資料科學概論期末專題

生機四 易峻葦 植微四 蔡宜豐

• 目標

利用Pytorch搭建出CNN網路辨識課程所提供的九種不同生長天數的鳳梨型態，每種天數的資料為200張圖片，總共有1800張圖片。

Stage501 Stage505 Stage509 Stage601 Stage605

Stage609 Stage701 Stage705 Stage709

• 資料處理

由於資料僅有1800筆，因此我們將圖片輸入進網路時有多一個處理圖片以增加資料多樣性的步驟，程式碼如下:

```
train_tfm = transforms.Compose([
        # TODO: Add data augmentation for training data
        transforms.Resize((224, 224)),
        transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)),
        transforms.RandomGrayscale(p=0.5),
        transforms.RandomHorizontalFlip(p=0.5),
        transforms.RandomVerticalFlip(p=0.5),
        transforms.RandomRotation((0,180)),
        transforms.RandomAffine(degrees=(-45, 45), translate=(0, 0.2)),
        transforms.ToTensor(),

])

# No need augmentation on validation data
test_tfm = transforms.Compose([
        #transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)),
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
])
```

Normalize(mean, std) : output[channel] =(input[channel]-mean[channel])/std[channel)

RandomGrayscale : 以50%的機率將圖片轉成灰階圖

RandomHorizontalFlip : 以50%的機率將圖片水平翻轉

RandomVerticalFlip : 以50%的機率將圖片垂直翻轉

RandomRotation : 將圖片隨機轉一個0度~180度的角度

RandomAffine : 將圖片隨機旋轉-45度~45度，以及線性平移(0~0.2)

- 給定模型參數
  1. Batchsize : 64
  2. Epoch : 60
  3. Learning rate : 0.0003× $0.97^{(epoch\ num)}$，會選擇0.97是因為一開始我們設定的0.9或者0.95皆會使learning rate 衰減太快，影響模型學習的能力。我們透過Learning rate scheduling讓model較容易收斂。
  4. Criterion : Cross Entropy Loss
  5. Weight decay (L2 regularization): 0.0001
  6. Optimizer: Adam
- 模型建置
  1. Model 1

     Convolution Layer

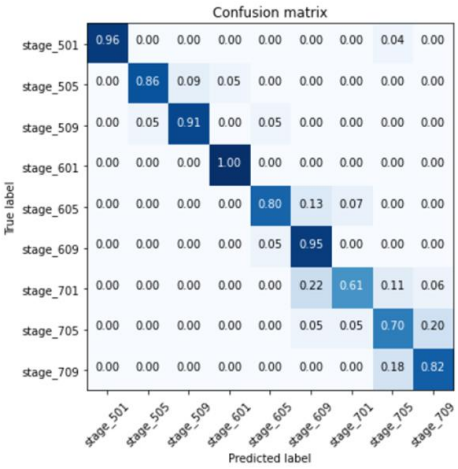
```
nn.Conv2d(3,  64,  3,  1,  1),#output  shape(64,  224,  224 )
nn.Conv2d(64,  64,  3,  1  ,1),
nn.BatchNorm2d(64),
nn.ReLU(),
nn.MaxPool2d(2,  2,  0),#output  shape(64,  112,  112 )

nn.Conv2d(64,  128,  3,  1,  1),#output  shape(128,  112,  112
nn.Conv2d(128,  128,  3,  1,  1),
nn.BatchNorm2d(128),
nn.ReLU(),
nn.MaxPool2d(2,  2,  0),#output  shape(128,  56,  56 )

nn.Conv2d(128,  256,  3,  1,  1),#output  shape(256,  56  56 )
nn.Conv2d(256,  256,  3,  1,  1),
nn.BatchNorm2d(256),
nn.ReLU(),
nn.MaxPool2d(2,  2,  0),#output  shape(256,  28,  28 )

nn.Conv2d(256,256,  3,  1,  1),#output  shape(128,  28,  28 )
nn.Conv2d(256,512,  3,  1,  1),
nn.BatchNorm2d(512),
nn.ReLU(),
nn.MaxPool2d(2,  2,  0),#output  shape(256,  14,  14 )
```

