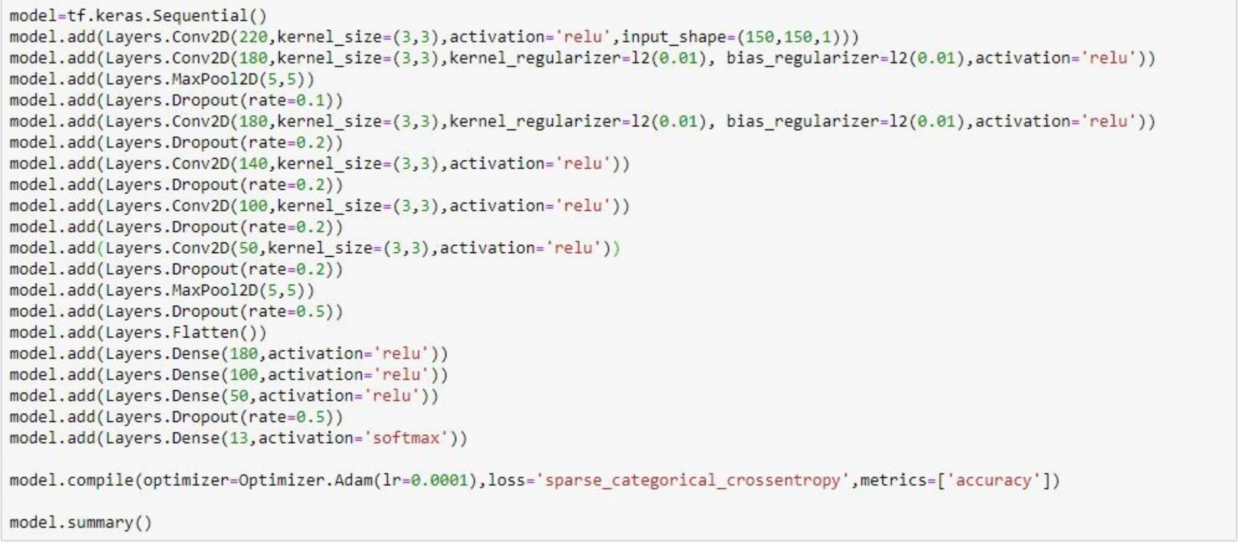
```
X_size=150
Y_size=150
num=0
X_train=np.zeros((X_size,Y_size))
Y_train=np.zeros(1)

for folders in glob.glob(r'C:\Users\Wei\Desktop\DL&CV\cs-ioc5008-hw1\dataset\dataset\train\*'):
    print(folders)
    label=os.path.basename(folders)
    print(label)
    for filename in os.listdir(folders):
        img_dir=os.path.join(folders, filename)
        Img=Image.open(img_dir)
        test=Img.resize((X_size,Y_size),Image.BILINEAR)
        test=np.array(test,dtype=float)/255
        X_train=np.append(X_train,test)
        Y_train = np.vstack((Y_train,classes_labels[label]))
        print(num)
        num+=1

Y_train=np.delete(Y_train,0,axis=0)
X_train=X_train[X_size*Y_size:]
X_train=X_train.reshape(num,X_size,Y_size,1)
X,y=shuffle(X_train,Y_train,random_state=817328462)
X_4D=X.reshape(X.shape[0],X_size,Y_size,1).astype('float32')
y_OneHot = np_utils.to_categorical(y)
```

以下為 model 的建置，我自己是從最簡單的 conv 層,pooling 層,dense 層慢慢往上加上去建置的，因為測試資料有點少，我在訓練的時候有參考過網路上其他比較強的 model，但是很容易 OOM，所以就以比較單純的建置方法下手，接著訓練後也發現說很容易 overfitting，所以會加入 regularizer 以及 dropout.

