專案計劃書

1.題目: [Plant Pathology Challenge](#) [[kaggle]](#)

2. 資料描述:

傳統上使用人工的方式，消耗人力及費時。再者，對於疾病的錯誤判斷可能導致抗藥性的病原體產生，使得疾病難以被根治。此為CVPR2020的FGVC7研討會挑戰。
挑戰目標:

1. 準確分類test資料集中，健康圖片或不同患病的類別
2. 分類多種疾病類別
3. 可根據現實生活中的情況下量化疾病的研究程度
4. 處理少見的類別
5. 解決深度感知問題(葉子的角度、光線、陰影及年紀)
6. 將專家知識整合到識別中(量化及利用計算機視覺方法引導計算)

2-1 輸入與輸出

輸入

特徵(feature):

feature:RGB照片(1822張)

答案:

healthy、multiple_diseases、rust、scab

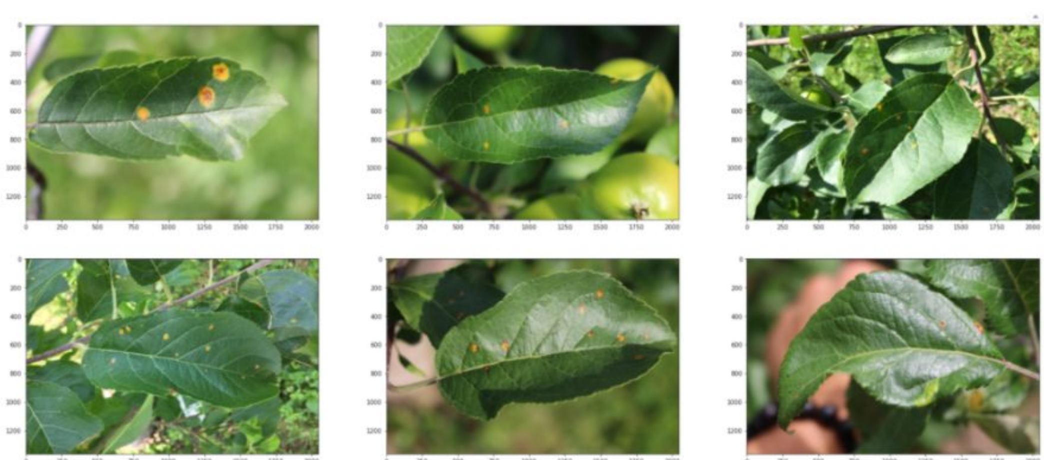
輸出

給定test資料集，所檢測的結果。

輸入答案 - scab

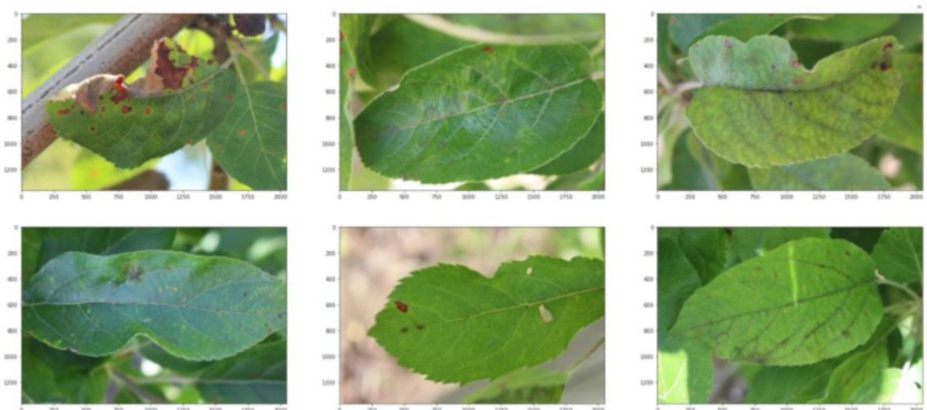
有明顯的棕色斑點、污漬，真菌或細菌引起的各種植物病害，
並導致在果實，葉或根上形成硬殼狀斑點。