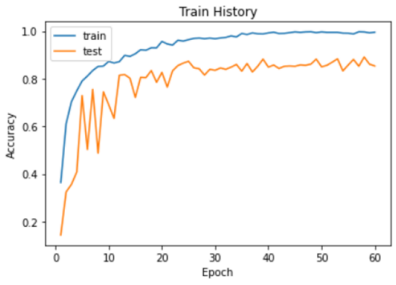
     Fully connected Layer

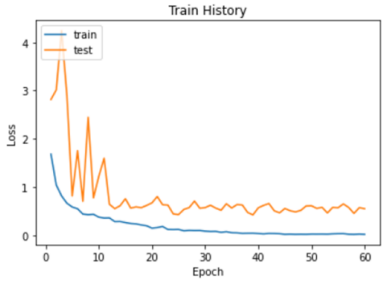
```
nn.Linear(512 * 14 * 14, 256),
torch.nn.Dropout(0.4),
nn.ReLU(),
nn.Linear(256, 128),
nn.ReLU(),
nn.Linear(128, 64),
nn.ReLU(),
nn.Linear(64, 32),
nn.ReLU(),
nn.Linear(32, 9),
```

該模型的特點為每做兩次卷積後才進行一次池化，每次池化是取2乘2的範圍，並且還有取Batch Normalization。

- ❖ 成果:







從confusion matrix 以及 accurarcy 和 loss 的圖表可知，該模型的準確率並沒有太理想，大概只有80%左右。所以接下來我們嘗試先不要將模型建得太複雜，將原本兩兩一組的卷積層拿掉一層，形成Model 2，其餘的參數則是保持一致。

2.Model 2

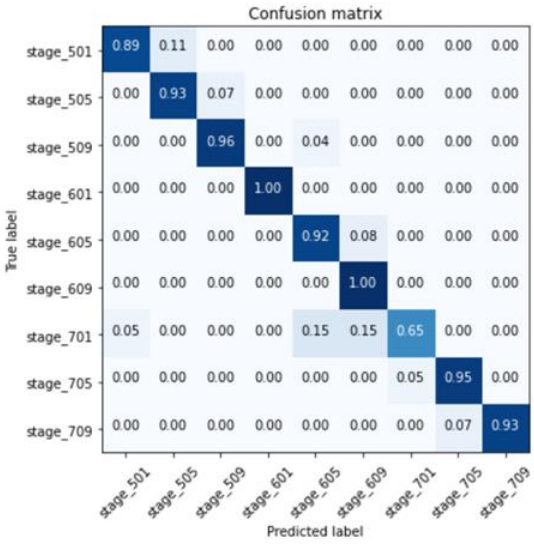
Convolution Layer

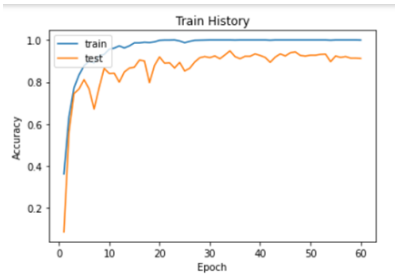
```
nn.Conv2d(3,  64,  3,  1,  1),#output  shape(64,  224,  224  )
nn.BatchNorm2d(64),
nn.ReLU(),
nn.MaxPool2d(2,  2,  0),#output  shape(64,  112,  112  )

nn.Conv2d(64,  128,  3,  1,  1),#output  shape(128,  112,  112  )
nn.BatchNorm2d(128),
nn.ReLU(),
nn.MaxPool2d(2,  2,  0),#output  shape(128,  56,  56  )

nn.Conv2d(128,  256,  3,  1,  1),#output  shape(256,  56  56  )
nn.BatchNorm2d(256),
nn.ReLU(),
nn.MaxPool2d(2,  2,  0),#output  shape(256,  28,  28  )

nn.Conv2d(256,256,  3,  1,  1),#output  shape(128,  28,  28  )
nn.BatchNorm2d(256),
nn.ReLU(),
nn.MaxPool2d(2,  2,  0),#output  shape(256,  14,  14  )

nn.Conv2d(256,512,  3,  1,  1),#output  shape(128,  28,  28  )
nn.BatchNorm2d(512),
nn.ReLU(),
nn.MaxPool2d(2,  2,  0),#output  shape(512,  7,  7  )
```

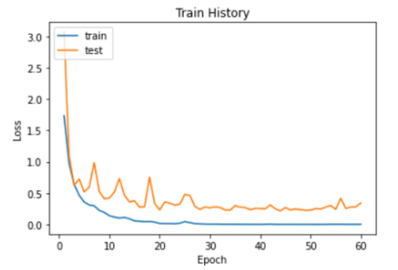
Fully connected Layer

```
nn.Linear(512  *  7  *  7,  256),
nn.ReLU(),
nn.Linear(256,  128),
nn.ReLU(),
nn.Linear(128,  64),
nn.ReLU(),
nn.Linear(64,  32),
nn.ReLU(),
nn.Linear(32,  9),
```

