HW4
尋找其他 AI 應用方式、code

組別：為什麼要醬組

組員一：陳胤琲　B103012001

組員二：林凡皓　B103012002

組員三：洪漢霖　B103012011

組員四：吳尚恩　B103040033

目錄

一、使用背景

1. 解決甚麼問題：

人工智慧的蓬勃發展給人類的生活帶來許多便利，但同時也產生出許多新的隱憂，例如：破解密碼、惡意軟體.....等，其中，透過 AI 生成與修改圖片來進行詐騙為很嚴重的問題。之所以 AI 生成圖片可以拿來做詐騙，是因為 AI 生成或修改過的圖片看起來跟真的差不多，人眼很難做區隔。以下為範例：

圖片來源：https://zh.cn.nikkei.com/trend/cool-japan/31609-2018-08-02-05-00-20.html

上面四個女生的照片皆為利用日本新創企業 DataGrid 所開發的自動生成虛擬偶像頭像照片的 AI 所生成的圖片。乍看之下其實跟真實的人類幾乎一樣，單單靠肉眼是無法順利辨別的，因此我們希望透過深度學習的幫助，來辨別 AI 生成圖片。

2. 使用 AI 的優勢是什麼

就如同上面所講的範例，透過 AI 生成或是修改過的圖片單靠肉眼是無法順利辨別，主要原因在於肉眼無法有效地抓取到 AI 生成圖片與真實圖片之間特徵的差距。但是對於神經網路來說，抓取特徵基本上是輕而易舉，因此透過神經網路的幫助，我們可以有更高的準確度來分辨是否為 AI 生成圖片。此外，如果是靠人眼來分辨是否為 AI 生成圖片，將會需要大量人力資源投入這項工作，導

致效率低的問題，而透過 AI 的輔助，人力資源可以節省許多，效率也會有所提升。

二、 實際運行 source code

這次作業我們主要分成兩個部分，分別為偵測 AI 生成物件的圖片以及偵測 AI 生成人臉的圖片。

1. 偵測 AI 生成物件的圖片 ：

- Source code & dataset 來源 ：

  Source code 來源 ：

  https://www.kaggle.com/code/manothamdamnoen/vgg16-model

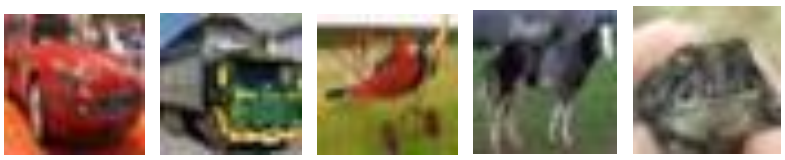
  Dataset 來源 ：https://www.kaggle.com/datasets/birdy654/cifake-real-and-ai-generated-synthetic-images/data

- Source code & dataset 說明 ：

  CIFAKE 為結合 AI 生成圖片與 CIFAR-10 資料集的結果。

  CIFAKE 中含有 60000 張 AI 合成的圖片以及 60000 張真實圖片，AI 合成圖片都是根據真實圖片做一些修改而產生，部分資料如下 ：

  (1) REAL :

  (2) FAKE :

