機器學習技法 期末報告

組名：量子糾纏的愛情
組員：b06705002 黃啟宏
b06705013 黃奕滔
b06705018 朱家儀

問題描述(Problem discription and overview)

愛文芒果採收後依品質篩選可以分為 A、B、C 三等級，依序為出口用、內銷用、加工用。然而愛文芒果依靠人工篩選有約 10% 的誤差。我們需要透過建立影像辨識演算法模型，對愛文芒果影像進行三種等級分類。

我們在開始此項專題前，先預想了以下幾種方法訓練模型：

1. 透過調整參數及層數（Fine-tune）找出最適合的CNN模型。
   優點：若能找到適合次CNN模型的參數及層數，可使準確率提升非常多。
   缺點：需要花費許多時間進行調整，也未必能找到最佳模型。
2. 利用HOG訓練SVM模型，透過計算和統計圖像局部區域的梯度方向直方圖來構成特徵，並透過Grid search找到最佳的模型。
   優點：HOG適合用於影像辨識。
   缺點：準確率可能不比複雜的模型來的高。
3. 套用網路上提供的模型套件，例如ResNet50。
   優點：不需要自己堆疊層。
   缺點：不能保證芒果的訓練和模型的適合度。

資料蒐集(Data gathering)

從人工智慧共創平台的競賽專區中取得三種資料集，分別為 ：

A. 訓練集（Training Data）
B. 開發集（Development Data）
C. 測試集（Test Data）

訓練和開發資料集均包括 1. 影像圖檔（Image）和 2. 影像對應之等級標籤（Grade），而測試資料集只包含影像圖檔，在上傳模型結果後可以得到此模型對測試資料集的準確率。

資料預處理(Data preparation)

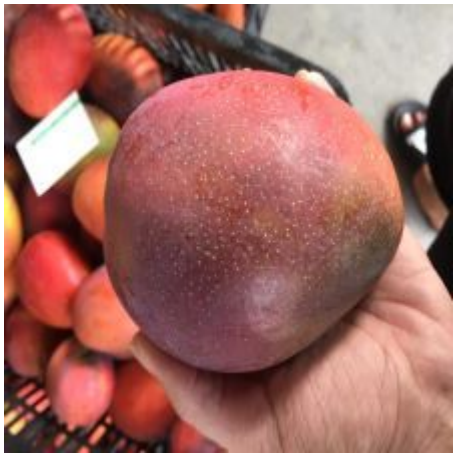
1. **資料清理(Data cleaning)**

   眾多芒果圖中，有一些圖片在拍照的時候因為沒有對焦到而模糊，例如05369.jpg、05201.jpg、03226.jpg都是非常不清楚的圖片，於是被刪除了。

2. 特徵提取與處理(Feature selection and engineering)

主辦方提供的資料集圖片中，有各種解析度的影像，這些不同影像在丟進模型訓練時會有兩個問題：(1)特徵數量不一致 (2)特徵數量過多。前者有機率可以用不同的模型解決，但後者會造成運算資源不足或是過適(overfitting)的現象，所以我們進行圖片的解析度調整(resize)，依照慣例線性調整成224*224的大小。

除此之外，我們也發現圖檔的背景雜亂不一，有時候也會出現背景裡有其他一整箱芒果的狀況(如：00242.jpg)，這些不重要的資訊都不應該是model作為判斷的基礎，所以我們進行了「圈選芒果」的前置作業。圈選芒果的方法又選擇了opencv的circle object detection，使用CV_HOUGH_GRADIENT算法偵測，偵測到芒果以外的部分則會被黑色填滿。



因此資料集被分成兩組：

(1) 解析度調整並圈選芒果

(2)僅解析度調整

這兩組圖片都會被拿來訓練模型，並比較何者表現較佳。

而在SVM中，我們首先透過rgb2grey()將圖片轉為灰階，使得圖片的shape由(224,224,3)轉為(224,224,1)。

接下來，使用HOG，透過計算和統計圖像局部區域的梯度方向直方圖來構成特徵，由於HOG是在圖像的局部方格單元上操作，所以對圖像幾何的和光學的形變都能保持很好的不變性。