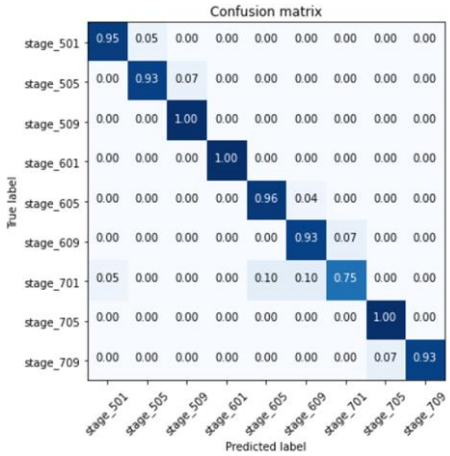
- 成果:

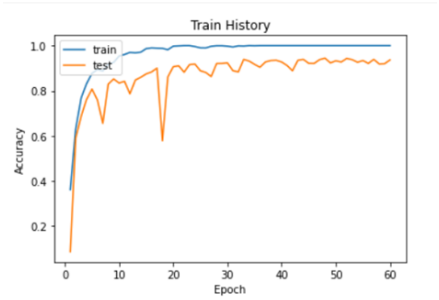






在複雜不及model 1 的情形型下，model 2 反而有比較好的準確率，從原來的80%左右提升至90%左右。然而從confusuion matrix 中可看出，該模型對天數為701天的圖像有很差的辨識能力，主要是會誤認成605以及609的圖像。所以接著我們把資料集的比例從8:2調成9:2:





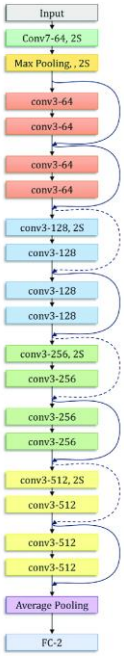
可以發現，整體的準確率又更高了，但唯獨701天的準確率還是慘不忍睹，仍然會誤認成605以及609。

stage 605　　stage 609　　stage 701

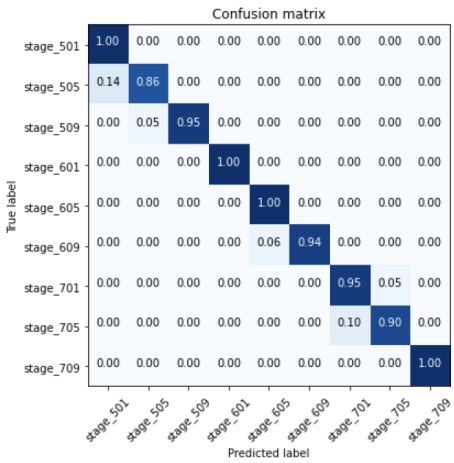
從這次期末專題也讓我體認到，一個模型越複雜，準確率卻不一定會更高的道理，可能算是一種overfitting

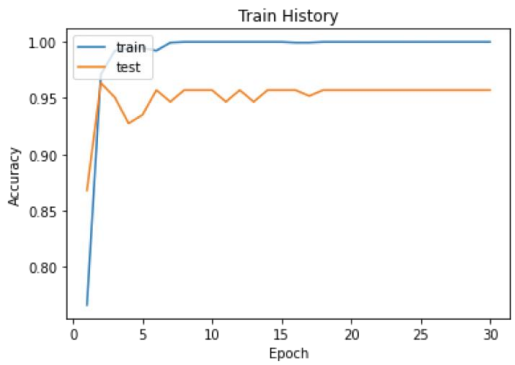
3.ResNet18(Pretrained)

```
#  model  =  Classifier().to(device)
model  =  models.resnet18(pretrained=True)
model  =  model.to(device)
#  For  Resnet  Finetunning
```



我們使用pytorch的ResNet model，挑選ResNet18是因為考量到訓練的時間，模型越大所需要訓練的時間也越長。我們有試過使用沒有Pretrained過的ResNet直接拿我們資料下去做訓練，發現在跑了60個epochs後，準確度並沒有高出我們自己設計的CNN model太多。因此我們使用了Pretrained過的ResNet18 model去做fine tuning，訓練了大概20多個epochs，準確度就可以收斂在97%左右。