輸入答案 - rust



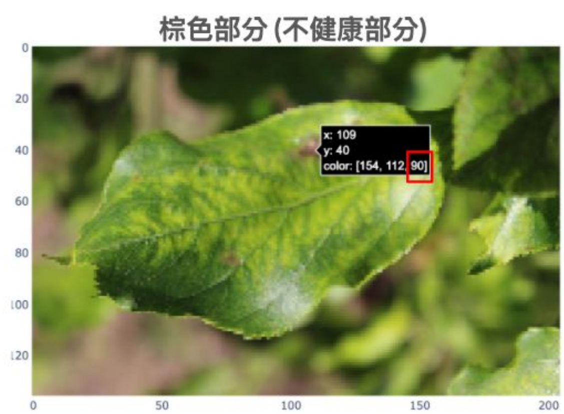
有明顯的棕黃色斑點，特殊真菌感染

輸入答案 - multiple diseases



有黃色、棕色斑點，每一種有兩種以上

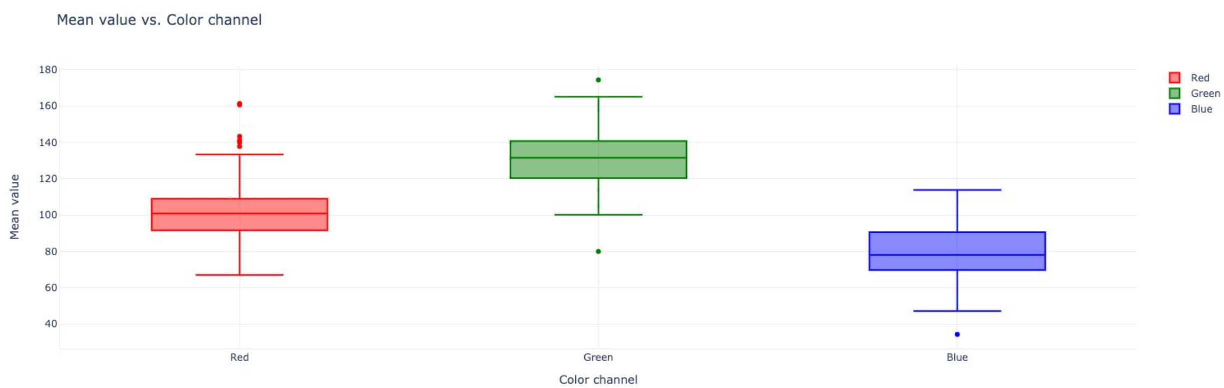
輸入資料視覺化



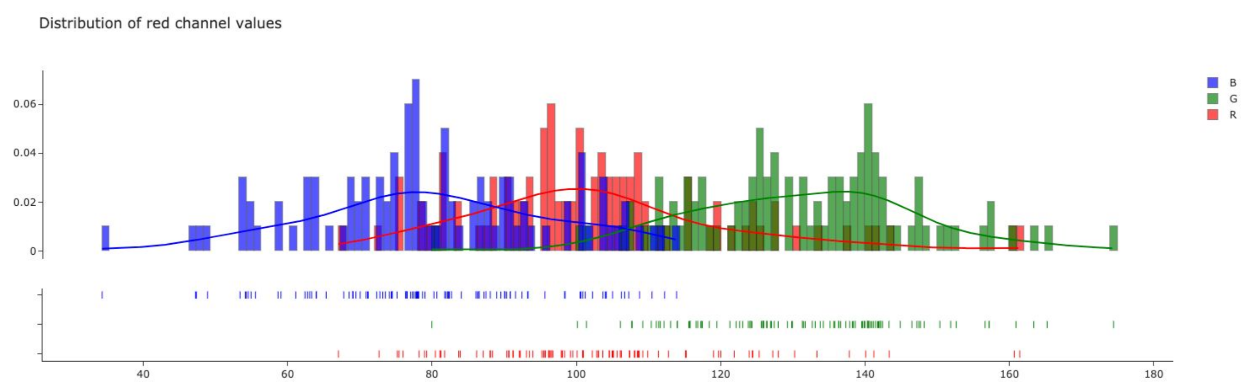
棕色斑點的B通道(藍色)值較高

2-2 資料分佈情形

2-2-1 盒鬚圖分析



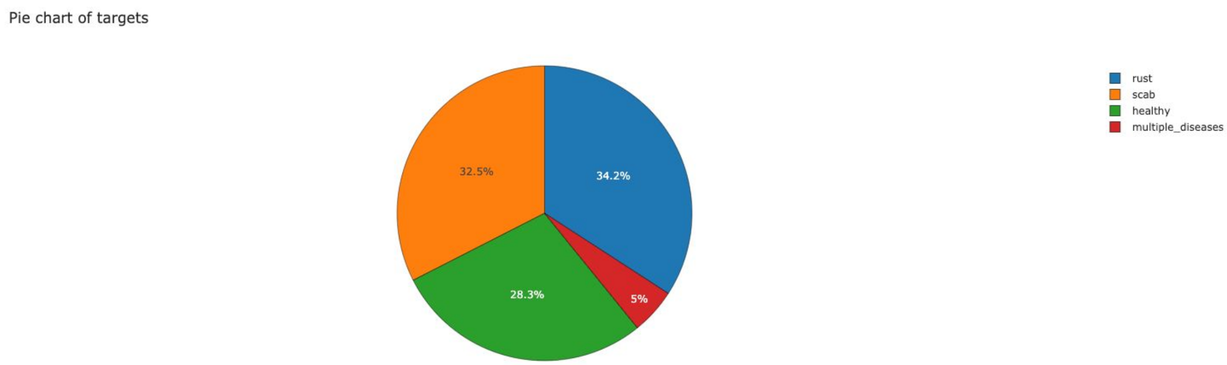
2-2-2 離散分佈



由上可知資料的分佈情形，綠色較多，藍色的較少，每個顏色都是呈現常態分佈

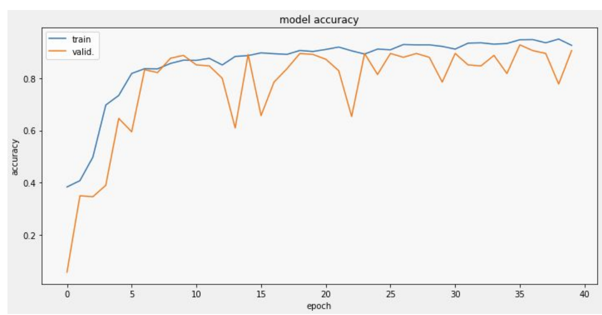
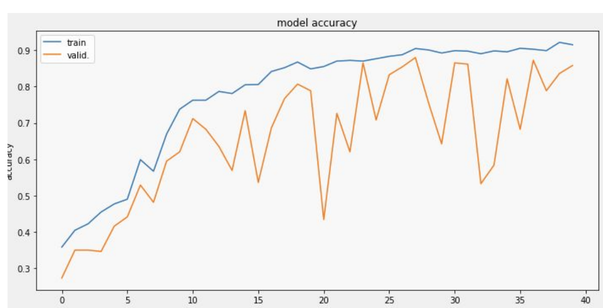
2-2-3 訓練集類別的分佈情形