最後，再將三維陣列轉為一維陣列，即放入SVM模型中。

3. **特徵標準化(Data standardize)**

   我們把圖片的色彩數值同除以255作為scaling。

4. **資料增強(Data augmentation)**

   在處理資料時，我們發現訓練資料集中只有 5600 筆資料。為了確保在訓練時不會產生 over-fitting 的現象，我們藉由旋轉、調整大小、比例尺寸、改變亮度色溫、翻轉等等變形處裡的方法，創造出更多的圖片來讓機器學習，彌補資料量不足的困擾。在下圖中，我們利用 keras 的 ImageDataGenerator 實作資料增強。

```
from tensorflow.python.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rotation_range=40,          ← 旋轉角度
                                   width_shift_range=0.2,      ← 上下左右平移
                                   height_shift_range=0.2,
                                   shear_range=0.2,            ← 逆時針方向剪切
                                   zoom_range=0.2,             ← 放大縮小
                                   channel_shift_range=10,     ← 改變顏色
                                   horizontal_flip=True,       ← 水平翻轉
                                   fill_mode='nearest')        ← 對缺失進行補全
```

列出候選模型並選出最佳配適模型(Select best model from candidates)

1. **CNN**

   在特別做超參數的調整之前，使用train_test_split = 0.1, monitor = val_loss, patient = 5，我們以上述的兩組資料集：(1)解析度調整並圈選芒果 (2)僅解析度調整，作為訓練集，得到的正確率依序分別為64.11%及!!!!!!!!。

   出乎意料的，有圈選芒果的資料集比起沒有圈選的表現得還差，我們進行了各種操作變因是圈選與否的實驗，確定了在一般NN下，普遍經過圈選預處理的結果都比未處理的資料還差。

   對此，我們的猜想是：因為NN的複雜度本身就足夠model找出圖片中的芒果，而且這個找出芒果的準確度甚至高過於我們利用HOUGH_CIRCLE_GRADIENT得到的芒果輪廓，所以其實和網路上分享的經驗類似，通常選擇一個合適的model，以及控管好複雜度，這些機器學習演算法本身就會幫我們更細緻的處理好資料預處理的問題，因而得到更好的結果。

2. SVM

   在第一次SVM中，可以得到63%的準確度。

```
sgd_clf = SGDClassifier(random_state=42, max_iter=1000, tol=1e-3)
sgd_clf.fit(x_train, y_train)

SGDClassifier(alpha=0.0001, average=False, class_weight=None,
              early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
              l1_ratio=0.15, learning_rate='optimal', loss='hinge',
              max_iter=1000, n_iter_no_change=5, n_jobs=None, penalty='l2',
              power_t=0.5, random_state=42, shuffle=True, tol=0.001,
              validation_fraction=0.1, verbose=0, warm_start=False)
```

3. ResNet50

隨着net的加深，有可能會出現訓練資料集的準確率下降的現象。Resnet提供了兩種路徑選擇方式，分別為 identity mapping 和 residual mapping。如果net 已經到達最優，繼續加深網絡 residual mapping 將被更新爲 0，只剩下 identity mapping。如此一來，net會一直處於最優狀態，準確率便不會隨着深度的增加而降低。

在此，我們套用網路上已經建立好的ResNet50模型，作為上述兩種方法CNN和SVM的對照組。在此模型中，我們套用 keras 提供的ResNet50套件進行訓練。由於設備規格不足，在訓練模型時我們將前幾層凍結，以免我們的設備跑不動。下圖為epoch為20、batch為1400的訓練過程截圖：

```
net_final.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

print(net_final.summary())
```

```
conv5_block3_3_conv (Conv2D)    (None, 7, 7, 2048)   1050624     conv5_block3_2_relu[0][0]
__________________________________________________________________________________________________
conv5_block3_3_bn (BatchNormali (None, 7, 7, 2048)   8192        conv5_block3_3_conv[0][0]
__________________________________________________________________________________________________
conv5_block3_add (Add)          (None, 7, 7, 2048)   0           conv5_block2_out[0][0]
                                                                 conv5_block3_3_bn[0][0]
__________________________________________________________________________________________________
conv5_block3_out (Activation)   (None, 7, 7, 2048)   0           conv5_block3_add[0][0]
__________________________________________________________________________________________________
flatten (Flatten)               (None, 100352)       0           conv5_block3_out[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 100352)       0           flatten[0][0]
__________________________________________________________________________________________________
softmax (Dense)                 (None, 3)            301059      dropout[0][0]
==================================================================================================
Total params: 23,888,771
Trainable params: 23,826,179
Non-trainable params: 62,592
__________________________________________________________________________________________________
None
```

```
Epoch 17/20
1400/1400 [==============================] - 4266s 3s/step - loss: 0.6248 - accuracy: 0.7259 - val_loss: 0.5761 - val_accuracy:
0.7700
Epoch 18/20
1400/1400 [==============================] - 4101s 3s/step - loss: 0.6216 - accuracy: 0.7261 - val_loss: 0.5112 - val_accuracy:
0.7887
Epoch 19/20
1400/1400 [==============================] - 3989s 3s/step - loss: 0.6078 - accuracy: 0.7304 - val_loss: 0.5950 - val_accuracy:
0.7312
Epoch 20/20
1400/1400 [==============================] - 4007s 3s/step - loss: 0.5975 - accuracy: 0.7341 - val_loss: 0.5464 - val_accuracy:
0.7675
```