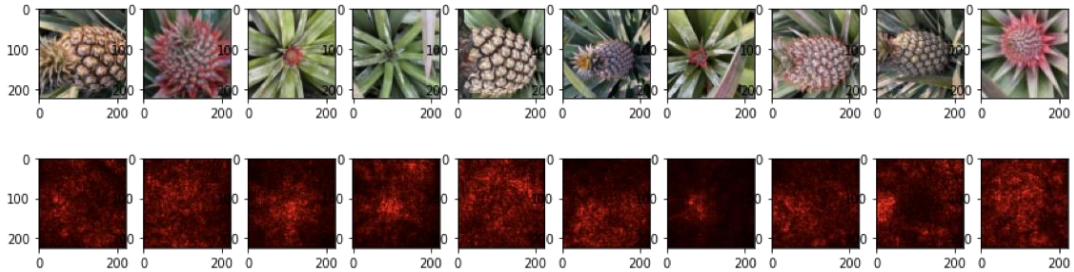
- ExplanableAI

我們想確認模型判斷圖片的位置是否跟我們想像中的一樣，因此使用三種explanable

AI的方式來分析我們的CNN model。分別為Integrated gradient、Saliency map及LIME。

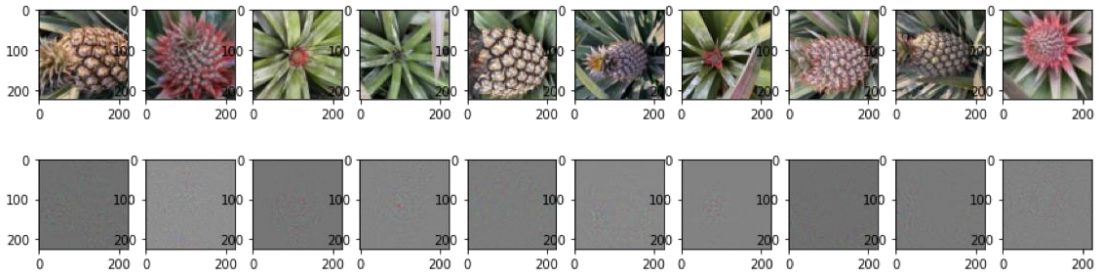
1.Saliency Maps:

透過模型中的Weight來推測圖片中哪些位置對於預測結果有顯著的影響，而Weight我們可以透過model對input的導數計算出來。



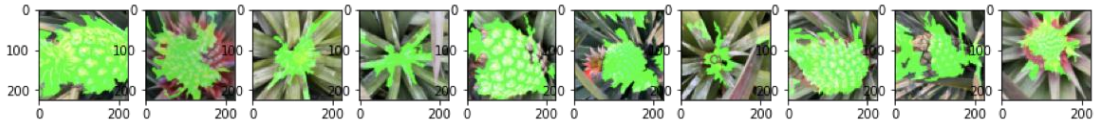
2.Integrated Gradients:

使用saliency maps會遇到一個問題，當一張圖片已經被預測將近100%是某個class時，會導致判斷該位置的weight達到飽和，使得saliency maps計算gradient會得到0。為了解決這個問題，可以將gradient積分出來，再比較圖片中每個位置的重要性。



3.LIME:

LIME的全名為Local Interpretable Model-agnostic Explanations。資料分割成小區塊，隨機擾動預解釋的個體的小區塊 (Perturbed Instance) 產生新樣本，丟進模型預測。再根據擾動樣本建立出一個簡單線性回歸，並且找出係數值最大的 M 個特徵。



- 討論

我們認為模型會有誤差主要有以下幾點:

1.資料集不足:

這次的資料及每個階段有200張照片，即使有先經過data augmentation 的步驟資料量仍略顯不足。

2.在一開始Label的時候可能就有誤:

這次的辨別目標是辨認鳳梨的生長天數，如果有看資料集的圖片就會知道，有些天數的照片根本是一模一樣，有可能其實在不同label裡會有相同生長天數的鳳梨，標記時並不像是課堂中練習的 CIFAR10 一樣可以武斷的判斷出該資料是什麼類別。

3.模型的複雜度不夠:

一般而言，一個模型的準確率總不可能100%，但像模型二那樣對某一類別的準確率只有60、70%左右就代表是模型複雜度的問題。但一個模型的結構該如何去架設本來就是一門學問，如果要去探討的話絕非易事。

4 .Data Augmentation的選擇:

在Data augmentation 中，有一個步驟是轉灰階圖像，但後來發現顏色也是判斷的其中一個依據，因此對模型訓練的準確率可能造成影響。