Final Project Report

組別:大黑

成員:
B07501024 土木四 陳冠亦
B07501029 土木四 黃司睿
B07501061 土木四 謝語哲

目錄

1. 如何處理圖片

步驟1　ImageDataGenerator

Training data 的部分我們有用 rotation, zoom, width_shift, height_shift, horizontal_flip, 而照片都用 1./255. rescale, validation 用 20%的照片。

步驟2　ImageDataGenerator.flow_from_dataframe

我們試過許多照片尺寸，從 200 多到最後試到結果最好的是 600，batch_size 經試驗過最後取 10，我們自己覺得是非常消耗時間的設置，加上我們自己沒有硬體，跑得很痛苦。

```
train_generator = gen.flow_from_dataframe(
train_df, # dataframe
directory = train_dir, # images data path / folder in which images are there
x_col = 'Name',
y_col = 'Type',
subset="training",
color_mode="rgb",
target_size = (600,600), # image height , image width
class_mode="categorical",
batch_size=10,
shuffle=True,
seed=42,
)


validation_generator = gen.flow_from_dataframe(
train_df, # dataframe
directory = train_dir, # images data path / folder in which images are there
x_col = 'Name',
y_col = 'Type',
subset="validation",
color_mode="rgb",
target_size = (600,600), # image height , image width
class_mode="categorical",
batch_size=10,
shuffle=True,
seed=42,
)
Found 7420 validated image filenames belonging to 4 classes.
Found 1860 validated image filenames belonging to 4 classes.
```

步驟3　adding class_weight

在 fit model 的時候我們發現加上 class weight 也有改善結果，之後的 model 都有加上 class weight 將 imbalance 的問題考慮進 model fitting 裡面。

```
from sklearn.utils import class_weight
ClassWeights = dict(zip(np.unique(train_labels),
                        class_weight.compute_class_weight('balanced',
                                                          classes=np.unique(train_labels),y=tr
ain_labels)))
print(ClassWeights)
ClassWeights ={0.0: 0.8672276764843385, 1.0: 1.3081805359661496, 2.0: 1.190629011553
2734, 3.0: 0.8047722342733189}
```

2. 使用遷移學習 transfer learning

步驟4　使用 InceptionResNetV2 當作 Basemodel

在了解到 transfer learning 好像是最好的選項後，我們開始沒多久就直接嘗試 transfer learning 的 model 了，其中試過 VGG, EfficientNetB3, B4, Xception, 等多種模型，最後結果最好的是 InceptionResnetV2，我們就以這個模型為主開始訓練。

```
base_model = tf.keras.applications.InceptionResNetV2(
                    include_top=False,
                    weights='imagenet',
                    input_shape=(600,600,3)
```

步驟5　增加 Dense layer

我們接著用 Sequential 在 base model 後面加了幾層 layer，如下圖。

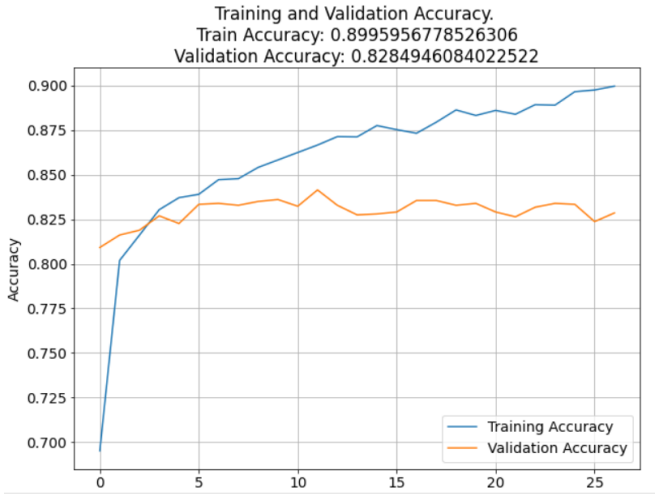
```
base_model.trainable=False

model = tf.keras.Sequential([
        base_model,
        tf.keras.layers.BatchNormalization(renorm=True),
        tf.keras.layers.GlobalAveragePooling2D(),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(256, activation='relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(4, activation='softmax')
    ])
```

