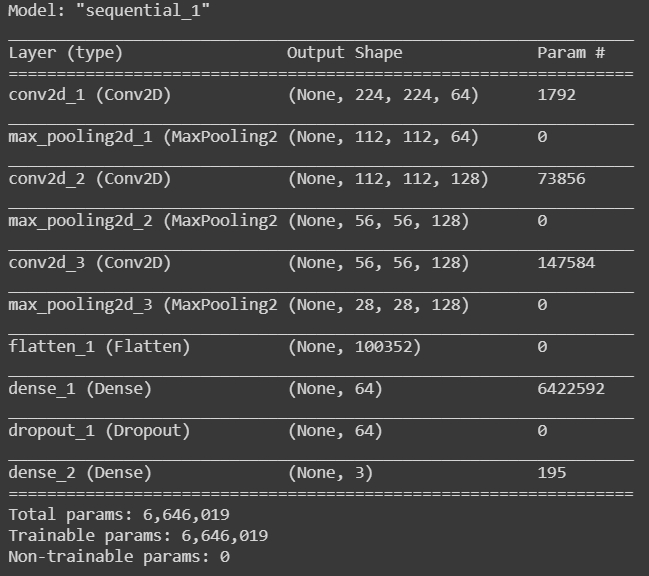
**HW4\_E14056114\_徐顯舜**

1. Model(Architecture):



由於我們的資料集相對於其他如cifar-10等模型，我們5600筆資料可說是small data。因此使用深且具有大量神經元數量的模型可能會造成過度擬合且訓練過程及緩長等問題。因此最後我們決定先使用手刻模型。

**參考資料:**[Building powerful image classification models using very little data](https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html)

文中的模型主要使用三個keras提供的function:conv2D，maxpoolng2D，Flatten、Dense以及Dropout ，我們的圖片輸入大小為224\*224\*3

首先我們先建立我們的卷積層去找出我們圖形中的特徵，一開始我們的filter數量為64(會得到有64層的feature map)，kernel size為3\*3矩陣(取3\*3是希望能擷取出較細微的特徵)，接者使用maxpooling(strides = (2,2))將我們的尺寸減半(在kernel size裡面取最大值，通常物理意義上是指比較明顯的特徵值)，這樣子連續做三層，其中的差別為第二與第三層的filter數量為128(算是得到比較好效果的組合)，因為我們的芒果等級只有三種，加上他們都是芒果(大多特徵值都差不多)，因此如果filter太多分太細反而容易造成過度擬和。

接著再使用平坦層將我們所有的資料轉為1維矩陣，再通過hidden layer將結果轉為64個值，並丟棄20%神經元去防止我們過度擬。這邊注意Dense的值也不宜太大，因為我們的本身相異的特徵值不多，訓練資料量也不大，因此dense太多會太勉強，反而對模型不好，最後我們選擇使用64。

最後在dense為3(3種ranking)，我們使用softmax使用機率的方式去決定它是屬於哪種等級。

2. Preprocessing：

i. Method:

1. 圖片去背：

導入Detectron2的物件偵測技術，以提取芒果圖片進行訓練，目的降低原本圖片中背景產生干擾訊號之影響。

在Detectron2的原始預測函式中，輸出結果為原圖並繪出boundary-box、類別名稱及預測機率，以及不同顏色之遮罩。為符合本次實作需要，本組改寫了detectron2.utils.visualizer中的draw\_instance\_predictions，判定所有預測物件其一是否符合預期類別，並僅保留「最大boundary-box」的物件，使用其遮罩及opencv的bitwise函式進行去背處理，回傳僅有一乾淨去背芒果的圖片。

由於Detectron2預設模型是以COCO資料集進行訓練，然而其中並沒有Mango類別，而本次經測試過後發現apple及donut兩類別對於外型預測相當接近，礙於時間因素，因此先行使用這兩個類別進行物件及輪廓偵測。

1. 圖片壓縮：

使用keras.preprocessing.image中的load\_img函式圖片壓縮，將原始圖片壓縮成224x224的正方形圖片，以統一經去背處理後產生的不同長寬大小。

