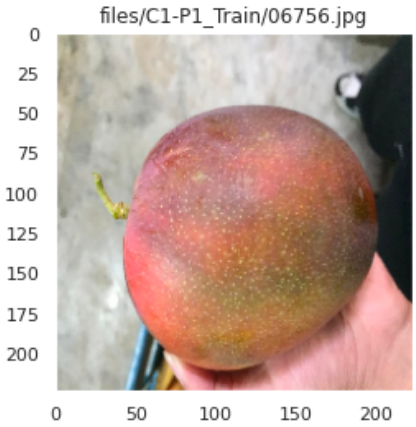
Machine Learning Techniques Final Project

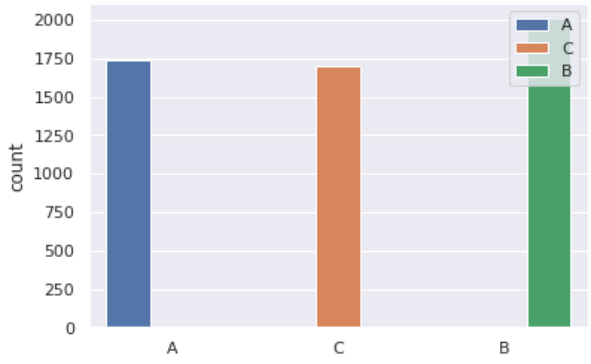
1. Data description & preprocess

這次的芒果辨識比賽中，主要目的為將芒果的圖像辨識為A, B, C三個分類其中之一。主辦單位提供的Data分為training set, dev set, test set，數量分別為5600張、800張、1600張，大小各異，約為1000x1000左右、採用RGB顯示，其中只有training和dev有提供正確的label，而test僅在開放上傳的幾天內提供下載。我們在這次的比賽中，主要使用training set作為訓練用，dev set作為validation使用。

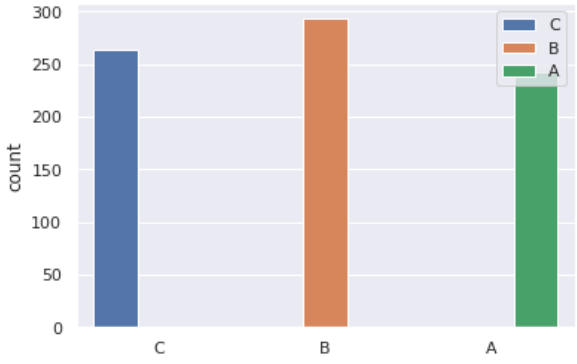
在一開始，我們選用主辦單位提供的教學影片中的設置，將圖片壓縮至800x800訓練，但我們很快地發現，800x800似乎過大，僅僅300張訓練時便會將12G的RAM塞滿，於是我們進行壓縮的實驗，我們使用100x100後，發現圖片似乎損失過多訊息（如下左圖），我們在參考了其他pretrained model後，發現最常見的224x224效果最好，因此我們決定將所有圖片壓縮為224x224（如下右圖）



並將label由A, B, C的表示法改為one-hot encoding，三種label在training和dev set上的分佈相當平均（見圖(1)、圖(2)），因此我們選擇將Accuracy作為選擇模型的指標。



圖(1)



圖(2)

2. Model

我們在這次的實驗中選擇了三種model，分別為：

1. 自行架設的CNN
2. Extra Trees Classifier
3. VGG16 pre-trained model w/ fine tuning

model/index	Efficiency	Scalability	Popularity	Interpretability
CNN	中	高	高	中
Extra trees	高	中	高	中
VGG16	低	高	高	低

在我們初步的猜想中，三種model的表現大概為：

因此，我們決定先使用自行架設的CNN進行實驗。

(1)CNN

在CNN上，我們選擇實作最為快速容易的Keras package，搭配Tensorflow作為後端，在Google colab上使用GPU進行訓練，在batch size = 16的情況下，一個epoch大概需要花7秒左右，一共訓練100個epoch。

我們一開始使用該比賽教學所提供的CNN模型作為基礎，在一開始，我們準度約為69%左右，我們嘗試各種改進方法，如添加、減少層數與神經元等等。

我們觀察到，validation set跟training set的accuracy不同步，代表模型可能有overfitting的現象，因此我們試著添加Batch Normalization、Dropout等，我們發現，不一定越多層數、神經元越好，但Drop out和Batch Normalization的使用有時會導致模型不收斂，最後我們得到的模型結構如右圖：

一共有三層卷積層，window大小為(3, 3)，神經元數量每層分別為32, 32, 64，兩層全連接層的神經元數量分別為64, 3，最後一層Activation為softmax，輸出三種類別的機率。

我們的Loss採用categorical cross entropy，Optimizer我們使用Adam，learning rate為0.001，最後在validation得到約71.9%的準確度。

```
18/18 [==============================]
[0.6411421298980713, 0.71875]
```