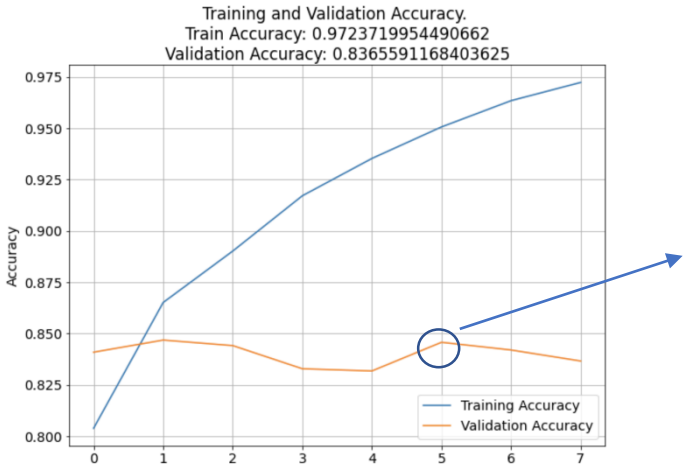
步驟6　訓練過程

我們訓練的方式是，首先，把 base model freeze 起來，去 train 我們自己加的那些 Dense layer，用的 optimizer 是 Adam，learning rate = 1e-4，其中一次的 curve 如下:

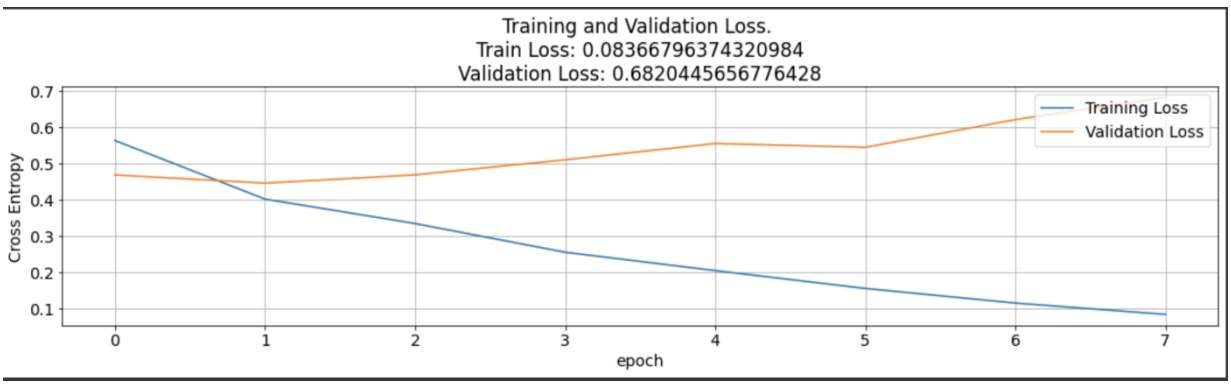
```
Epoch 1/50
742/742 [==============================] - ETA: 0s - loss: 0.8326 - accuracy: 0.6951
Epoch 00001: val_accuracy improved from -inf to 0.80914, saving model to
gdrive/MyDrive/final_project/code/resnetV2_augment_lr_450_3.h5
742/742 [==============================] - 2737s 4s/step - loss: 0.8326 - accuracy: 0.6951 - val_loss:
0.5136 - val_accuracy: 0.8091
Epoch 2/50
742/742 [==============================] - ETA: 0s - loss: 0.5689 - accuracy: 0.8019
Epoch 00002: val_accuracy improved from 0.80914 to 0.81613, saving model to
gdrive/MyDrive/final_project/code/resnetV2_augment_lr_450_3.h5
742/742 [==============================] - 646s 870ms/step - loss: 0.5689 - accuracy: 0.8019 -
val_loss: 0.4935 - val_accuracy: 0.8161
Epoch 3/50
742/742 [==============================] - ETA: 0s - loss: 0.5336 - accuracy: 0.8160
Epoch 00003: val_accuracy improved from 0.81613 to 0.81882, saving model
to .gdrive/MyDrive/final_project/code/resnetV2_augment_lr_450_3.h5
```



接著，這部分訓練結束後，我們將整個 base model 變成 trainable，再將 learning rate 調更小，做微調，會這樣調是我們嘗試之後的結果，雖然上課好像有說不應該全部解開，但我們只解凍幾層的結果沒有比較好，通常這部分可以看到的現象是 val_accuracy 會較上一階段跳高，但也很早就不會繼續成長，即使如此，因為我們的模型設定，都還是要跑很久。