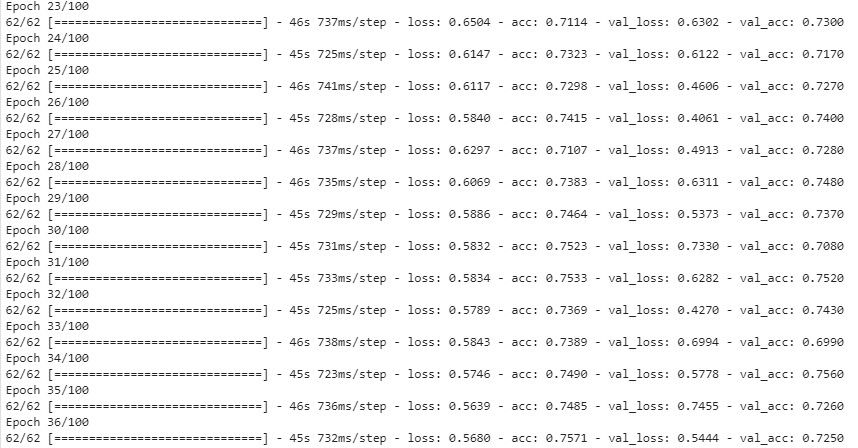
1. 資料集製作：

本次訓練用圖片共5600張，經去背並篩選掉不可用/偵測失敗的照片後剩約5150張，因此本組將train/validation資料集大小訂為4000/1000張，並使用python中的random.sample函式，確保資料輸入的隨機性。

ii. Example:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 檔名 | 00030.jpg | 00132.jpg | 00403.jpg | 00719.jpg |
| 處理結果 |  |  |  |  |

3.Result：



* 1. Evaluation with training dataset：

Train Loss: 0.5410

Train Accuracy: 0.718

* 1. Evaluation with testing dataset：

Test Loss: 0.4791

Test Accuracy: 0.769

* 1. Confusion matrix：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 layer 手刻 | | Predict | | |
| A | B | C |
| LABEL | A | **174** | 61 | 8 |
| B | 37 | **213** | 43 |
| C | 11 | 40 | **213** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BatchNormalization | | Predict | | |
| A | B | C |
| LABEL | A | **185** | 45 | 9 |
| B | 53 | **209** | 31 |
| C | 16 | 57 | **191** |

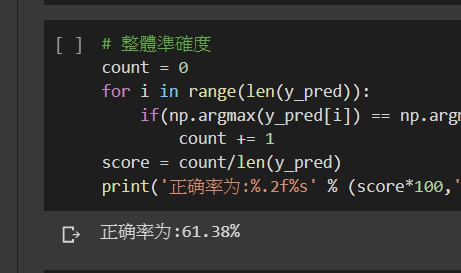
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 layer 手刻 | | Predict | | |
| A | B | C |
| LABEL | A | **198** | 36 | 9 |
| B | 49 | **215** | 29 |
| C | 15 | 62 | **187** |

4.個人工作報告

我主要是負責整理前處理的code(不是去背那塊，而是csv讀檔以及轉乘rgb等那邊)，以及嘗試看看不同的模型。

一開始我使用的是transfer learning，然後挑的模型是VGG16，但由於他的模型資料量太大，都要跑超過12小時，因此無法實測出能達到準確率多少，但訓練過程最高大約在62%(大多維持50%上下)。因此後來使用資料量比較小的InceptionV3和ResNet50v2，這兩個model速度就比VGG16還快，一個epoch大約8-10分鐘，出來的結果大約55%。

後來查詢網路文章，看到有人將兩個model merge再一起，因此我將InceptionV3和ResNetv2合併再一起，再做dropout避免過擬和，並使用去背後的資料，出來的結果稍微好一點，大約62%，也嘗試了比如正則化和layer凍結等方式，但也沒有特別好轉。最後看了一篇文章([為什麼深度學習模型準確率不會提昇?](https://medium.com/life-of-small-data-engineer/%E7%82%BA%E4%BB%80%E9%BA%BC%E6%B7%B1%E5%BA%A6%E5%AD%B8%E7%BF%92%E6%A8%A1%E5%9E%8B%E6%BA%96%E7%A2%BA%E7%8E%87%E4%B8%8D%E6%9C%83%E6%8F%90%E6%98%87-f6445ef7ae47))和有做的同學討論，決定使用手刻看看。

最後反而準確率更為提高，有可能我們的data真的太少，以及要找出他們的差異性可能要從不同的方法下手，可能去看convolution出的結果這些照片的差異?或者是去使用GridSearch去找出最佳化的參數，之後可能就要請教助教。