姓名 ： 林子恩
學號 ：109403048
年級 ： 資管三 B
Colab :https://colab.research.google.com/drive/1NS5keDXxZLvi8Ue7M85sN21KBgJcJSr9?usp=sharing
Test Accuracy : 50.18%

撰寫過程

```
from google.colab import drive
drive.mount('/content/drive')

import zipfile
zip_ref = zipfile.ZipFile('/content/drive/MyDrive/AI_cnn/train.zip', 'r')
zip_ref2 = zipfile.ZipFile('/content/drive/MyDrive/AI_cnn/test.zip', 'r')

zip_ref.extractall('/content/train')
zip_ref2.extractall('/content/test')

zip_ref.close()
zip_ref.close()
```

用 google 雲端硬碟讀取，相信最後一個禮拜原本上面附的網址會被用爛掉

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
```

2. 取得資料集

- **我們來檢視一下 artist.csv**，重要內容包含:
  - 畫家名稱 (name)
  - 風格 (genre)
  - 資料集內畫作數量 (paintings)

總共有 50 位畫家，意謂著有 50 個 class 要去辨識。

```
[ ] train_dir = "/content/train/train_resized/"
    test_dir = "/content/test/test_resized/"
    artists = pd.read_csv("/content/train/artists.csv")
    num_classes = artists.shape[0]
    print("Number of artists : ", num_classes)
    artists.head()
```

```
Number of artists : 50
```

	id	name	years	genre	nationality	bio	wikipedia	paintings
0	0	Amedeo Modigliani	1884 - 1920	Expressionism	Italian	Amedeo Clemente Modigliani (Italian pronunciat...	http://en.wikipedia.org/wiki/Amedeo_Modigliani	193
1	1	Vasiliy Kandinskiy	1866 - 1944	Expressionism,Abstractionism	Russian	Wassily Wassilyevich Kandinsky (Russian: ?а?и?...	http://en.wikipedia.org/wiki/Wassily_Kandinsky	88
2	2	Diego Rivera	1886 - 1957	Social Realism,Muralism	Mexican	Diego María de la Concepción Juan Nepomuceno E...	http://en.wikipedia.org/wiki/Diego_Rivera	70
3	3	Claude Monet	1840 - 1926	Impressionism	French	Oscar-Claude Monet (; French: [klod mɔ̃nɛ]; 14 ...	http://en.wikipedia.org/wiki/Claude_Monet	73
4	4	Rene Magritte	1898 - 1967	Surrealism,Impressionism	Belgian	René François Ghislain Magritte (French: [ʁ?ne...	http://en.wikipedia.org/wiki/René_Magritte	194

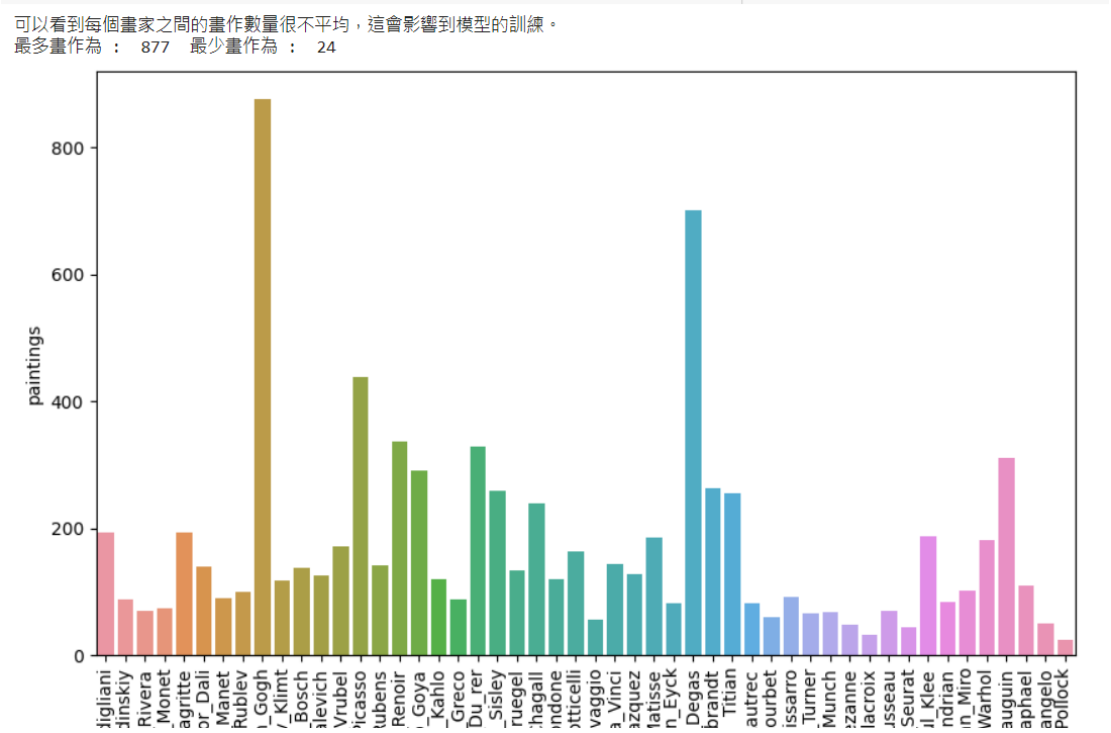
取得資料集

- 只取出名字與畫的數量，把名字用下底線連起來

