人工智慧與機器學習作業一　CNN 判別名畫

資管三 B 109403533 林采璇

一、專案連結：Google Colab 連結　（ACC = 53.5%）

二、過程：

1. 取得資料集：

   原本擔心公用IP會被鎖，所以有把資料集載下來丟雲端，並註解上mount資料夾、進行解壓縮的程式碼，以備不時之需。所幸後來運氣還不錯，沒用到。

```
import random
import os

# 大家盡量先把資料保存在本地端，然後要訓練時用本地端上傳做訓練
# 以節省學術網路資源，避免 IP 被封鎖

if not os.path.isfile("./train.zip"):
    !wget -O train.zip "http://140.115.83.111/files/art/train.zip"
    !wget -O test.zip "http://140.115.83.111/files/art/test.zip"
    !unzip train.zip
    !unzip test.zip
else:
    !echo "檔案已存在"

#萬一ip被鎖
'''
from google.colab import drive
drive.mount('MyDrive')

!unzip "/content/MyDrive/MyDrive/train.zip"
!unzip "/content/MyDrive/MyDrive/test.zip"
!echo 'unzip success'
'''
```

2. 資料前處理：

   - 建立字典：在make_author_dict函式把author的name跟index map起來，存於字典。再將字典反向從index map回name，存在rev_class_name內。

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

     class_name = make_author_dict()
     print (class_name)

     # 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)
```

- 路徑處理：
  - get_label：用split把pic_name從檔名中提取出來
  - get_path：把dir的路徑跟pic_name合併 return
  - make_paths_label：透過呼叫get_label、get_path將經過處理的label、path存到list內，再用to_catergorial轉成one_hot format

```
[ ]  def get_label(pic_name):
         # 請取出 label 並轉成數字
         # EX: Claude_Monet_1.jpg -> Claude_Monet -> 1
         label = ("_".join(pic_name.split("_")[:-1]))
         index = class_name[label]
         return index


     def get_path(dir, pic_name):
         # 請將路徑合併
         # EX: ./train_resized/ + Claude_Monet_1.jpg => ./train_resized/Claude_Monet_1.jpg
         path = dir + pic_name
         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 path in img_list:
           paths.append(get_path(dir,path))
           labels.append(get_label(path))

         # 將 labels 轉成 onehot
         from tensorflow.keras.utils import to_categorical
         onehot_labels = to_categorical(labels, num_classes)
         return paths, onehot_labels
```

- 資料不平衡處理(class_weight)：
  網路上拜讀許多文章，有看到資料不平衡可以使用under sampling、over sampling、class weight等方式解決，但對於我這個機器學習菜雞而言，怕貿然使用under sampling、over sampling會失真，所以便使用class weight處理。class weight的設定也相對單純，將每一類權重設成該類資料量的倒數。

```
[ ]  # 類別權重 資料不平衡處裡
     pic_list = os.listdir(train_dir)
     total_len = len(pic_list)
     print("training 畫作總共畫作有 : ",total_len)

     i = 0
     pic_num_list = {}
     class_weight = {}

     while i < 50:
       pic_num_list[i] = 0
       class_weight[i] = 0
       i += 1

     def getPicNumList():
       for name in pic_list:
         num = get_label(name)
         pic_num_list[num] += 1
       return pic_num_list

     def getClassWeight():
       pic_num_list = getPicNumList()
       for index in pic_num_list:
         class_weight[index] = total_len / pic_num_list[index]
       return class_weight

     pic_num_list = getPicNumList()
     class_weight = getClassWeight()
     print(pic_num_list)
     print(class_weight)
```

- 設定圖片長寬、shuffle_buffer size
  get_image：對圖片做正規化(映射0-1)
  make_dataset：將one-hot label與tensor map起來，然後打散

```
[ ]  # 決定你輸入模型的圖片長寬
     IMG_WIDTH = 256
     IMG_HEIGHT = 256
     IMG_SIZE = None
     # shuffle buffer size
     SHUFFLE_BUFFER = 3000


     def get_image(path):
         # read image from path
         file = tf.io.read_file(path)
         img = tf.io.decode_jpeg(file, channels=3)
         img = tf.cast(img, tf.float32)