在最後，模型仍有overfitting，我們決定試用下一個模型。

```
Layer (type)                 Output Shape
=================================================
conv2d_10 (Conv2D)           (None, 222, 222, 32)
activation_16 (Activation)   (None, 222, 222, 32)
max_pooling2d_10 (MaxPooling (None, 111, 111, 32)
conv2d_11 (Conv2D)           (None, 109, 109, 32)
activation_17 (Activation)   (None, 109, 109, 32)
max_pooling2d_11 (MaxPooling (None, 54, 54, 32)
conv2d_12 (Conv2D)           (None, 52, 52, 64)
activation_18 (Activation)   (None, 52, 52, 64)
max_pooling2d_12 (MaxPooling (None, 26, 26, 64)
flatten_4 (Flatten)          (None, 43264)
dense_7 (Dense)              (None, 64)
activation_19 (Activation)   (None, 64)
dropout_4 (Dropout)          (None, 64)
dense_8 (Dense)              (None, 3)
activation_20 (Activation)   (None, 3)
=================================================
Total params: 2,797,795
Trainable params: 2,797,795
Non-trainable params: 0
```

```
loss: 1.0998 - acc: 0.3591 - val_loss: 1.0944 - val_acc: 0.3772
loss: 1.0622 - acc: 0.4255 - val_loss: 1.0517 - val_acc: 0.4062
loss: 0.9919 - acc: 0.4947 - val_loss: 0.9600 - val_acc: 0.4978
loss: 0.9488 - acc: 0.5215 - val_loss: 0.9834 - val_acc: 0.4799
loss: 0.8566 - acc: 0.5985 - val_loss: 0.7902 - val_acc: 0.6172
loss: 0.7292 - acc: 0.6716 - val_loss: 0.7116 - val_acc: 0.6797
loss: 0.6571 - acc: 0.7115 - val_loss: 0.7298 - val_acc: 0.6596
loss: 0.6115 - acc: 0.7397 - val_loss: 0.6550 - val_acc: 0.6953
loss: 0.5775 - acc: 0.7567 - val_loss: 0.6367 - val_acc: 0.7065
loss: 0.5421 - acc: 0.7567 - val_loss: 0.6761 - val_acc: 0.6953
loss: 0.5237 - acc: 0.7759 - val_loss: 0.6811 - val_acc: 0.6853
loss: 0.4660 - acc: 0.8011 - val_loss: 0.9094 - val_acc: 0.6283
loss: 0.4434 - acc: 0.8055 - val_loss: 0.8145 - val_acc: 0.6875
loss: 0.4176 - acc: 0.8284 - val_loss: 0.6506 - val_acc: 0.7009
```

(2)Extra Trees Classifier

之後我們選擇使用非Neural network的模型，在現在較為受歡迎的模型中(SVM, Adaboost, Gradient boost, Random forest等)，我們選擇使用Random forest，其中，我們使用其變化版Extra Trees，不使用bootstrap，並把分割feature的點從最佳點改為隨機選擇。

我們選擇sci-kit learn中提供的Extra trees的實作，把壓縮至224x224的training set跟dev做flatten之後，進行訓練，大概需要47秒，較CNN快速許多。

```
[ ] extra_tree = ExtraTreesClassifier(n_jobs = -1, verbose = 1)
    extra_tree.fit(X_train.reshape(5439, -1), y_train)

    [Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 2 concurrent workers.
    [Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:   22.0s
    [Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:   47.9s finished
```

在dev set上，我們得到約67.7%的準確率，稍微差強人意。

```
0.6767857142857143
```

(3)VGG16

第三個Model，我們選擇使用pre-trained的VGG16抓取特徵，並進行fine tuning，我們使用Keras中的VGG16模型，包含使用imagenet訓練的卷積層，最後的top model則自己訓練。輸出的

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
dropout_1 (Dropout)          (None, 25088)             0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               6422784
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
batch_normalization_1 (Batch (None, 256)               1024
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 771
=================================================================
Total params: 6,424,579
Trainable params: 6,424,067
Non-trainable params: 512
```

activation一樣採用三個輸出的softmax，optimizer則使用SGD配合較低的learning rate(0.0001)，並在Dense層中加入kernel regularizer，避免overfitting。在測試各種batch normalization、drop out後，得到的top model如下：
Batch size, epoch數依然設為16, 100，可惜的是，模型依然有overfitting的現象，在dev set上只得到71.6%的準確率

```
0.7160714268684387
```

3. Data augmentation

實驗過後，我們認為，提升模型複雜度較為不可行，可能引發更強烈的overfitting，在觀察了資料圖片之後，我們認為可能是因為資料較為不足，因此我們決定採用Data augmentation的方法。

我們使用Keras提供的ImageDataGenerator，並使用了旋轉、左右上下移動、縮放、左右翻轉等功能，觀察產生的資料後，我們認為芒果的特徵並沒有受到破壞，因此我們將經過augmentation的data用作training。

(1) CNN

在剛剛使用的CNN模型上，我們使用經過augmentation的training set進行訓練，在經過100個epoch後，得到了約74.6%的準確率，可說是大大提高，然而，可能因為需要額外運算的原因，每個epoch的運算時間增加到了60秒左右，訓練效率降低了將近10倍，但training set的accuracy已經不再跟validation不同步，代表overfitting的現象較為減緩。