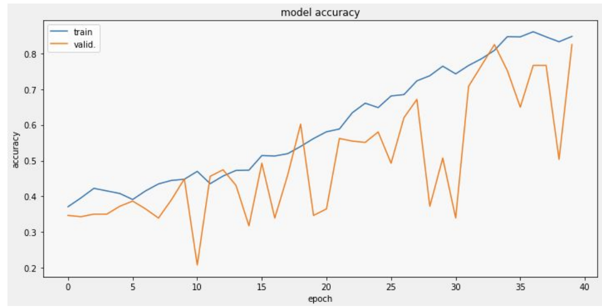
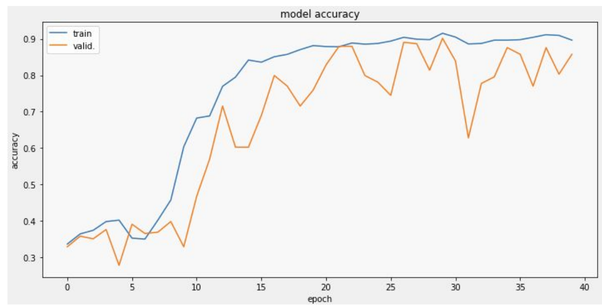
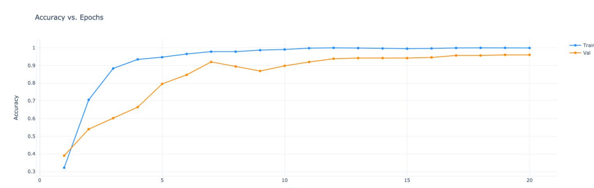
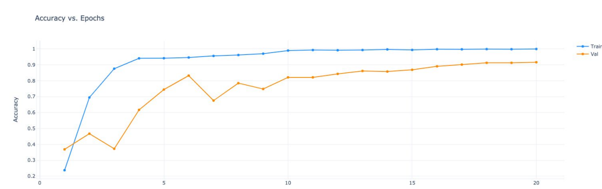
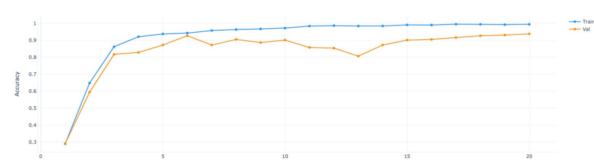
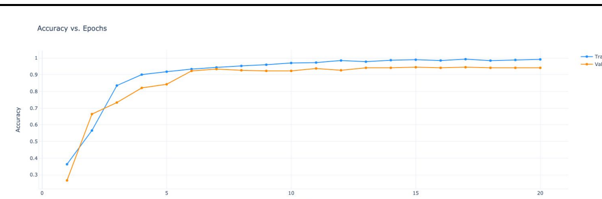
ㄋ

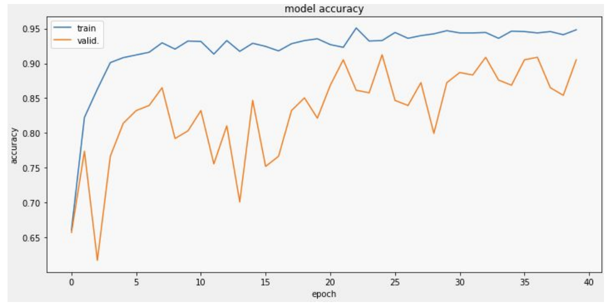


scab最多，多種疾病最少

3. 模型&訓練過程

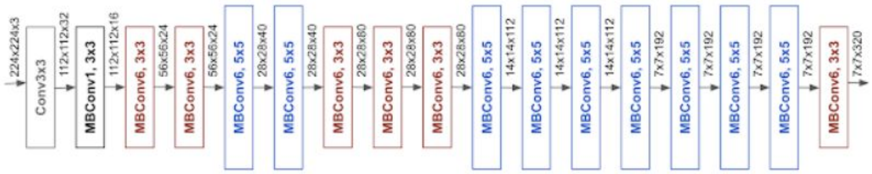
leafNet (out model) 👑 Conv,k(3,3) 64*2、128*2、256*2、512*3 Maxpool,(3,3) -- acc:0.95	
Conv,k(3,3) 64*2、128*2、256*2 Maxpool,(3,3) -- acc:0.9	

<table>
<tr>
<td>Conv,k(3,3)
64*2、128*2、256*2、512*3
Maxpool,(2,2)
----
acc:0.8</td>
<td>
</td>
</tr>
<tr>
<td>Conv,k(3,3)
64*2、128*2、256*2、512*3、512*3
Maxpool,(2,2)
----
acc:0.9</td>
<td>
</td>
</tr>
<tr>
<td>DenseNet121
with GlobalAveragePooling2D
----
acc:0.9941</td>
<td>
</td>
</tr>
<tr>
<td>DenseNet121-modified
with
GlobalAveragePooling2D
Dense(2048) *2
----
acc:0.9993</td>
<td>
</td>
</tr>
<tr>
<td>EfficientNetB7
----
acc:0.9889</td>
<td>
</td>
</tr>
<tr>
<td>EfficientNetB7 - modified
----
acc:0.9935</td>
<td>
</td>
</tr>
</table>

<table>
<tr>
<td>InceptionResNetV2
----
acc:0.969</td>
<td>
</td>
</tr>
<tr>
<td>Model
----
acc:</td>
<td></td>
</tr>
</table>

類神經模型結構 - EfficientNet-B0

[5]



The architecture for our baseline network EfficientNet-B0 is simple and clean, making it easier to scale and generalize.