  關於 source code 的部分，作者採用 VGG16 來做 transfer leaning，VGG16 架構如下

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 32, 32, 3)]       0

 block1_conv1 (Conv2D)       (None, 32, 32, 64)        1792

 block1_conv2 (Conv2D)       (None, 32, 32, 64)        36928

 block1_pool (MaxPooling2D)  (None, 16, 16, 64)        0

 block2_conv1 (Conv2D)       (None, 16, 16, 128)       73856

 block2_conv2 (Conv2D)       (None, 16, 16, 128)       147584

 block2_pool (MaxPooling2D)  (None, 8, 8, 128)         0

 block3_conv1 (Conv2D)       (None, 8, 8, 256)         295168

 block3_conv2 (Conv2D)       (None, 8, 8, 256)         590080

 block3_conv3 (Conv2D)       (None, 8, 8, 256)         590080

 block3_pool (MaxPooling2D)  (None, 4, 4, 256)         0
...
Total params: 14879041 (56.76 MB)
Trainable params: 14879041 (56.76 MB)
Non-trainable params: 0 (0.00 Byte)
```

關於資料預處理，作者只有使用 normalization (即將 input 資料除以 255)。

參數設置部分，optimizer 採用 SGD、learning rate = 0.001、loss function 使用 binary cross entropy。

- 版本問題 :

由於此 source code 採用的是最新版本的 tensorflow，如果希望放到樹莓派上做運行的話就需要將 tensorflow 版本修改成 1.13.1。

我們嘗試兩種方法解決版本問題

1. 安裝最新版本 tensorflow :

   我們重新架設一個新的環境，新環境中 tensorflow 版本為 2.15.0。透過這個新環境能夠順利執行 source code。

2. 修改 code :

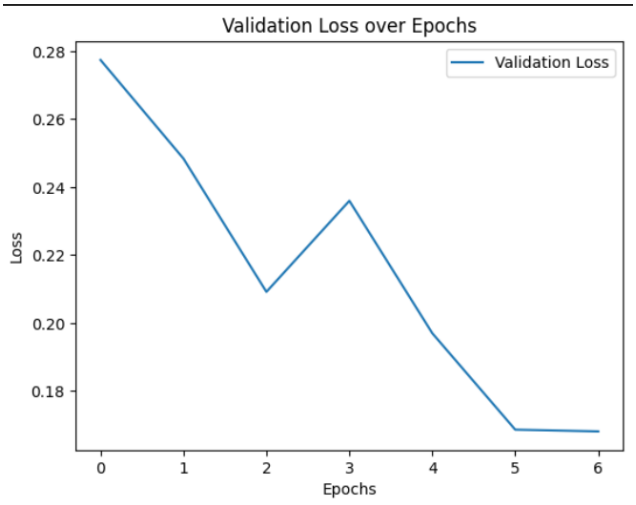
   我們直接將 source code 放到 tensorflow1.13.1 版本上運行，結果會出現像是不支援某函數、layer 名稱不一樣或是 import 套

件的路徑不一樣等問題。我們根據編譯器報錯資訊，並到一些網站像是 stack overflow、github、CSDN 上去搜尋解決辦法。

- 訓練過程與結果 ：

這邊的結果為在 tensorflow1.13.1 上運行的結果。

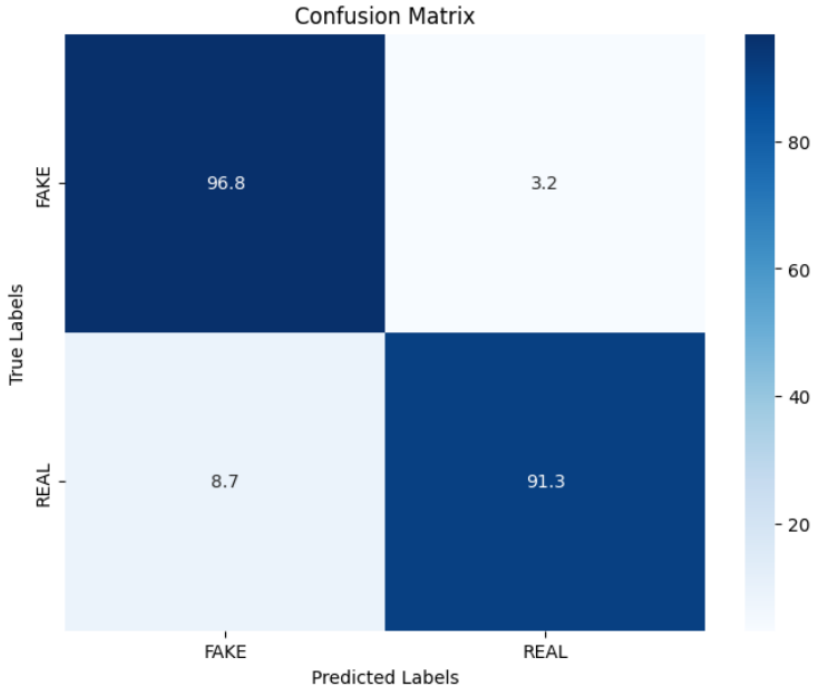
將訓練過程的 loss 視覺化，結果如下圖所示



訓練完成後，我們使用 testing dataset (都是訓練過程中沒有使用的資料) 來評估我們模型的表現。結果如下

```
Accuracy: 0.9408

Confusion Matrix:
[[9684  316]
 [ 868 9132]]

Classification Report:
              precision    recall  f1-score   support

        FAKE       0.92      0.97      0.94     10000
        REAL       0.97      0.91      0.94     10000

    accuracy                           0.94     20000
   macro avg       0.94      0.94      0.94     20000
weighted avg       0.94      0.94      0.94     20000


Mean Average Precision (mAP): 0.9886067057960659
```

可以看到最終模型 F1 socre 落在 0.94 左右，接著透過 confusion matrix 來查看偵測真實圖片與 AI 合成圖片時的表現差異，結果如下



- 心得討論 ：

  這個模型是透過 CIFAKE 資料及訓練出來的結果，資料中大部分圖片都是物件而非人臉，我們嘗試將這個模型用來偵測真實人臉圖片與 AI 合成人臉圖片，結果如下 ：