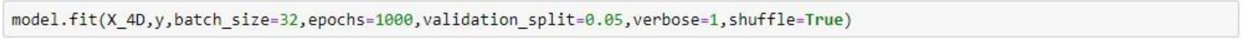
訓練了很久之後，交上去 kaggle 的準確度到達 69 分



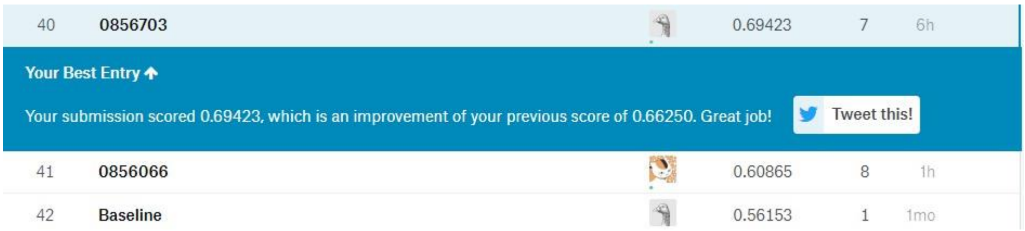
```
model=tf.keras.Sequential()
model.add(Layers.Conv2D(220,kernel_size=(3,3),activation='relu',input_shape=(150,150,1)))
model.add(Layers.Conv2D(180,kernel_size=(3,3),kernel_regularizer=l2(0.01), bias_regularizer=l2(0.01),activation='relu'))
model.add(Layers.MaxPool2D(5,5))
model.add(Layers.Dropout(rate=0.1))
model.add(Layers.Conv2D(180,kernel_size=(3,3),kernel_regularizer=l2(0.01), bias_regularizer=l2(0.01),activation='relu'))
model.add(Layers.Dropout(rate=0.2))
model.add(Layers.Conv2D(140,kernel_size=(3,3),activation='relu'))
model.add(Layers.Dropout(rate=0.2))
model.add(Layers.Conv2D(100,kernel_size=(3,3),activation='relu'))
model.add(Layers.Dropout(rate=0.2))
model.add(Layers.Conv2D(50,kernel_size=(3,3),activation='relu'))
model.add(Layers.Dropout(rate=0.2))
model.add(Layers.MaxPool2D(5,5))
model.add(Layers.Dropout(rate=0.5))
model.add(Layers.Flatten())
model.add(Layers.Dense(180,activation='relu'))
model.add(Layers.Dense(100,activation='relu'))
model.add(Layers.Dense(50,activation='relu'))
model.add(Layers.Dropout(rate=0.5))
model.add(Layers.Dense(13,activation='softmax'))

model.compile(optimizer=Optimizer.Adam(lr=0.0001),loss='sparse_categorical_crossentropy',metrics=['accuracy'])

model.summary()
```



```
model.fit(X_4D,y,batch_size=32,epochs=1000,validation_split=0.05,verbose=1,shuffle=True)
```