類神經模型結構 - DenseNet

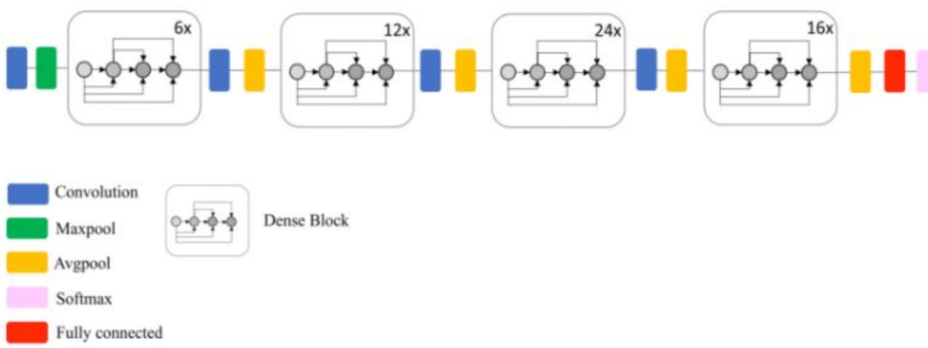
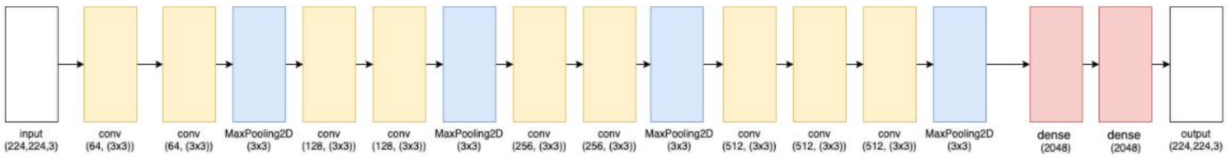


Figure 6. Schematic diagram of DenseNet model (compressed view).

[2]

類神經模型結構 - leafNet (our model)



類神經模型結構 - InceptionResNetV2

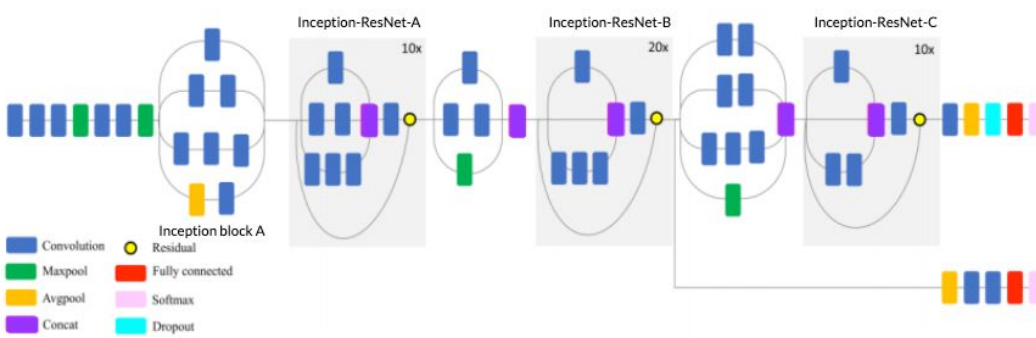


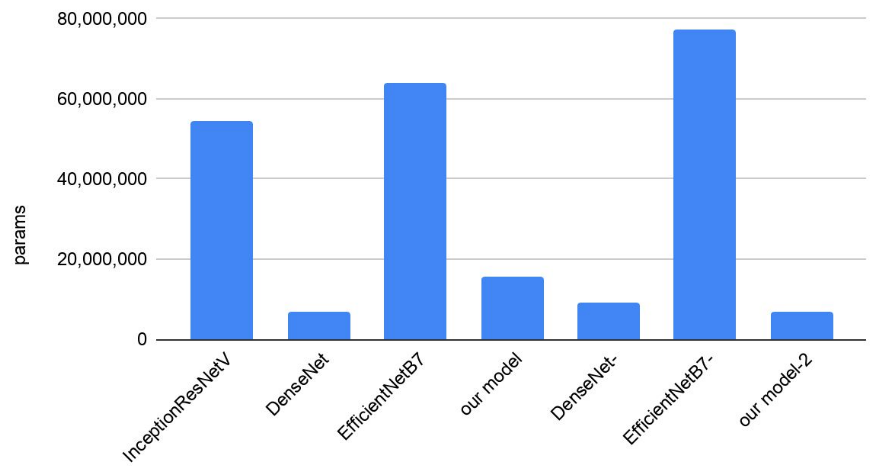
Figure 5. Schematic diagram of InceptionResNetV2 model (compressed view).

[2]