```
Accuracy: 0.5577319587628866

Confusion Matrix:
[[494  42]
 [387  47]]

Classification Report:
              precision    recall  f1-score   support

        FAKE       0.56      0.92      0.70       536
        REAL       0.53      0.11      0.18       434

    accuracy                           0.56       970
   macro avg       0.54      0.51      0.44       970
weighted avg       0.55      0.56      0.47       970


Mean Average Precision (mAP): 0.4890927245292659
```

  可以看到準確度下降到只有 55.8 %左右，表現差強人意，因此我們重新尋找訓練資料，試圖訓練出一個專門針對偵測 AI 頭像與真實頭像的模型。

2. 偵測 AI 生成人臉的圖片 ：

- Source code & dataset 來源 ：

  source code 來源 ：

https://www.kaggle.com/code/venvennnn/using-mobilenetv1

dataset 來源 ：

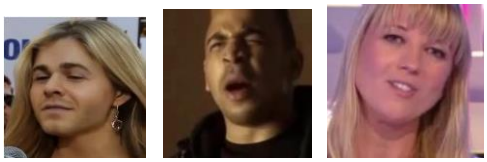
https://www.kaggle.com/datasets/manjilkarki/deepfake-and-real-images/data

- Source code & dataset 說明 ：

  這次使用的資料集主要內容為 AI 合成頭像與真實頭像。部分資料如下 ：

  (1) REAL :

  (2) FAKE :



  關於 source code，作者採用 MobileNet 做 transfer learning。模型架構如下

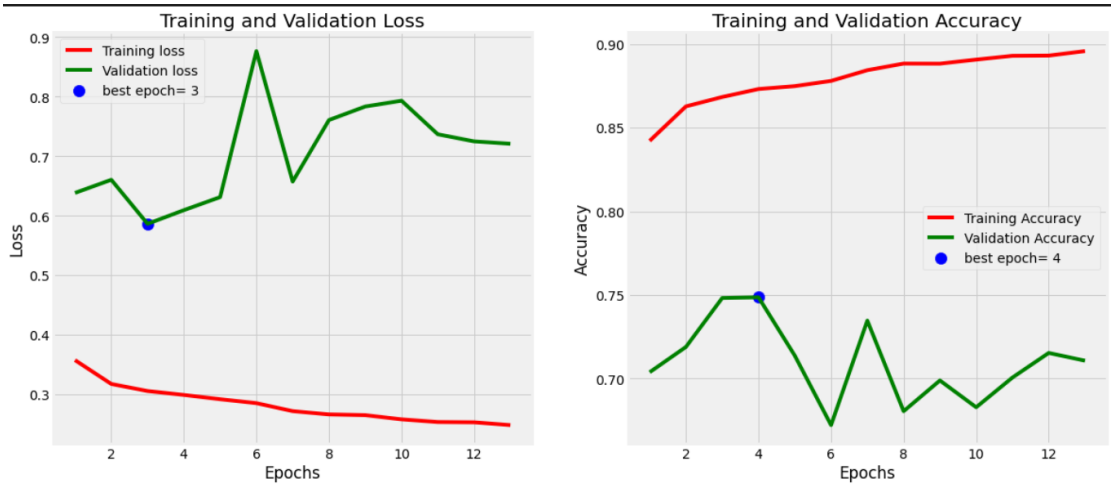
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
conv1_pad (ZeroPadding2D)    (None, 225, 225, 3)       0
_________________________________________________________________
conv1 (Conv2D)               (None, 112, 112, 32)      864
_________________________________________________________________
conv1_bn (BatchNormalization (None, 112, 112, 32)      128
_________________________________________________________________
conv1_relu (ReLU)            (None, 112, 112, 32)      0
_________________________________________________________________
conv_dw_1 (DepthwiseConv2D)  (None, 112, 112, 32)      288
_________________________________________________________________
conv_dw_1_bn (BatchNormaliza (None, 112, 112, 32)      128
_________________________________________________________________
conv_dw_1_relu (ReLU)        (None, 112, 112, 32)      0
_________________________________________________________________
conv_pw_1 (Conv2D)           (None, 112, 112, 64)      2048
_________________________________________________________________
conv_pw_1_bn (BatchNormaliza (None, 112, 112, 64)      256
_________________________________________________________________
conv_pw_1_relu (ReLU)        (None, 112, 112, 64)      0
_________________________________________________________________
...
Total params: 54,912,178
Trainable params: 51,680,978
Non-trainable params: 3,231,200
```