```
artists = artists.loc[:, ["name", "paintings"]]
artists["name"] = artists["name"].str.split(" ").apply(lambda parts: "_".join(parts))
artists.head()
```

	name	paintings
0	Amedeo_Modigliani	193
1	Vasiliy_Kandinskiy	88
2	Diego_Rivera	70
3	Claude_Monet	73
4	Rene_Magritte	194

確定需要的資料



發現不同畫家有其中幾個的畫作數量特別多，有想過要調權重，但實行後，效果不盡理想，之後就沒有放了。

```
# 請建立將英文映射成數字的 dict。EX: Van_Gogh --> 0
def make_author_dict():
        name_dict = {}
        for i, name in enumerate(artists["name"]):
            name_dict[name] = i
        print(name_dict)
        ##########
        # todo #
        ##########
        return name_dict


class_name = make_author_dict()

# 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 class_name.items()}
print(rev_class_name)
```

```
{'Amedeo_Modigliani': 0, 'Vasiliy_Kandinskiy': 1, 'Diego_Rivera': 2, 'Claud
{0: 'Amedeo_Modigliani', 1: 'Vasiliy_Kandinskiy', 2: 'Diego_Rivera', 3: 'C
```

建立字典

```
# 決定你輸入模型的圖片長寬
IMG_WIDTH = 128
IMG_HEIGHT = 128
IMG_SIZE = (IMG_WIDTH,IMG_HEIGHT)
# shuffle buffer size
SHUFFLE_BUFFER = 3000
```

參數設定，原本想說調整圖片解析度小一點，準確率可以比較容易升高，後來還是用預設的 128*128 比較清楚，也比較不脫離模型設計的初衷。

```
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)
```

```
input_shape = (IMG_HEIGHT,IMG_WIDTH,3)
output_shape = (50,)

# 自訂你的 model
###########
# todo #

model = keras.Sequential([
        keras.Input(shape=input_shape),


        layers.Conv2D(32, kernel_size=(3, 3), activation="relu", padding='same'),
        layers.BatchNormalization(),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu", padding='same'),
        layers.BatchNormalization(),
        layers.MaxPooling2D(pool_size=(2, 2)),

        layers.Conv2D(64, kernel_size=(3, 3), activation="relu", padding='same'),
        layers.BatchNormalization(),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu", padding='same'),
        layers.BatchNormalization(),
        layers.MaxPooling2D(pool_size=(2, 2)),