Validation Accuracy 最好可到 0.8495



步驟7　用 pseudo label 重複訓練

我們到了這步雖然已經過 strong baseline，成績卻也都上不太去，我們就嘗試 pseudo labeling 的方式，我們設定了模型有 85%以上肯定答案才把圖留下來，第一次成績並沒有提高，但二、三個 iteration 都有不錯的提升，非常開心，我們總共做了 4 個 iteration，就沒時間繼續跑了，有點可惜。

步驟8　Ensemble

我們最後還選擇了一些成績好的作 ensemble，成績也有繼續往上，但這部分都是看著 public score 做的，感覺有可能只是在 overfit public score，但是看到有提高還是很興奮。

3. 使用的 model 以及 fine tuning 的心得

模型成績紀錄

Model	Validation Accuracy	Public Accuracy
NASNetLarge (299,299,3)	0.83	0.81
vgg19	0.76	0.74
efficientnetb3	0.34	x
InceptionResNetV2 (384, 384,3)	0.82	0.8
InceptionResNetV2 (420, 420,3)	0.84	0.817
加上 image augmentation 和 class_weight		
InceptionResNetV2 (450, 450,3)	0.85	0.834
InceptionResNetV2 (600, 600,3)	0.85	0.837
用 pseudo unlabel 再去訓練		
600_psuedo_iter1	0.84	0.82
600_psuedo_iter2	0.854	0.84
600_psuedo_iter3	0.86	0.845
ensemble		
Ensemble 0.83 以上的	x	0.856

一開始用很多不同的 CNN 的模型，像是近期表現最好的 efficientnet 系列但效果都不佳，比較簡單的 Vgg19 成績也不好，其中 InceptionResNetV2 是表現最好的一個，因此決定先專心在 InceptionResNetV2 上，並加上 image augmentation 和 將圖片放大到(600*600)，把訓練資源投入在 InceptionResNetV2，而之所以不將圖片放大到 800*800，是因為記憶體會裝不下。
過程中我們也將原本有點過大(0.001)的 learning rate 下修，也是我們成績提升的一個重要調整。

<u>最終使用模型架構:</u>

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_resnet_v2 (Functi  (None, 12, 12, 1536)     54336736
onal)

batch_normalization_203 (Ba  (None, 12, 12, 1536)     10752
tchNormalization)

global_average_pooling2d (G  (None, 1536)             0
lobalAveragePooling2D)

dense (Dense)                (None, 128)              196736

dropout (Dropout)            (None, 128)              0

dense_1 (Dense)              (None, 256)              33024

dropout_1 (Dropout)          (None, 256)              0

dense_2 (Dense)              (None, 4)                1028

=================================================================
Total params: 54,578,276
Trainable params: 54,510,052
Non-trainable params: 68,224
```

InceptionResNetV2 模型介紹

下面 InceptionResNetV2 在 keras 的官網上的成績，在 keras 中排名非常前面

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
InceptionResNetV2	215	0.803	0.953	55,873,736	572	130.19	10.02

InceptionResNetV2 結合了 Inception Module 與 ResNet 提出的殘差直連通路，使用極深的網路架構時，遭遇梯度彌散的可能性降低，模型更容易收斂，並提升模型準確率

4. 附加檔案解釋

我們總共附件了 4 個檔案，有點多不好意思 ，因為我們常常跑一跑就被踢出來，所以有些步驟就分開寫分開跑。

第一個是 ResnetV2_original，這個檔案跑的是我們凍住的模型，跑完後就會丟到 ResnetV2_mod 裡面跑，這部分就是前面提到我們把 base model 改成 trainable 的部分。接著其他兩個是 pseudo labeling 的 code，第一個是 ResnetV2_unlabel_predict，這個檔案是在預測沒有 label 的模型，其中設定門檻 85%，而且最後會輸出一個包含原本有 label 及後來我們預測 label 的 csv 檔讓我們訓練用。最後就是 pseudo label 的訓練，其實就是把 ResnetV2_mod 和 ResnetV2_original 合在一起，吃進剛剛產生的 csv 而已。