超參數選擇(Tuning process)

1. CNN

batch_size	monitor (early stopping)	patient (early stopping)	accuracy

3	val_loss	5	51%
4	val_loss	5	64%
4	val_loss	7	53%
4	val_loss	6	48%
4	val_loss	4	58%
4	val_acc	5	59%
4	val_acc	4	52%
4	val_acc	7	54%
3	val_acc	3	57%

```
[ ]  y_pred1 = model.predict(xt)
     count = 0
     for i in range(len(y_pred1)):
         if(np.argmax(y_pred1[i]) == np.argmax(yt[i])): #argmax函数找到最大值的索引，即为其类别
             count += 1
     score = count/len(y_pred1)
     print('正确率为:%.2f%s' % (score*100,'%'))

     正确率为:64.11%
```

2. SVM

針對SVM，我們使用GridSearchCV而不是手動修改參數。首先，我們嘗試改進HOGTransformer。GridSearchCV將檢查每個字典中的所有組合，因此我們每個字典有2個，總共4個。由於最佳的預處理會隨模型而變化，因此透過搜索他們的網格來找到最佳解。使用CV = 3，設置交叉驗證，將數據集分為多個部分（在這種情況下為3個），並且完成了多次訓練。在每一輪中，一折用於驗證，另一折用於訓練。如此可以在不影響測試數據的情況下針對一部分訓練數據來驗證和改進模型。對於分數則使用準確性。

```
grid_search = GridSearchCV(HOG_pipeline,
                           param_grid,
                           cv=3,
                           n_jobs=1,
                           scoring='accuracy',
                           verbose=1,
                           return_train_score=True)

grid_res = grid_search.fit(x_train, y_train)
```

```
Fitting 3 folds for each of 4 candidates, totalling 12 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  12 out of  12 | elapsed: 46.5min finished
```

最後，獲得65%的準確度。

ResNet50

由列出候選模型並選出最佳配適模型中ResNet50的截圖可以知道，每個epoch都需要超過1個小時的時間。由於比賽時間的壓力，在此模型中我們沒有進行 tuning，而是先將參數都先預想好才開始進行訓練，得出理論上較佳的結

果。我們將此訓練結果輸出為h5模型檔後，對開發資料集(Dev)計算準確率。準確率如下：

```
# print(D_TL)
D_L = np.array(dev_list)
# print(D_L)
print(error_rate(D_L, D_TL))
print(1-error_rate(D_L, D_TL))
```

```
0.27125
0.72875

Accuracy:
0.72875
```

結論(Conclusion)

首先，ResNet50比起一般我們建的CNN模型和SVM，表現的準確率高了將近10%，所以我們選擇它作為最佳模型。可惜的是ResNet的訓練時間實在是太長了，在我們的個人電腦上甚至訓練一組就要花費二十餘小時，所以不太可能做Tuning，否則正確率應該會更高一些。

References:

Machine Vision based Fruit Classification and Grading - A Review
https://www.researchgate.net/publication/318486455_Machine_Vision_based_Fruit_Classification_and_Grading_-_A_Review
Hands-On_Machine_Appendix_B
https://tdgunes.com/COMP6246-2019Fall/lab1/extra1_3.pdf
Hough Circle Transform — OpenCV-Python Tutorials 1 documentation
https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_houghcircles/py_houghcircles.html
Data augmentation
https://chtseng.wordpress.com/2017/11/11/data-augmentation-%E8%B3%87%E6%96%99%E5%A2%9E%E5%BC%B7/
ResNet50 introduction
https://www.cnblogs.com/bonelee/p/8972139.html
ResNet50
https://blog.gtwang.org/programming/keras-resnet-50-pre-trained-model-build-dogs-cats-image-classification-system/
SVM
https://rpubs.com/Sharon_1684/454441
https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python