        layers.GlobalAveragePooling2D(),
        layers.Dense(512, activation="relu"),
        layers.BatchNormalization(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
])
model.summary()
```

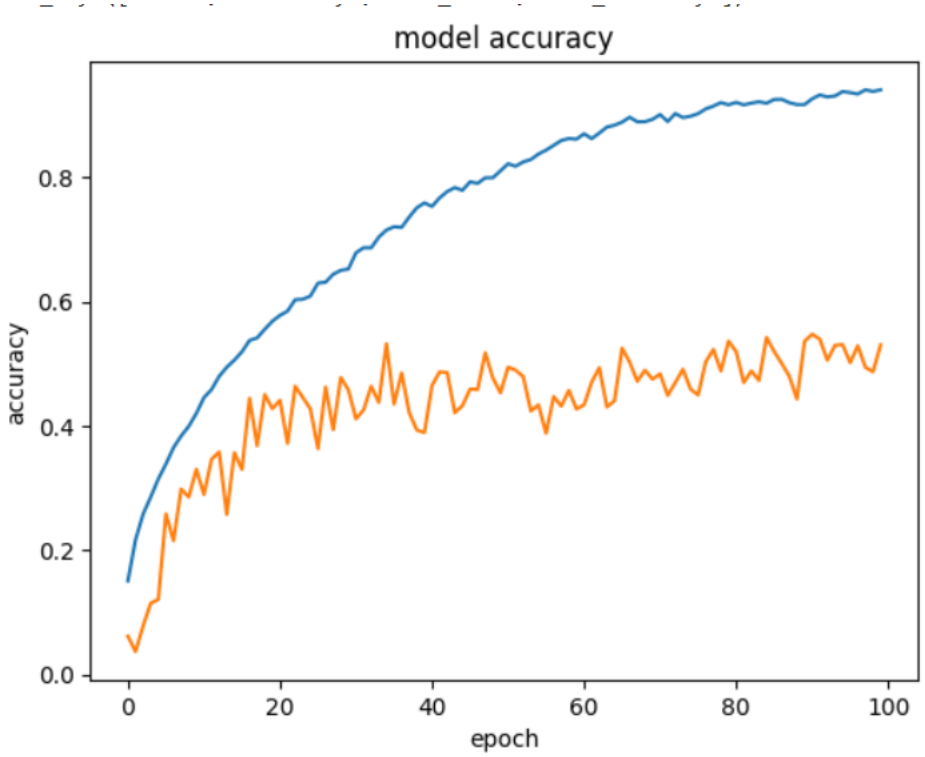
模型設計，調了很多次

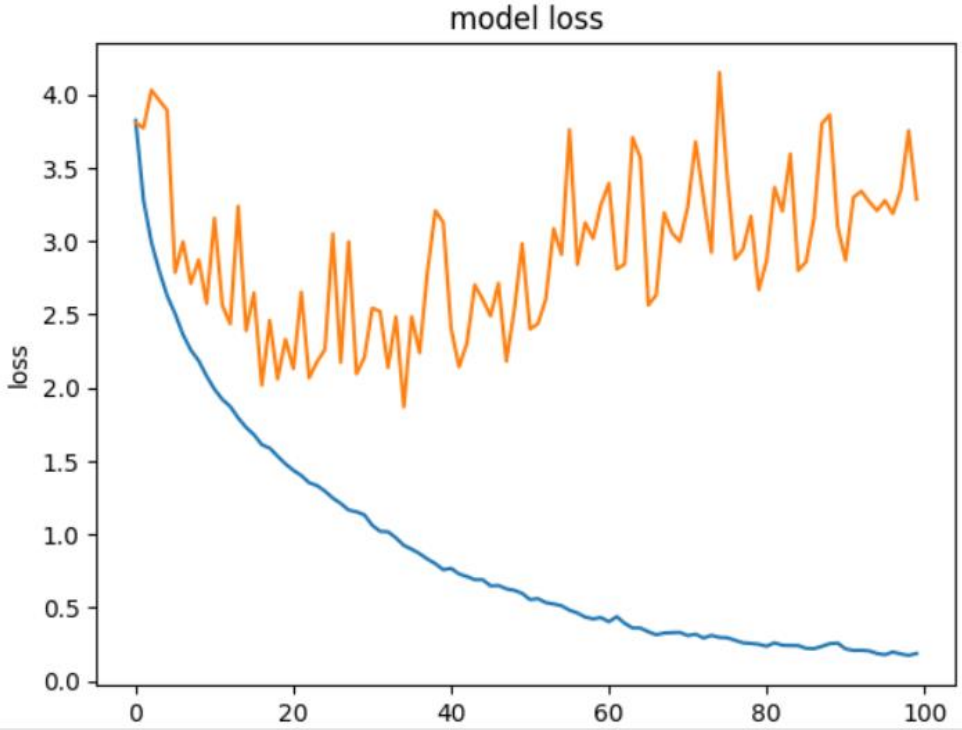
```
EPOCHS = 100

###########
# todo #
###########
# model.compile 決定 learning strategy、Loss calculator
model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
history = model.fit(train_ds , batch_size=BATCH_SIZE, epochs=EPOCHS, validation_data=val_ds)
```