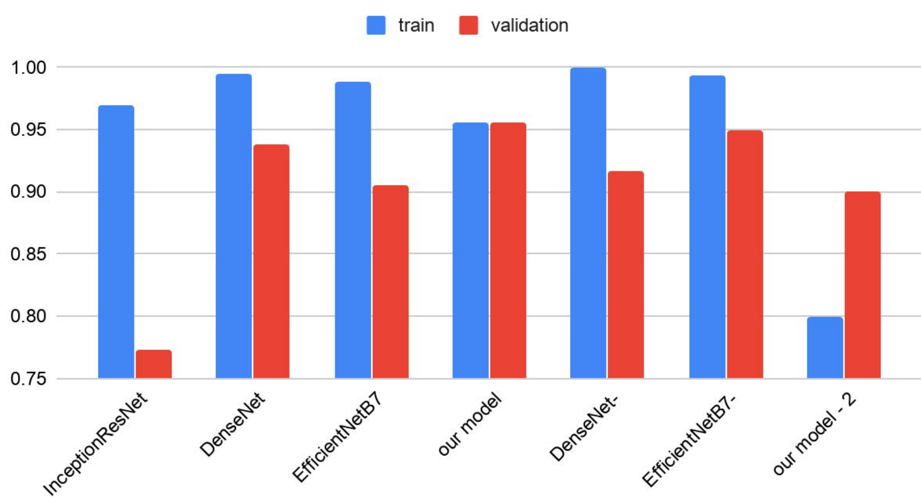
參數請參考附錄

4. 結果比較

縱軸：params，橫軸：

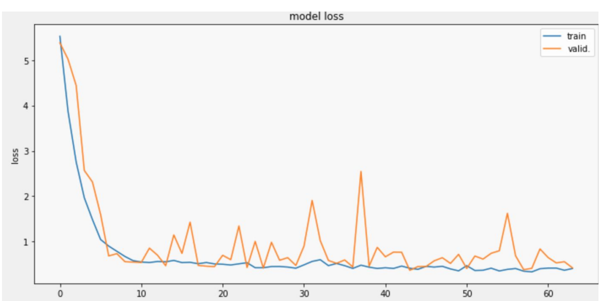
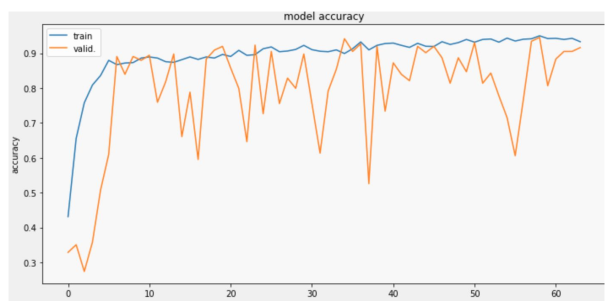


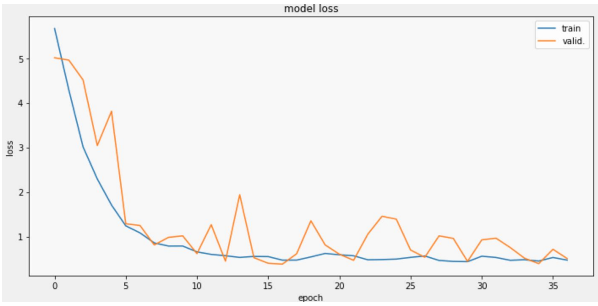
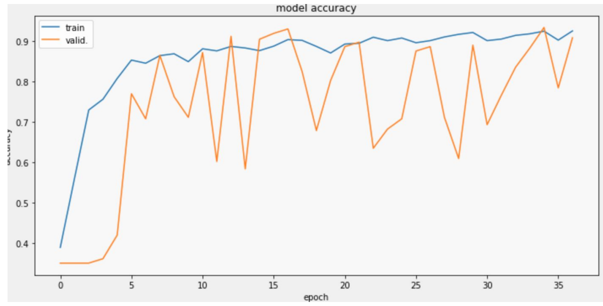
train和validation



5. 心得

上完一學期的類神經網路的課，也對深度學習有了更深的認識，也因為老師平常的作業練習及理論的推導、參數的計算等等，使這次的期末作業也能如期的順利完成，雖然剛開始做Fine-tuning的時候會很痛苦，看到好不容易計算好的model竟然準確度結果很差，後來我選擇站在巨人的肩膀上，參考了VGG19的網路架構再做一點改善(因為直接使用VGG19效果不好)，我加大Maxpooling的原因是從訓練資料看有瑕疵的地方都不會很大，而且前面也有提到在藍色通道可以看出瑕疵的地方，因此使用Maxpooling，而縮減VGG19的原因是，VGG19的參數比較多，有發生under-fitting的問題，我們簡化網路少掉最後的三層512，而達到0.95的準確率且over-fitting的問題也不會很嚴重，雖然從曲線上看訓練有點不穩，validation的曲線程度震盪很大，因此可以加上L1、L2制約、添加Dropout及增加epoch數可以使模型在更穩定。除此之外，而且stack overflow也有許多的資源，Kaggle本身也有很多善心的大大提供更有效率的kernel，也讓我認識了一個互動式的視覺化工具-plotly及Tensorflow分散化訓練(tpu及多GPU的training方式)及資料pipline的方法，改進了許多期中以前使用fit_generator所遇到的問題，有一些別人的方法也讓我認識到了一些在做模型辨識state-of-art的演算法，從閱讀論文開始，了解近期的研究方向及別人是怎麼去設計架構的，來增加往後設計模型的知識。我也觀察了別人重新訓練model的技巧，並改善了一點最後幾層來做transfer learning，我們是把他們的最後一層拿掉，加上GlobalAvgPooling及加上幾層的FCN層，因為這樣可以使得他們比較不會遇到單層線性不可分問題，改進了一點準確率，不過在DenseNet反而使over-fitting問題更加嚴重。
特別感謝王豐緒老師一學期的用心教導，帶我們不只了解了FCN網路，更了解了CNN、RNN、LSTM、及臉部辨識的理論基礎及實作，也感謝專題老師賈叢林老師提供的運算資源，讓我能順利完成此份期末作業。有機會也會希望能再繼續修老師的進階深度學習的課程。

