【智慧企業整合】

Project 3

王老先生有塊地

糧食作物分類

110034560 王勇盛

指導教授：邱銘傳 教授

中華民國 111 年 1 月 7 日

目錄

圖目錄

表目錄

一、 背景說明

(一)、現況描述

全球 2050 年人口預估達 75-105 億人，糧食需求將面臨倍增的壓力。臺灣為糧食進口國，以熱量為基礎的糧食自給率相對偏低，同時在氣候變遷趨勢所致極端氣候日趨嚴重的困境下，糧食供應短缺及糧價上升無可避免，再加上近年來農村人口老化及少子化的影響，從事農業人力大幅短缺，農業生產力受到相當衝擊；且受限於我國自然環境限制，農業生產成本偏高，較難與國際競爭，若擬提升農業生產力，就必須強化產業結構調整及科技研發創新

(二)、問題描述 5W1H

我們的目標是建立一個深度學習模型，該模型可以無人飛機穿梭於農田上空，一邊分析目前種植產物種類，一邊將資料傳送雲端，透過雲端運算，進行符合成本與對環境傷害最少的農藥與化肥施用分析及對水資源最有效的管理，而農民只要透過一只手機或平板電腦連上雲端，了解目前農作物的生長狀況。

表(一)、 5W1H

What	大範圍農作物，不易巡田
When	農作物生長時期
Who	農民
Where	農田
Why	了解作物分布情況
How	深度學習、影像處理

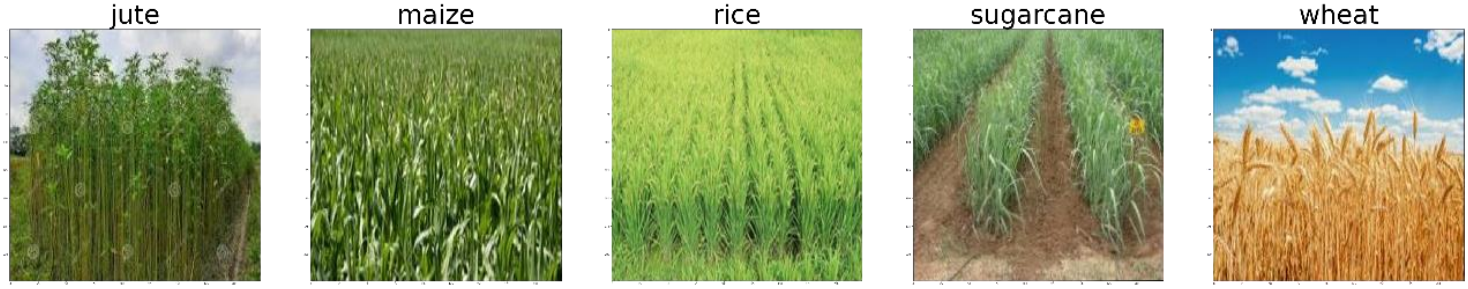
二、 資料介紹、處理

(一)、資料內容

由 Kaggle 公開數據集中取得 Agriculture crop images 的圖像資料集，資料集共分成 5 種類別：

- Jute(黃麻)
- Maize(玉米)
- Rice(稻米)
- Sugarcane(甘蔗)
- Wheat(小麥)

利用此資料集作為辨別農作物分類的的訓練資料。



圖(一)、農作物圖片

(二)、資料標籤

利用 OneHotEncoder 轉換其欲預測的 Label，效果如下：

圖(二)、OneHotEncoder 轉換

	path	labels
0	./input/kag2/jute\jute001a.jpeg	0
1	./input/kag2/jute\jute001ahf.jpeg	0
2	./input/kag2/jute\jute001ahs.jpeg	0
3	./input/kag2/jute\jute001arot.jpeg	0
4	./input/kag2/jute\jute002a.jpeg	0
...	...	...
799	./input/kag2/wheat\wheat039arot.jpeg	4
800	./input/kag2/wheat\wheat040a.jpeg	4
801	./input/kag2/wheat\wheat040ahf.jpeg	4
802	./input/kag2/wheat\wheat040ahs.jpeg	4
803	./input/kag2/wheat\wheat040arot.jpeg	4