關於資料預處理，作者採用 tensorflow 內建的 ImageDataGenerator 來讀取資料，並將資料做 normalization (即除以 255)。

參數設置部分，optimizer 採用 adam、loss function 採用 sparse categorical cross entropy。此外，作者還有使用 early stopping、reduce learning rate 和 check point 來避免 overfitting 的發生。

- 訓練過程與結果 (在 tensorflow1.13.1 上執行)：

將訓練結果視覺化後，結果如下



可以看到模型在 epochs = 4 時可以得到最佳準確度為 75 %。之所以訓練曲線看起來會如此奇怪，原因在於 source code 是使用 pre-trained model 做訓練，因此在最一開始模型的準確度就達到最佳值，導致說 validation accuracy 上下彈跳，而不是穩定提升。

接著利用 testing set(皆為模型沒看過的資料)做測試，結果如下

	precision	recall	f1-score	support
0	0.78	0.57	0.66	5492
1	0.66	0.83	0.74	5413
accuracy			0.70	10905
macro avg	0.72	0.70	0.70	10905
weighted avg	0.72	0.70	0.70	10905

可以看到 testing F1 score 為 0.7。

- 討論 ：

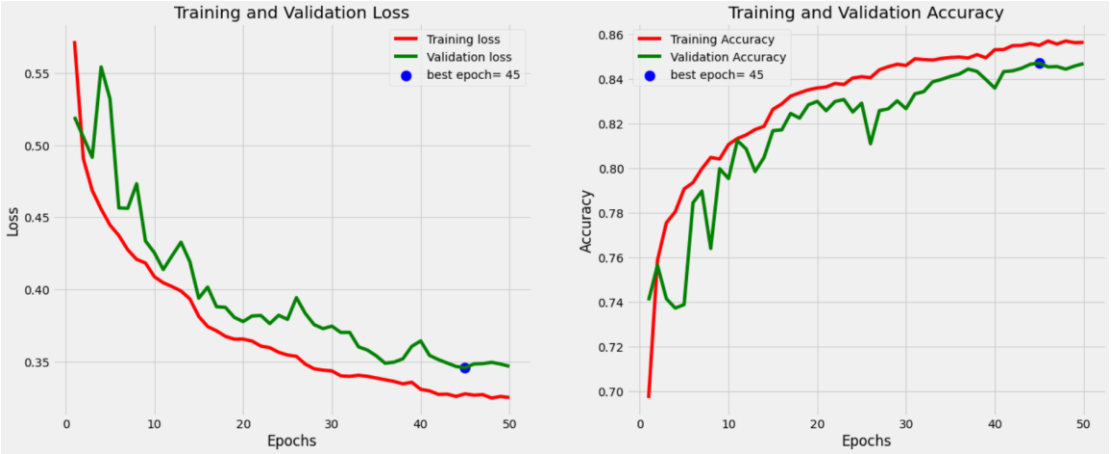
  考慮到前一次作業將 MobileNet 放到樹莓派上之後，運算速度下降很多，使用 ResNet50 甚至無法在樹莓派上正常運作，因此這次在找 source code 的時候就有特別注意到這點。

  根據這個 source code 訓練出來的模型 F1 score 只有 70 %，此外，他輸入的圖片大小為 224*224，放到數莓派上很可能會動不了，因此我們嘗試修改 source code 試圖提升準確度以及算速度。

- 修改 source code 結果與比較 ：

  首先，source code 在做 transfer learning 的時候並沒有做 fine tuning，因此我們對 source code 訓練完的模型做 fine tuning。此外，為了加速在樹梅派上運算速度，我們將圖片大小改為 32*32。

  訓練過程視覺化後結果如下



可以看到最好的結果發生在 epochs = 45 的時候，validation accuracy 約為 85 %左右。

拿此模型對 testing set 做測試，結果如下

	precision	recall	f1-score	support
0	0.73	0.84	0.78	5492
1	0.81	0.68	0.74	5413
accuracy			0.76	10905
macro avg	0.77	0.76	0.76	10905
weighted avg	0.77	0.76	0.76	10905

可以看到經過 fine tuning 之後，即便將圖片大小縮小到 32*32，

模型 F1 score 仍然可以提升到 0.76。

比較修改圖片大小前後的運算速度(筆電上測試)，結果如下

```
Found 10905 images belonging to 2 classes.

Time for predicting : 10.522435903549194

              precision    recall  f1-score   support

           0       0.72      0.85      0.78      5492
           1       0.81      0.67      0.74      5413

    accuracy                           0.76     10905
   macro avg       0.77      0.76      0.76     10905
weighted avg       0.77      0.76      0.76     10905
```