         # 請固定每張圖片大小為 IMG_HEIGHT、IMG_WIDTH
         # 並將圖片每個 pixel 映射到 [0,1] 之間
         img /= 255.0
         img = tf.image.resize(img, [IMG_HEIGHT, IMG_WIDTH])
         return img
```

```
     # 將所有資料轉成 Tensor -> Tensor 轉成圖片
     # 圖片 Tensor 與 label Tensor Zip 起來成一個 pair
     # shuffle 打散
     def make_dataset(dir):
         paths, onehot_labels = make_paths_label(dir)
         paths_ds = tf.data.Dataset.from_tensor_slices(paths)
         train_label = tf.data.Dataset.from_tensor_slices(onehot_labels)

         # 將路徑 tensor 映射成圖片 tensor
         train_image = paths_ds.map(get_image)
         # 合併圖片與 label 資料集
         full_ds = tf.data.Dataset.zip((train_image, train_label))
         # 打散
         full_ds = full_ds.shuffle(SHUFFLE_BUFFER, reshuffle_each_iteration=False) #iteration若設true?
         return full_ds


     full_ds = make_dataset(train_dir)
```

- Batch Size 設定：32

```
[ ]  # 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 = 32   #設大於32會overfitting

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

3. 建立模型：

   誠實而言，Conv2D、MaxPooling的疊疊樂以及參數設定，基本上可以說是純靠直覺。只記得老師上課有說MaxPooling別放太多避免失真，所以沒有疊太多(雖然也有聽說有人疊不少，train的也還不錯的)。與此同時，搭配BatchNormalization以及Dropout避免過擬合。另外，老師上課提到的GlobalAveragePooling，實作起來比Flatten好，便取而代之了。

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

     # 自訂你的 model
     model = keras.Sequential(
         [
             keras.Input(shape = input_shape),
             layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
             layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
             layers.MaxPooling2D(pool_size=(2, 2)),
             layers.BatchNormalization(),

             layers.Conv2D(128, kernel_size=(3, 3), activation="relu"),
             layers.MaxPooling2D(pool_size=(2, 2)),
             layers.BatchNormalization(),

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

     model.summary()
```

4. 訓練計畫：

   最終版本的訓練計畫，是先跑15個沒權重的Epoch再跑100個有權重的Epoch。至於為甚麼這樣，只能說發現這種跑法，是個美麗的錯誤(?)

```
[ ]  from tensorflow.python.eager.monitoring import Metric
     # todo
     EPOCHS = 15   #隨便讓他多跑幾次

     ##########
     # todo #
     ##########
     # model.compile 決定 learning strategy、Loss calculator
     model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])

     history = model.fit(train_ds, epochs=EPOCHS, validation_data=val_ds)
     #history = model.fit(train_ds, epochs=100, validation_data=val_ds, class_weight = class_weight)
```

```
[ ]  from tensorflow.python.eager.monitoring import Metric
     # todo
     EPOCHS = 100   #隨便讓他多跑幾次

     ##########
     # todo #
     ##########
     # model.compile 決定 learning strategy、Loss calculator
     model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])

     #history = model.fit(train_ds, epochs=15, validation_data=val_ds)
     history = model.fit(train_ds, epochs=EPOCHS, validation_data=val_ds, class_weight = class_weight)
```

- ［沒有權重］
  最初的版本，是只有單純的模型(還沒加權重前)。參數調來調去以及模型疊疊樂的各種排列組合，最好的訓練結果的ACC一直在40%左右徘徊。

- ［混合的開端］
  後來，便決定處理資料不平衡的問題，加上權重。殊不知第一次要跑有權重的版本的時候，忘記把權重加上，先跑15個沒權重的Epoch。當時就想，那就將錯就錯，加上權重繼續多跑幾輪Epoch。結果發現，雖然剛開始，加上權重會「暫時」降低ACC，但後面訓練的Epoch增多，使它趨近訓練完全，結果就還不錯。該輪跑了Epoch=60，ACC有達到47.9%。(要不是閒置過久，google重新初始化執行階斷，不然還想再讓它多跑幾輪)

  - 權重「暫時」降低ACC的現象：
    https://ithelp.ithome.com.tw/articles/10269361

  - 該輪結果截圖：