```
0.7464285492897034
```

(2) Extra trees classifier

Extra trees上無法使用Keras的ImageDataGenerator，且由於表現較不佳的原因，我們選擇不使用。

(3) VGG16

在VGG16上使用經過augmentation的training set進行訓練，由於圖片為generator隨機產生，因此無法先輸入VGG16再進行Top model的訓練，因此我們複製了VGG16卷積層的結構與

weight，並與top model結合為完整的model，再將卷積層的權重凍結，然而，資料仍需要在每次訓練時經過每一層，因此仍會減慢我們訓練的efficiency。

在batch size = 16，epochs = 100的訓練後，我們也得到了74.6%的準確率，然而，每個epoch的訓練時間拉長為80秒左右，但是，我們的training accuracy跟validation accuracy幾乎同步，因此我們認為我們已經大幅的降低了overfitting的問題。

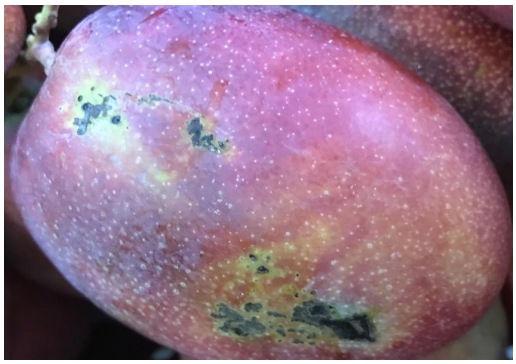
```
0.7446428537368774
```

4. Object detection

在觀察我們的資料集後，我們發現圖片含有大量的noise，照片中的手、背景光影容易被辨認為芒果的一部份，背景的黑色籃子可能被認為是黑斑等等。我們假設準確率無法進一步提升的原因是data的noise，因此，我們決定透過object detection的方法，偵測並裁減出原始照片上的芒果，再將裁減完的照片進行壓縮。

我們首先把training data的前300張照片做手動標記，並將這300張標記後的照片使用imageai這個套件在YOLOv3的model中train，最後成功做出可以精準辨識芒果位置的模型。如果照片中辨識出超過一顆芒果，程式會自動裁切出x軸最靠近中心的那顆。

原始圖片



經過模型裁切出的芒果

(1) CNN

在使用了裁減過的照片，並將照片進行augmentation後，我們嘗試各種不同的optimizer及learning rate，但模型仍無法收斂，在時間壓力下，我們只能放棄該模型並繼續嘗試其他模型。

(2) Extra trees classifier

將裁減並flatten過的照片進行訓練後，我們得到65.9%的準確率，較沒有裁減時為低，因此我們繼續實驗下個模型。

```
extra_tree.score(X_dev.reshape(800, -1), y_dev)

[Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done  46 tasks      | elapsed:    0.0s
[Parallel(n_jobs=2)]: Done 100 out of 100 | elapsed:    0.1s finished
0.65875
```

(3) VGG16

在使用了裁減過的照片，並將照片進行augmentation後，我們一樣使用batch size = 16, epochs = 100進行訓練，最後得到78.5%的結果，也沒有遭遇overfitting。

```
0.7850000262260437
```

5. Conclusion

我們將Accuracy化為表格：

最後，在準確率上由VGG16搭配Data augmentation及Object detection為最高。

Model/Data preprocessing	None	Data augmentation	Object detection
自架CNN	71.9%	74.6%	Not converge
Extra Trees	67.7%	N/A	65.9%
VGG16	71.6%	74.5%	78.5%

四個面向的表格修正為如下，其中去除Popularity，加入Accuracy：

model/index	Efficiency	Scalability	Interpretability	Accuracy	總分
CNN	中低	低	中	中高	7
Extra trees	高	高	中	低	9
VGG16	低	中	中	高	8

低為1分，中低為1.5分，以此類推。雖然Extra trees總分為最高，然而，該比賽只以準確率為指標，因此我們最為推薦VGG16搭配Data augmentation及Object detection。

6. Reference

Keras documentation:
https://keras.io/zh/
Extra tree classifier:
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html
官方教學:
https://www.youtube.com/playlist?list=PLJ6QzDAugy1muFIHX17go-OR62avvWr1A
VGG16 fine tuning:
https://codertw.com/程式語言/578317/
https://blog.csdn.net/akadiao/article/details/80532863
Data augmentation:
https://chtseng.wordpress.com/2017/11/11/data-augmentation-資料增強/
https://zhuanlan.zhihu.com/p/30197320
imageai:
https://imageai.readthedocs.io/en/latest/

7. Team member

B06705034 吳禹辰：Model building, training
B06705049 王松億：Data preprocessing
B06705029 黃榮豐：Method researching