HW1-ARTCNN-109403021 說明文件

姓名學號系級:資管二 A_109403021_傅珩洵

Colab 連結:連結

執行過程:

1.讀入封包

```
[ ] import numpy as np
    import tensorflow as tf
    from tensorflow import keras
    from tensorflow.keras import layers
    import cv2 as cv
    import matplotlib.pyplot as plt
    import pandas as pd
    import seaborn as sns
    import os
    import random
    from tensorflow.keras.layers import Conv2D, MaxPooling2D
    from keras.layers import Dense, Activation, Flatten, Dropout
    from sklearn.utils import class_weight
    from tensorflow.keras.optimizers import Adam
    from tensorflow.keras.layers import BatchNormalization
    from keras.layers import GlobalAveragePooling2D
```

下載完資料集後把我這次要用到的套件都 import 進來

2.取得資料集

```
[ ] artists = artists.loc[:,["name","paintings"]]
    artists["name"] = artists["name"].str.split(" ").apply(lambda parts: "_".join(parts))
    artists['class_weight'] = artists.paintings.sum() / (artists.shape[0] * artists.paintin

    class_weight = artists['class_weight'].to_dict()
    artists.head()
```

	name	paintings	class_weight
0	Amedeo_Modigliani	193	0.875233
1	Vasiliy_Kandinskiy	88	1.919545
2	Diego_Rivera	70	2.413143
3	Claude_Monet	73	2.313973
4	Rene_Magritte	194	0.870722

這邊我增加 class_weight 欄位以嘗試解決資料量的不平衡，paintings 數量越多其比重就會較低

3.資料前處理

```
[ ]  #  請建立將英文映射成數字的dict。EX:  Van_Gogh  -->  0

     author_Dict  =  dict()

     def  make_Authordict():
         index  =  0
         for  i  in  artists["name"]:
             author_Dict[i]  =  index
             index  =  index+1
         return  author_Dict

     class_name  =  make_Authordict()

     #  請建立將數字映射成英文的dict。  EX:  0  -->  Van_Gogh
     rev_class_name  =  {v:  k  for  k,  v  in  author_Dict.items()}
```

```
[ ]  def  get_label(picName):
         #  請取出label並轉成數字
         #  EX:  Claude_Monet_1.jpg  ->  Claude_Monet  ->  1
         baseline  =  "_"
         picName  =  picName.rpartition(baseline)[0]
         return  author_Dict[picName]

     def  get_path(dir,picName):
         #  請將路徑合併
         #  EX:  ./train_resized/  +  Claude_Monet_1.jpg  =>  ./train_resized/Claude_Monet_1.jpg
         path  =  dir  +  picName
         return  path

     def  make_paths_label(dir):
         img_list  =  os.listdir(dir)
         paths  =  []
         labels  =  []

         #  將preprocess完成的path、label用for迴圈放入paths和labels
         for  img  in  img_list:
             labels.append(get_label(img))
             paths.append(get_path(dir,img))
             onehot_labels  =  np.eye(50)[labels]
         return  paths,onehot_labels
         #  將labels轉成onehot
```

按照註解的說明將畫家名字映射成英文然後轉為 one_hot 形式成功輸出如下

```
paths :
./train_resized/Paul_Gauguin_139.jpg
./train_resized/Francisco_Goya_135.jpg
./train_resized/Pierre-Auguste_Renoir_202.jpg
./train_resized/Paul_Klee_147.jpg
./train_resized/Marc_Chagall_9.jpg
--------------------
labels :
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.
 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.
 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.
 0. 0.]
```

```
#  決定你輸入模型的圖片長寬
#  shuffle  buffle  size
IMG_WIDTH  =  128
IMG_HEIGHT  =  128
IMG_SIZE  =  ((IMG_WIDTH),(IMG_HEIGHT))
shuffle_buffer  =  2000
```

這裡我設定了圖片的長寬和 shuffle_buffer，其中長寬影響訓練結果頗深，多次嘗試後我定在 128*128

```
[ ]  #  切割成training  data與validation  data
     train_len  =  int(0.8*total_len)
     val_len  =  total_len  -  train_len

     train_ds  =  full_ds.take(train_len)
     val_ds  =  full_ds.skip(train_len)

     print("train  size  :  ",train_len,"  val  size  :  ",val_len)

     #  添加batch
     #  todo
     batch_size  =  64

     train_ds  =  train_ds.batch(batch_size)
     val_ds  =  val_ds.batch(batch_size)
```

```
train size :  6016  val size :  1504
```

```
[ ]  #  查看添加batch後的維度
     trainiter  =  iter(train_ds)
     x,y  =  trainiter.next()
     print("training  image  batch  shape  :  ",x.shape)
     print("training  label  batch  shape  :  ",y.shape)
```

```
training image batch shape :  (64, 128, 128, 3)
training label batch shape :  (64, 50)
```