```
[ ]  # 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])

     27/27 [==============================] - 5s 55ms/step - loss: 2.9705 - accuracy: 0.4790
     Test loss: 2.9704651832580566
     Test accuracy: 0.4790419042110443
```

- ［純權重］
  接著，我就想試著跑跑看，只有權重的版本，殊不知跑起來的狀況極差。甚至比沒有權重的版本還慘。測試了好幾輪，實在是找不出原因，便轉戰回混和版本。

- ［混合］
  前期沒有權重的試驗，讓我知道這個模型與訓練規畫若沒有權重，大概會在Epoch=15-20左右就會峰頂。所以我就姑且設沒有權重的Epoch=15。至於後面有權重的Epoch，原本大手一揮設了Epoch=200，發現差不多在120左右，ACC和Loss就收斂了差不多了。所以，後來重跑了一個Epoch = 100的，結果就達到ACC=53.5%，開心收工。

```
[ ]  # 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])

     27/27 [==============================] - 5s 51ms/step - loss: 2.9942 - accuracy: 0.5353
     Test loss: 2.994170665740967
     Test accuracy: 0.5353293418884277
```

5. 預測函式：

```
[ ] 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

        img = np.expand_dims(img,0)
        softmax = model.predict(img)
        softmax = softmax.flatten()
        softmax = softmax.tolist()
        refer = softmax.index(max(softmax))
        author_name = rev_class_name[refer]
        return author_name
```

```
[ ] plt.figure(figsize=(16, 16))
    for index, imgName in enumerate(show_imgs):
        img_path = train_dir + imgName
        img = cv.imread(img_path)
        img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
        plt.subplot(4, 5, index + 1)
        plt.axis("off")
        plt.imshow(img)
        img = cv.resize(img, (IMG_WIDTH, IMG_HEIGHT))
        img = img / 255.0
        plt.title(
            "True Author : {} \nPred Author : {}".format(
                "_".join(imgName.split("_")[:-1]), predict_author(img)
            ),
            size=11,
        )
```

結果：

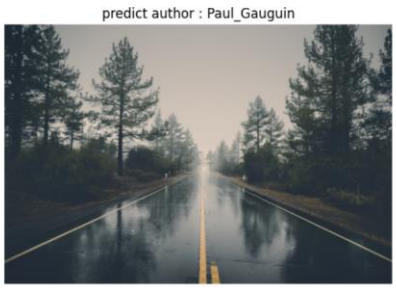
```
[ ] from google.colab import files


    def upload_img():
        uploaded = files.upload()
        img_name = list(uploaded.keys())[0]
        img = cv.imread(img_name)
        img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
        plt.imshow(img)
        img = cv.resize(img, (IMG_WIDTH, IMG_HEIGHT))
        img = img / 255.0
        return img


    def eval():
        img = upload_img()
        plt.title("predict author : {}".format(predict_author(img)))
        plt.axis("off")
        plt.show()
```

```
[ ] # 自己上傳一張圖片來試試看
    # Demo 圖片來自:
    # Interview with Cyberpunk 2077 "ponpon shit" producer Yuki Kawamura (https://block.fm/news/cyberpunk2077_uscracks_ENG)
    eval()
```

```
瀏覽... 未選擇檔案。   Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving road-landscape-nature-forest-39811.jpg to road-landscape-nature-forest-39811.jpg
1/1 [==============================] - 0s 24ms/step
```



三、心得：

剛開始建模時，常常不知道下手輕重，不是模型疊了太多層，就是參數也設的不知輕重，結果就常把 colab 的系統 RAM 和 GPU RAM 玩到超過上限，強制被初始化執行狀態，又或者是帳號被鎖也是常態(後來只好多個帳號切換來切換去的)。屏除掉這些不能跑模型的焦躁時光，其實跑模型以及觀察模型的過程，其實還滿有意思的。
最後，要感謝各個鐵人賽系列文章，讓我這個機器學習菜鳥，訓練起模型更有方向。
AI Facial Expression Recognition: Data, Model, Application
30 天在 Colab 嘗試的 30 個影像分類訓練實驗
學資料科學的小孩不會變壞- 從入門到實戰全攻略