leafNet 添加 Dropout(0.2) 及L2=0.01	
	

leafNet 添加 Dropout(0.4) 及L2=0.01	
	

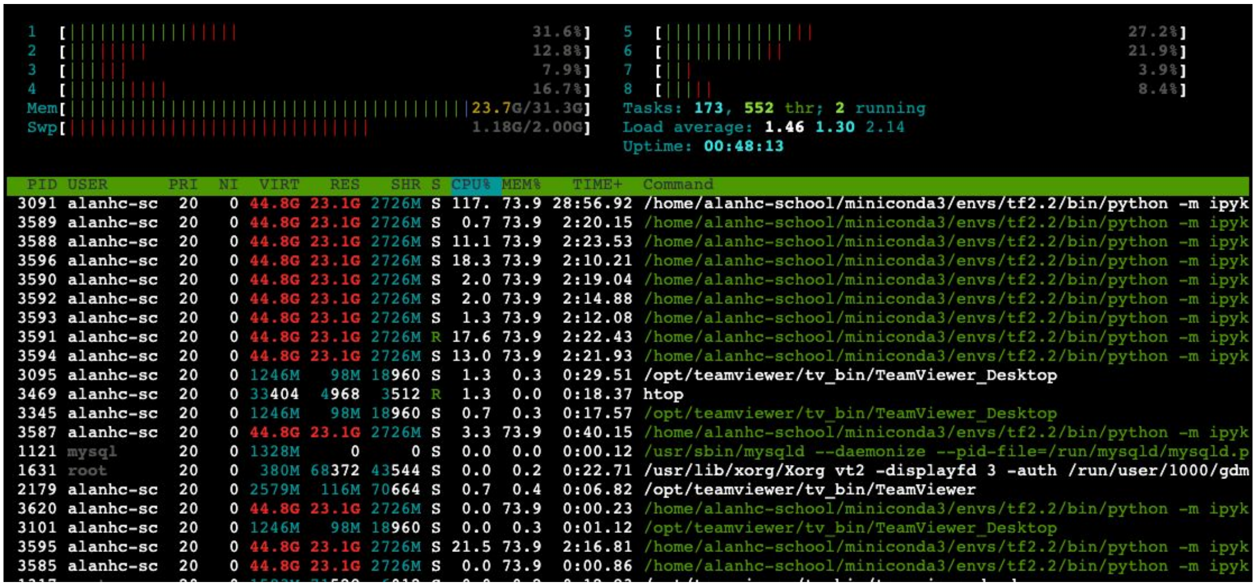
更詳細可以參考當天demo的投影片：

https://docs.google.com/presentation/d/1S1qe0ZPdjJQNO04-AOHunQ4qWvtONIWou6CSfR1ENGs/edit?usp=sharing

6. 採坑

- 圖片大小 image_size

原先使用800*800的resolution，因此在colab一直無法使用，因為讀圖片所需的RAM超過colab限制的12G，可以將image_size改小即可



- 訓練準確度很低

原先使用 800*800的圖片時，準確度會很低，大概在0.3-0.4，有試著將pool size放大去讓維度縮小，但效果不佳，可能需要更多時間去做調整，例如改Conv kernel與層數，目前已經試過加大kernel size及pool size。

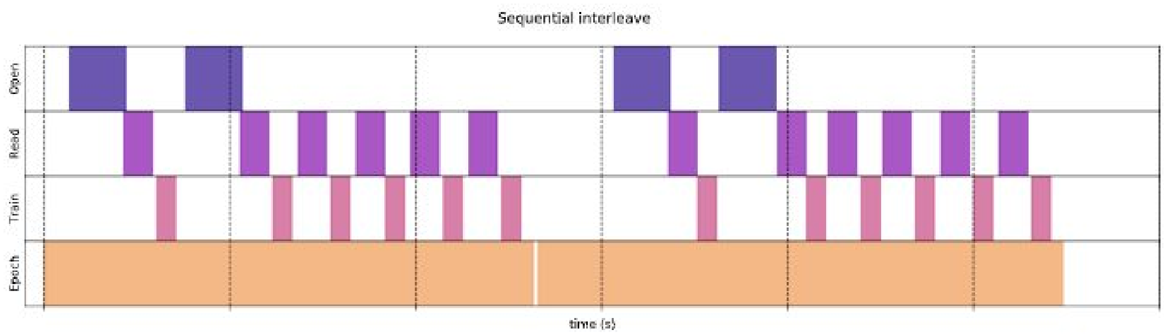
- OOM …Memory... error

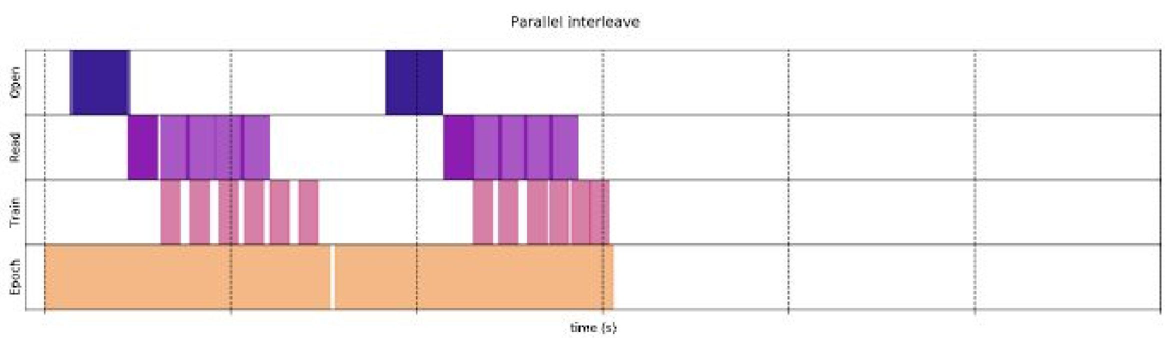
以前很常出現這錯誤，後來發現是flatten層與Dense差距過大就會產生此錯誤

