英文三 109102516 吳珮宜

- Colab Link：

https://colab.research.google.com/drive/1obNMXIQJz4tR2Lm6JQE055h8puahalv_#scrollTo=oKHSjy-yLdnb

- Accuracy：0.475

```
[34] #  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 [==============================] - 3s 59ms/step - loss: 3.8167 - accuracy: 0.4754
     Test loss: 3.8167316913604736
     Test accuracy: 0.4754491150379181
```

- Training

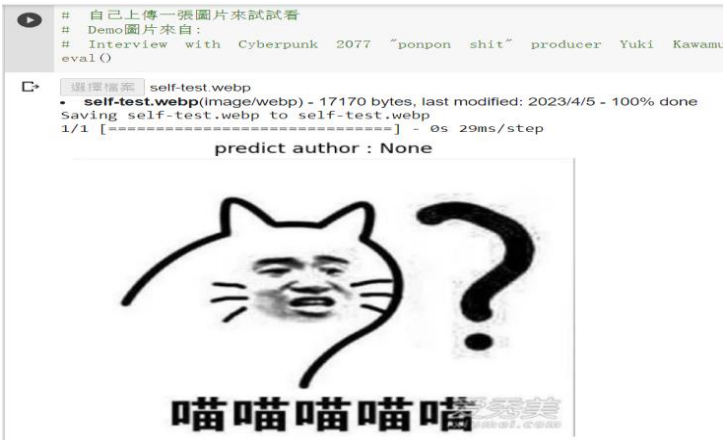
```
history  =  model.fit(train_ds,  epochs=EPOCHS,  validation_data=val_ds)

Epoch 1/100
94/94 [==============================] - 51s 449ms/step - loss: 3.5380 - accuracy: 0.1747 - val_loss: 8.0997 - val_accuracy: 0.0306
Epoch 2/100
94/94 [==============================] - 45s 452ms/step - loss: 3.0558 - accuracy: 0.2306 - val_loss: 8.0468 - val_accuracy: 0.0306
Epoch 3/100
94/94 [==============================] - 46s 473ms/step - loss: 2.8156 - accuracy: 0.2741 - val_loss: 6.2699 - val_accuracy: 0.0399
Epoch 4/100
94/94 [==============================] - 44s 449ms/step - loss: 2.6011 - accuracy: 0.3093 - val_loss: 6.1556 - val_accuracy: 0.0645
Epoch 5/100
94/94 [==============================] - 45s 462ms/step - loss: 2.4504 - accuracy: 0.3454 - val_loss: 4.8613 - val_accuracy: 0.1011
Epoch 6/100
94/94 [==============================] - 43s 446ms/step - loss: 2.2624 - accuracy: 0.3831 - val_loss: 3.8309 - val_accuracy: 0.1988
Epoch 7/100
94/94 [==============================] - 45s 453ms/step - loss: 2.1554 - accuracy: 0.4053 - val_loss: 2.7625 - val_accuracy: 0.3318
Epoch 8/100
94/94 [==============================] - 43s 446ms/step - loss: 2.0104 - accuracy: 0.4418 - val_loss: 2.6397 - val_accuracy: 0.3191
Epoch 9/100
94/94 [==============================] - 45s 452ms/step - loss: 1.8788 - accuracy: 0.4719 - val_loss: 2.5256 - val_accuracy: 0.3697
Epoch 10/100
94/94 [==============================] - 64s 670ms/step - loss: 1.7697 - accuracy: 0.4940 - val_loss: 3.3051 - val_accuracy: 0.2380
Epoch 11/100
94/94 [==============================] - 43s 446ms/step - loss: 1.6185 - accuracy: 0.5361 - val_loss: 2.5646 - val_accuracy: 0.3298
Epoch 12/100
94/94 [==============================] - 45s 450ms/step - loss: 1.5049 - accuracy: 0.5582 - val_loss: 2.3348 - val_accuracy: 0.3983
Epoch 13/100
94/94 [==============================] - 64s 672ms/step - loss: 1.3910 - accuracy: 0.5946 - val_loss: 2.5205 - val_accuracy: 0.3570
Epoch 14/100
94/94 [==============================] - 45s 466ms/step - loss: 1.3114 - accuracy: 0.6110 - val_loss: 2.3300 - val_accuracy: 0.3903
Epoch 15/100
94/94 [==============================] - 43s 447ms/step - loss: 1.1810 - accuracy: 0.6415 - val_loss: 2.5000 - val_accuracy: 0.3590
Epoch 16/100
94/94 [==============================] - 45s 450ms/step - loss: 1.0502 - accuracy: 0.6820 - val_loss: 2.4936 - val_accuracy: 0.3723
```