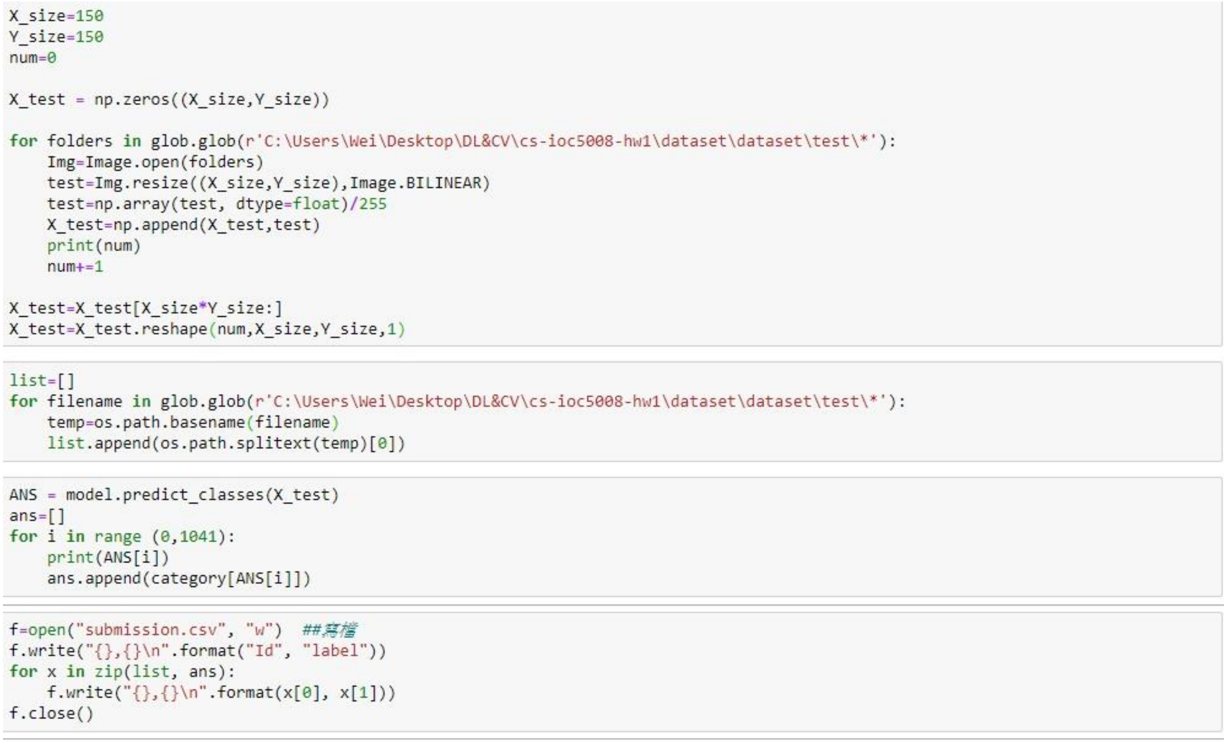
40	0856703	0.69423	7	6h

Your Best Entry ↑

Your submission scored 0.69423, which is an improvement of your previous score of 0.66250. Great job!

41	0856066	0.60865	8	1h
42	Baseline	0.56153	1	1mo

程式最後的部分就是匯入 test data 以及預測跟寫檔了~



```
X_size=150
Y_size=150
num=0

X_test = np.zeros((X_size,Y_size))

for folders in glob.glob(r'C:\Users\Wei\Desktop\DL&CV\cs-ioc5008-hw1\dataset\dataset\test\*'):
    Img=Image.open(folders)
    test=Img.resize((X_size,Y_size),Image.BILINEAR)
    test=np.array(test, dtype=float)/255
    X_test=np.append(X_test,test)
    print(num)
    num+=1

X_test=X_test[X_size*Y_size:]
X_test=X_test.reshape(num,X_size,Y_size,1)
```

```
list=[]
for filename in glob.glob(r'C:\Users\Wei\Desktop\DL&CV\cs-ioc5008-hw1\dataset\dataset\test\*'):
    temp=os.path.basename(filename)
    list.append(os.path.splitext(temp)[0])
```

```
ANS = model.predict_classes(X_test)
ans=[]
for i in range (0,1041):
    print(ANS[i])
    ans.append(category[ANS[i]])
```

```
f=open("submission.csv", "w")  ##寫檔
f.write("{},{}\n".format("Id", "label"))
for x in zip(list, ans):
    f.write("{},{}\n".format(x[0], x[1]))
f.close()
```

Summay

我覺得我自己在使用 model 所下的功夫不夠，而準確度只達到快 70%，跟其他人相比還差太多，而測資數量有點少，很容易 overfitting，我覺得我所使用的正規化跟 dropout 的方式有待加強，然後設計的時候，有時候參數太大也很容易 OOM,所以設計起來要很有 sense，能夠好好的去做 data preprocess 以及 layer 的加入,該如何去處理 overfitting 的問題，所以要訓練到準度高達 95%以上看起來真的很有難度，我也很想見識看看前幾名的做法，想看看其他人的做法。