7. 學到的小技巧

- Build TensorFlow input pipelines

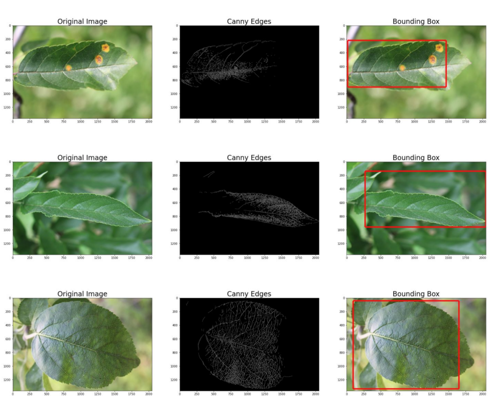
  在之前的作業裡面都是使用fit_generator的方式去做處理，發現處理資料佔大部分的時間這次期末作業就找加快讀取的技巧，就是使用 tf.data，使用pipline的方式去平行處理，節省等待時間。



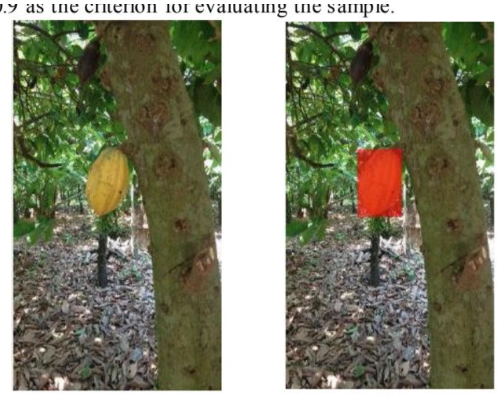


8. 可以改進的地方

根據[7-13]，我們知道了有一些論文著重在把背景去除，或者有部分是使用Object tracking的技術去擷取的水果的部位，雖然深度學習比起傳統特徵擷取方法對不同環境的適應性更高，但有時所需的計算資源可能也越多，以後可以試著去做擷取葉子的部分，來讓input的resolution較小，而不是等比例放大而已。



0.9 as the criterion for evaluating the sample.



(a) (b)

Fig. 12: (a) Image inputted, (b) Image result

- Distributed training with TensorFlow

  因為有部分模型我是使用Kaggle提供的GPU，也剛好學到了這個技巧，只要定義strategy就可以使用tpu或者多GPU的分散式訓練方式

  1. 定義

```
try:
    tpu = tf.distribute.cluster_resolver.TPUClusterResolver()
    print('Running on TPU ', tpu.master())
except ValueError:
    tpu = None

if tpu:
    tf.config.experimental_connect_to_cluster(tpu)
    tf.tpu.experimental.initialize_tpu_system(tpu)
    strategy = tf.distribute.experimental.TPUStrategy(tpu)
    print('use tpu')
else:
    if tf.config.list_physical_devices('GPU'):
      strategy = tf.distribute.MirroredStrategy()
      print('use gpu')
    else:  # use default strategy
      print('use cpu')
      strategy = tf.distribute.get_strategy()
```

使用多GPU：`tf.distribute.MirroredStrategy()`

使用TPU：`tf.distribute.MirroredStrategy()`

查看目前支援的加速裝置： `tf.config.list_physical_devices()`

  2. 使用定義好的strategy方式訓練

```
with strategy.scope():
    model3 = Sequential()

    model3.add(Conv2D(64, (3, 3),
input_shape=(image_size,image_size,3),padding='same'))
    model3.add(BatchNormalization(axis=-1))
    model3.add(Activation('relu'))
    model3.add(Conv2D(64, (3, 3),padding='same'))
 ...
```

參考資料

1. Thapa, R., Snavely, N., Belongie, S., & Khan, A. (2020). The Plant Pathology 2020 challenge dataset to classify foliar disease of apples. *arXiv preprint arXiv:2004.11958*.
2. Mahdianpari, M., Salehi, B., Rezaee, M., Mohammadimanesh, F., & Zhang, Y. (2018). Very deep convolutional neural networks for complex land cover mapping using multispectral remote sensing imagery. *Remote Sensing*, *10*(7), 1119.
3. Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017, February). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*.
4. Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4700-4708).
5. Tan, M., & Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*.
6. Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., & Le, Q. V. (2019). Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2820-2828).
7. Mazen, F. M., & Nashat, A. A. (2019). Ripeness classification of bananas using an artificial neural network. *Arabian Journal for Science and Engineering*, *44*(8), 6901-6910.
8. Hassan, N. M. H., & Nashat, A. A. (2019). New effective techniques for automatic detection and classification of external olive fruits defects based on image processing techniques. *Multidimensional Systems and Signal Processing*, *30*(2), 571-589.
9. Basri, H., Syarif, I., Sukaridhoto, S., & Falah, M. F. (2019). INTELLIGENT SYSTEM FOR AUTOMATIC CLASSIFICATION OF FRUIT DEFECT USING FASTER REGION-BASED CONVOLUTIONAL NEURAL NETWORK (FASTER R-CNN). *Jurnal Ilmiah Kursor*, *10*(1).
10. Sahu, D., & Dewangan, C. (2017). Identification and classification of mango fruits using image processing. *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol*, *2*(2), 203-210.
11. Sahu, D., & Potdar, R. M. (2017). Defect identification and maturity detection of mango fruits using image analysis. *American Journal of Artificial Intelligence*, *1*(1), 5-14.
12. Thendral, R., & Suhasini, A. (2017). Automated skin defect identification system for orange fruit grading based on genetic algorithm. *Current Sci*, *112*(8), 1704-1711.
13. Ireri, D., Belal, E., Okinda, C., Makange, N., & Ji, C. (2019). A computer vision system for defect discrimination and grading in tomatoes using machine learning and image processing. *Artificial Intelligence in Agriculture*, *2*, 28-37.
14. Veites-Campos, S. A., Ramírez-Betancour, R., & González-Pérez, M. (2018). Identification of cocoa pods with image processing and artificial neural networks. *International Journal of Advanced Engineering, Management and Science*, *4*(7).