- 資料前處理(基本上都照著助教的 todo 步驟撰寫。)

1. 資料集有先下載一份在 local 端，以防每次訓練時過度占用學術網路資源導致 IP 被鎖掉。
2. 從計算各個畫家的畫作數量看出 class 資料的不平均數量，會導致後面的模型訓練過度擬和首先丟入的類型，因此按照助教說的使用 shuffle()，讓數據打亂滿足所需的隨機性，讓模型成功學習不同數據的特徵避免 overfitting 的情況。

3. 要將作者名稱作為 class name，因此把作者英文名子跟數字用 make_author_dict()做 dict 以方便訓練。
4. 從畫作路徑提取作者名稱(label)，取出”_’前的字串並投過 dict 返回作者映射的數字。
5. 轉回 label 的數值沒有順序意義，再轉成 oneshot，由 1 的顯示位置可看出是哪一位作者。
6. 輸入模型的圖片長寬設定為(128,128)，將 SHUFFLE_BUFFER 設為 640 以及將圖片每個 pixel 映射到[0,1]之間。

- 建立模型
  1. 使用 Keras.Sequential 建立模型，設定 input size 後開始用 convolution 以及 maxpooling ，每一輪都使用 batch nomalization 使用模型收斂，後來也會使用 dropout 丟棄一定比例的 neuron，避免 overfitting 的狀況，發現準確率也有上升的趨勢。
  2. 原本使用 flattern 後改用 GlobalAveragePooling2D()，準確率上升。
- 制定訓練計畫
  1. Epochs 訓練週期設為 100 會相較 50 有準確率提高。
  2. 使用上課說的 optimization=Adam。
  3. 使用 model.fit()將 train.ds, epochs , validation_data 丟進去。
- 做預測：
  1. 在這邊不小心手殘改到一個 Function，抓了一兩小時才找出來。
  2. 使用 np.expand_dims()，增加 img 的維度，再使用 np.argmax 找出模型預測的最大值 index，找出最有可能的作家，用 make_author_dict() 將數字存入 authorname 回傳。



- 心得：

能夠完成這次的作業，首先還是得感謝我自己的交友廣闊跟強悍的意志力，之前接觸 ML 只有要寫很簡單的通識專題，雖然這是我第二次接觸 machine learning，但卻完全不知道如何踏出第一步。

所以我第一步就去問資工系的朋友(哭)，閱讀完他給我的很多資料，再去查詢前人們模型訓練的程式碼，大概了解建立模型的步驟，前面的簡單輸入路徑、映設英文跟數字等等都算簡單好處理，但在訓練模型的過程，準確率卻是不見很大的起色，而且每一次訓練過程都相當耗時，幸虧到後期，多次調整模型後，Test Accuracy 從原本 0.31 上升至了 0.475，但我覺得圖看起來是顯示模型有 Overfitting 的跡象，所以我就多輸入幾次正規化跟 Dropout()，但因為在重複嘗試的過程中，我的 GPU 一直被鎖起來，且也逼近了作業繳交期限，等之後學習了更多 CNN 的知識後，我再回來提升這次模型的準確率跟改善這個問題！