這裡 batch_size 我設在 64

4.建立模型

```
[ ]  input_shape  =  (IMG_WIDTH, IMG_HEIGHT, 3)

     #  自訂你的model
     model  =  keras.Sequential(
               [
                 keras.Input(shape=input_shape),

                 Conv2D(32,  kernel_size=(3,  3)),
                 BatchNormalization(),
                 Activation('relu'),
                 MaxPooling2D(pool_size=(2,  2),strides=(2,2)),


                 Conv2D(64,  kernel_size=(3,  3)),
                 BatchNormalization(),
                 Activation('relu'),
                 MaxPooling2D(pool_size=(2,  2),strides=(2,2)),

                 Conv2D(128,  kernel_size=(3,  3)),
                 BatchNormalization(),
                 Activation('relu'),
                 MaxPooling2D(pool_size=(2,  2),strides=(2,2)),

                 Conv2D(256,  kernel_size=(3,  3)),
                 BatchNormalization(),
                 Activation('relu'),
                 MaxPooling2D(pool_size=(2,  2),strides=(2,2)),

                 Conv2D(512,  kernel_size=(3,  3)),
                 BatchNormalization(),
                 Activation('relu'),
                 MaxPooling2D(pool_size=(2,  2),strides=(2,2)),
```

```
                 GlobalAveragePooling2D(),
                 #Flatten(),
                 Dense(512),
                 Activation('relu'),
                 Dropout(0.1),
                 Dense(num_classes),
                 Activation('softmax'),

               ]
     )

     #adan  =  Adam(lr=1e-4)

     model.summary()
```

花最多時間調整的模型，疊了五層的 convolution + maxpooling 後面我本來有用 flatten，但後來發現去除調會讓我的訓練更穩定些所以註解掉

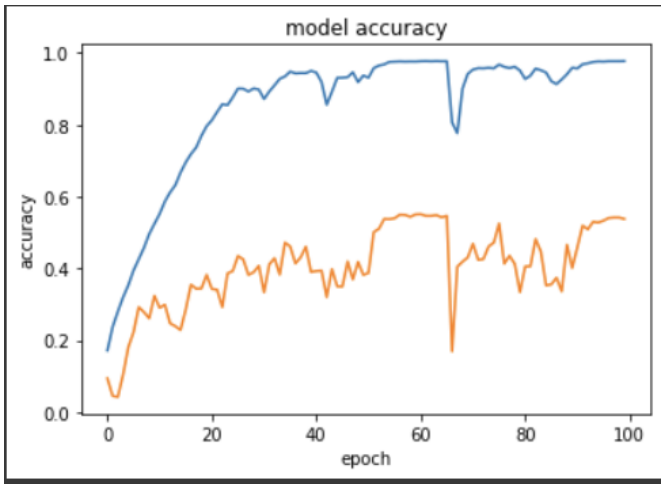
5.制定訓練計畫

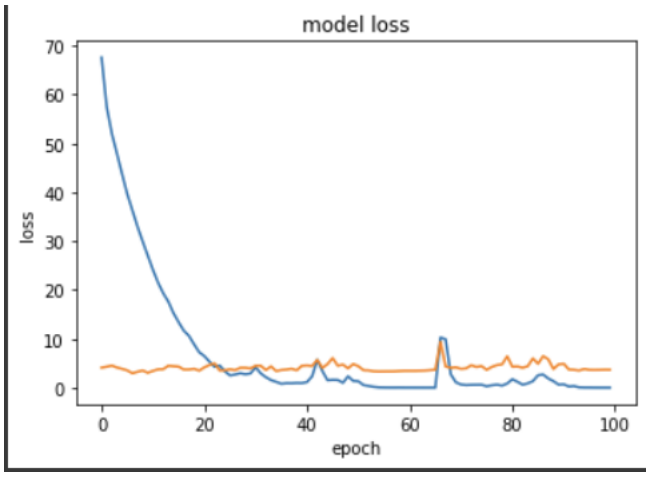
```
#  todo
epochs  =  100

model.compile(loss="categorical_crossentropy",  optimizer="adam",  metrics=["accuracy"])


history  =  model.fit(train_ds,  epochs=epochs,  class_weight=dict(enumerate(class_weight)),
```

```
loss: 0.0096 - accuracy: 0.9774 - val_loss: 3.6743 - val_accuracy: 0.5379
```