```
94/94 [==============================] - 46s 414ms/step - loss: 0.3919 - accuracy: 0.8715 - val_loss: 2.8426 - val_accuracy: 0.4940
Epoch 64/100
94/94 [==============================] - 67s 617ms/step - loss: 0.3605 - accuracy: 0.8812 - val_loss: 3.7054 - val_accuracy: 0.4309
Epoch 65/100
94/94 [==============================] - 67s 619ms/step - loss: 0.3593 - accuracy: 0.8841 - val_loss: 3.5639 - val_accuracy: 0.4408
Epoch 66/100
94/94 [==============================] - 67s 618ms/step - loss: 0.3344 - accuracy: 0.8891 - val_loss: 2.5626 - val_accuracy: 0.5253
Epoch 67/100
94/94 [==============================] - 49s 420ms/step - loss: 0.3137 - accuracy: 0.8971 - val_loss: 2.6307 - val_accuracy: 0.5027
Epoch 68/100
94/94 [==============================] - 47s 421ms/step - loss: 0.3257 - accuracy: 0.8896 - val_loss: 3.1920 - val_accuracy: 0.4721
Epoch 69/100
94/94 [==============================] - 47s 409ms/step - loss: 0.3284 - accuracy: 0.8898 - val_loss: 3.0541 - val_accuracy: 0.4900
```

模型訓練結果


model accuracy
accuracy
epoch
0.0
0.2
0.4
0.6
0.8
0
20
40
60
80
100


model loss
loss
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0
20
40
60
80
100

準確率

```
#  讀入測試資料並評估模型
test_ds  =  make_dataset(test_dir)
test_ds  =  test_ds.batch(BATCH_SIZE)
score  =  model.evaluate(test_ds)
print("Test  loss:",  score[0])
print("Test  accuracy:",  score[1])
```

```
14/14 [==============================] - 4s 37ms/step - loss: 3.5898 - accuracy: 0.5018
Test loss: 3.589810371398926
Test accuracy: 0.5017964243888855
```

第七題

```
def  predict_author(img):
        #  寫個單圖片模型預測  function
        #  input  :  opencv  img  (height,width,3)
        #  output  :  某個作家名字  E.g.  Claude_Monet
        #
        #  參考步驟:
        #  1.  expand  img  dimension  (height,width,3)  ->  (1,height,width,3)
        #  2.  丟入模型  model.predict
        #  3.  取出  softmax  後  (50,)  取最大值的  index  作為辨識結果
        #  4.  將辨識結果轉為畫作家名字

        author_name  =  ""
        ##########
        #  todo  #
        ##########
        img  =  tf.expand_dims(img,  axis=0)

        author_num  =  model.predict(img)
        pred  =  np.argmax(author_num)
        author_name  =  rev_class_name[pred]

        return  author_name
```

```
eval()
```

選擇檔案 未選擇任何檔案 Upload widget is only available when the cell has be

```
Saving 螢幕擷取畫面_20221209_040316.png to 螢幕擷取畫面_20221209_040316.png
1/1 [==============================] - 0s 28ms/step
```

predict author : Rene_Magritte

心得

最後用一張心情寫照的圖做預測當作結尾，算是第一次正式訓練模型，惡補了很多影片教學跟資料。在設計模型的時候，犯了很多錯誤，連參數的意義都搞不清楚，中間再做資料前處理的時候也有遇到一些疑惑。實際訓練之後，遇到了很多問題跟挑戰，但也真的經歷調參數的痛苦過程，當初不管怎麼調都一直在 30~35 甚至有時候會意想不到的低，心情真的 emo，覺得這個模型怎麼跟我一樣可憐。後來發現太多層會導致 overfitting、合理的用 BatchNormalization 跟用 GlobalPooling2D 比 flatten 還更適合我的模型等等。

一開始 epoch 只設了 10~15 次，後來到 40%的那次，我發現圖形收斂得還不錯，相比之前的模型 overfitting 的情況也比較輕微，之後就以那個版本做微調，之後 epoch 設 20 得到 45%、50 得到 48%發現收斂越來越好，震盪也沒有

到很明顯，之後調到 100 試試就得到 50%，然後我就想見好就收，但之後我發現之前有幾個跟訓練無關只是把畫作圖展示出的幾格被我註解了，怕被扣分(雖然之後看好像也跟作業模型無關，但還是想有個完美的結束)，所以我就另開個副本，順便執行 200 次，想說看收斂的結果會不會更好或準確率可以更高，然後就發現執行第 192 次的時候，colab 的 gpu 用光，白白浪費 3 個多小時，所以之後我就把原本 50%的上面幾格的再執行一次，發現這樣也可以，衝動是魔鬼。

然後這次我是第二個禮拜才做的，大概做了一個禮拜，沒課金 colab 時訓練時間真的慢，過程真的痛苦，但看到 50 出來，直接飛起。這個 cnn 作業原本是想要做到更高，但之後有閒的時候再做好了，50%滿分我就滿足了，讚。