網站資料

- https://www.tensorflow.org/guide/data
- https://www.tensorflow.org/guide/data_performance
- https://www.tensorflow.org/guide/distributed_training
- https://ai.googleblog.com/2016/08/improving-inception-and-image.html
- https://ai.googleblog.com/2016/08/improving-inception-and-image.html
- https://becominghuman.ai/updated-my-99-40-solution-to-udacity-nanodegree-project-p2-

traffic-sign-classification-5580ae5bd51f

- https://ai.googleblog.com/2018/08/mnasnet-towards-automating-design-of.html
- https://arxiv.org/abs/1807.11626
- https://arxiv.org/abs/1801.04381
- https://en.wikipedia.org/wiki/Neural_architecture_search

附錄

leafNet(our model)

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 224, 224, 64)      1792
_________________________________________________________________
batch_normalization_11 (Batc (None, 224, 224, 64)      256
_________________________________________________________________
activation_12 (Activation)   (None, 224, 224, 64)      0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 224, 224, 64)      36928
_________________________________________________________________
batch_normalization_12 (Batc (None, 224, 224, 64)      256
_________________________________________________________________
activation_13 (Activation)   (None, 224, 224, 64)      0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 74, 74, 64)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 74, 74, 128)       73856
_________________________________________________________________
batch_normalization_13 (Batc (None, 74, 74, 128)       512
_________________________________________________________________
activation_14 (Activation)   (None, 74, 74, 128)       0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 74, 74, 128)       147584
_________________________________________________________________
batch_normalization_14 (Batc (None, 74, 74, 128)       512
_________________________________________________________________
activation_15 (Activation)   (None, 74, 74, 128)       0
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 24, 24, 128)       0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 24, 24, 256)       295168
_________________________________________________________________
```

```
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 24, 24, 256)       295168
_________________________________________________________________
batch_normalization_15 (Batc (None, 24, 24, 256)       1024
_________________________________________________________________
activation_16 (Activation)   (None, 24, 24, 256)       0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 24, 24, 256)       590080
_________________________________________________________________
batch_normalization_16 (Batc (None, 24, 24, 256)       1024
_________________________________________________________________
activation_17 (Activation)   (None, 24, 24, 256)       0
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 8, 8, 256)         0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 8, 8, 512)         1180160
_________________________________________________________________
batch_normalization_17 (Batc (None, 8, 8, 512)         2048
_________________________________________________________________
activation_18 (Activation)   (None, 8, 8, 512)         0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 8, 8, 512)         2359808
_________________________________________________________________
batch_normalization_18 (Batc (None, 8, 8, 512)         2048
_________________________________________________________________
activation_19 (Activation)   (None, 8, 8, 512)         0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 8, 8, 512)         2359808
_________________________________________________________________
batch_normalization_19 (Batc (None, 8, 8, 512)         2048
_________________________________________________________________
activation_20 (Activation)   (None, 8, 8, 512)         0
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 2, 2, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_3 (Dense)              (None, 2048)              4196352
```

```
_________________________________________________________________
dense_3 (Dense)              (None, 2048)              4196352
_________________________________________________________________
batch_normalization_20 (Batc (None, 2048)              8192
_________________________________________________________________
activation_21 (Activation)   (None, 2048)              0
_________________________________________________________________
dense_4 (Dense)              (None, 2048)              4196352
_________________________________________________________________
batch_normalization_21 (Batc (None, 2048)              8192
_________________________________________________________________
activation_22 (Activation)   (None, 2048)              0
_________________________________________________________________
dense_5 (Dense)              (None, 4)                 8196
_________________________________________________________________
activation_23 (Activation)   (None, 4)                 0
=================================================================
Total params: 15,472,196
Trainable params: 15,459,140
Non-trainable params: 13,056
_________________________________________________________________
```

EfficientNetB7

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
efficientnet-b7 (Model)      (None, 16, 16, 2560)      64097680
_________________________________________________________________
global_average_pooling2d_1 ( (None, 2560)              0
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 10244
=================================================================
Total params: 64,107,924
Trainable params: 63,797,204
Non-trainable params: 310,720
_________________________________________________________________
```

EfficientNet-modified

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
efficientnet-b7 (Model)      (None, 16, 16, 2560)      64097680
_________________________________________________________________
global_average_pooling2d_1 ( (None, 2560)              0
_________________________________________________________________
dense_3 (Dense)              (None, 2560)              6556160
_________________________________________________________________
dense_4 (Dense)              (None, 2560)              6556160
_________________________________________________________________
dense_5 (Dense)              (None, 4)                 10244
=================================================================
Total params: 77,220,244
Trainable params: 76,909,524
Non-trainable params: 310,720
_________________________________________________________________
```

DenseNet

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
densenet121 (Model)          (None, 16, 16, 1024)      7037504
_________________________________________________________________
global_average_pooling2d (Gl (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 4)                 4100
=================================================================
Total params: 7,041,604
Trainable params: 6,957,956
Non-trainable params: 83,648
_________________________________________________________________
```

DenseNet - modified

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
densenet121 (Model)          (None, 16, 16, 1024)      7037504
_________________________________________________________________
global_average_pooling2d (Gl (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 1024)              1049600
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              1049600
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 4100
=================================================================
Total params: 9,140,804
Trainable params: 9,057,156
Non-trainable params: 83,648
_________________________________________________________________
```

Model

期末HW

06160485 曾宏鈞

05360365 徐友笙

05360153 詹瑞

06370136 蔡毓丞