跑了 100 次的和第 100 次的輸出結果

6.評估模型



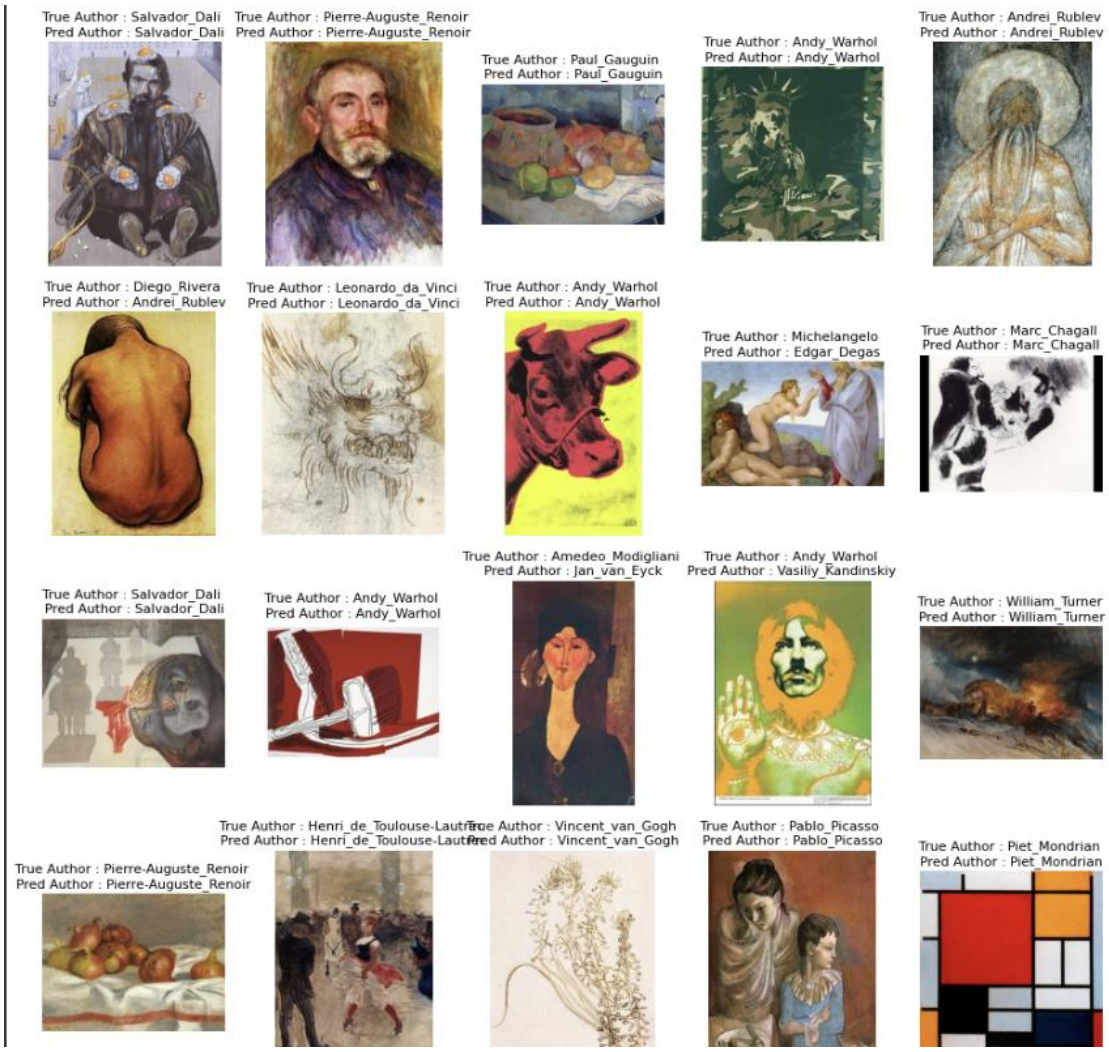



```
14/14 [==============================] - 3s 34ms/step - loss: 4.3362 - accuracy: 0.5006
Test loss: 4.336187839508057
Test accuracy: 0.5005987882614136
```

最終得到

Test_Acc = 0.5005987882614136

7.做預測



將圖片丟入模型預測

自己上傳的圖片預測

心得:

這次的作業對我來說十分有挑戰性，還有就是我覺得我真該早點寫，畢竟大部分時間都在等待其訓練，早點寫我只要偶爾去改一下參數就可以訓練很多次。而且沒有付費的話，它的 GPU 用量限制真的蠻麻煩的，好在我用了三個帳號來解決被鎖的問題。

修改模型花了我不少時間，我先堆出兩層 convolution + maxpooling，然後調整參數之外又多堆了幾層，然後 acc 居然達到穩定的 1，loss 小到不是用小數表示，但 val_loss 卻是一直越來越大，val_acc 也超低，是嚴重的過度擬和。為解決我才插入了 Dropout 並從 0.5 開始往下調整，然後又查到了據網路某文所述更推薦的 BatchNormalization，使用過後果然有效解決嚴重的過度擬和。

我覺得我的模型訓練時 accuracy 的起伏偏大，有時甚至會驟降許多，這應該是我的模型有很大進步空間的一個地方。訓練期間我有時看到前幾次 run 的蠻差的就直接中止，但後來我發現 epochs 不到一定數量可能都還很難說，所以後來我都盡量讓他跑完再下結論做修改。

很快就要寫下一個作業了，希望我能從這次作業的所學和教訓讓下次作業做得更順利更好。