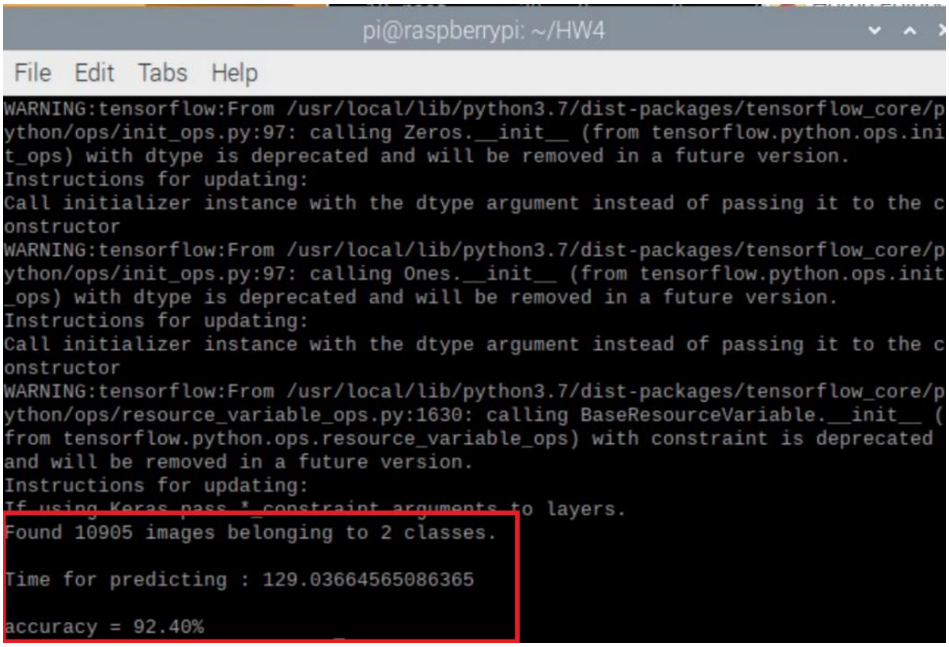
```
Found 10905 images belonging to 2 classes.

Time for predicting : 116.55790781974792

              precision    recall  f1-score   support

           0       0.78      0.57      0.66      5492
           1       0.66      0.83      0.74      5413

    accuracy                           0.70     10905
   macro avg       0.72      0.70      0.70     10905
weighted avg       0.72      0.70      0.70     10905
```

上面的圖片為 32*32 的測試結果，下面的圖片為 224*224 的測試結果。可以看到對 10905 張測資作測試的時間差來到 106 秒左右，差非常多。

這次作業中，我們選擇犧牲掉一些準確度來換取運算速度上的提升。對於偵測真實人臉與 AI 合成人臉這項任務來說，其實是非常困難的，原因在於真實人臉與 AI 合成人臉的特徵差距非常細微，想要去抓取到所有特徵將會需要很多層卷積層以及參數的協助，因此使用 MobileNet 這種小型神經網路做這項任務物 F1 score 大約都落在 70 %上下。

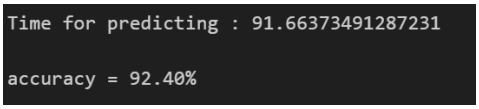
三、 實作於樹莓派

這次作業我們主要是實作偵測 AI 合成頭像與真實頭像的模型，因此樹梅派實作部分我針對修改 source code 後的模型做測試。由於在樹莓派上裝 scikit-learn 時，無法順利安裝 scipy，因此我們將 F1 score 改為 accuracy ($acc = \frac{correct\ samples}{total\ samples}$)，並利用 python time 套件觀察預測時間上的差距。以下為樹莓派運行結果。



可以看到對 10905 張測資作預測總共花費 129 s 左右，準確度為 92.4 %。

接著放上在筆電上運行結果，筆電規格如下 ：13th Gen Intel® Core™ 17-13700H、NVIDA GeForce RTX-4050 Laptop GPU。運行結果如下圖



準確度和在樹莓派上測試結果相同，但是預測時間為 91.66 s 左右，比樹莓派快上 37 s 左右。

四、 組員分工比例

組員	陳胤璁 B103012001	林凡皓 B103012002	洪漢霖 B103012011	吳尚恩 B103040033
分工比例	25%	25%	25%	25%
簽名	陳胤璁	林凡皓	洪漢霖	吳尚恩