	path	labels	label0	label1	label2	label3	label4
0	./input/kag2/jute\jute001a.jpeg	0	1.0	0.0	0.0	0.0	0.0
1	./input/kag2/jute\jute001ahf.jpeg	0	1.0	0.0	0.0	0.0	0.0
2	./input/kag2/jute\jute001ahs.jpeg	0	1.0	0.0	0.0	0.0	0.0
3	./input/kag2/jute\jute001arot.jpeg	0	1.0	0.0	0.0	0.0	0.0
4	./input/kag2/jute\jute002a.jpeg	0	1.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...
799	./input/kag2/wheat\wheat039arot.jpeg	4	0.0	0.0	0.0	0.0	1.0
800	./input/kag2/wheat\wheat040a.jpeg	4	0.0	0.0	0.0	0.0	1.0
801	./input/kag2/wheat\wheat040ahf.jpeg	4	0.0	0.0	0.0	0.0	1.0
802	./input/kag2/wheat\wheat040ahs.jpeg	4	0.0	0.0	0.0	0.0	1.0
803	./input/kag2/wheat\wheat040arot.jpeg	4	0.0	0.0	0.0	0.0	1.0

(三)、資料處理

為了避免 Overfitting 的情況將訓練集資料，透過 Data augmentation 的方式增加訓練及驗證張數。

Keras 透過 ImageDataGenerator class 提供 Data augmentation 相關的功能並輸出儲存，其功能如下如：

- 資料的正規化：我們的圖像在 RGB 通道都是 0~255 的整數，這樣的操作可能使圖像的值過高或過低，所以我們將這個值定為 0~1 之間的數。
- 進行隨機水平翻轉：隨機的對圖片進行水平翻轉

三、 模型介紹

(一)、VGG19

VGG 有兩種結構，分別是 VGG16 和 VGG19，兩者並沒有本質上的區別，只是網絡深度不一樣。

VGG 原理：

VGG16 是採用連續的幾個 3x3 的卷積核代替較大卷積核（11x11，7x7，5x5)。採用堆積的小卷積核是優於採用大的卷積核，因爲多層非線性層可以增加網絡深度來保證學習更複雜的模式，而且代價還比較小（參數更少)。

簡單來說，在 VGG 中，使用了 3 個 3x3 卷積核來代替 7x7 卷積核，使用了 2 個 3x3 卷積核來代替 5*5 卷積核，這樣做的主要目的是在保證具有相同的條件下，提升了網絡的深度，在一定程度上提升了神經網絡的效果。

比如，3 個步長爲 1 的 3x3 卷積核的一層層疊加作用可看成一個大小爲 7 的感受野（其實就表示 3 個 3x3 連續卷積相當於一個 7x7 卷積)，其參數總量爲 3x(9xC^2) ，如果直接使用 7x7 卷積核，其參數總量爲 49xC^2 ，這裏 C 指的是輸入和輸出的通道數。很明顯，27xC2 小於 49xC2，即減少了參數；而且 3x3 卷積核有利於更好地保持圖像性質。

這裏解釋一下爲什麼使用 2 個 3x3 卷積核可以來代替 5x5 卷積核：5x5 卷積看做一個小的全連接網絡在 5x5 區域滑動，我們可以先用一個 3x3 的卷積濾波器卷積，然後再用一個全連接層連接這個 3x3 卷積輸出，這個全連接層我們也可以看做一個 3x3 卷積層。這樣我們就可以用兩個 3x3 卷積級聯（疊加）起來代替一個 5x5 卷積。

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

圖(三)、VGG 網路架構

VGG 優缺點：

優點：

- VGGNet 的結構非常簡潔，整個網絡都使用了同樣大小的卷積核尺寸（3x3）和最大池化尺寸（2x2）。
- 幾個小濾波器（3x3）卷積層的組合比一個大濾波器（5x5 或 7x7）卷積層好：驗證了通過不斷加深網絡結構可以提升性能。

缺點：

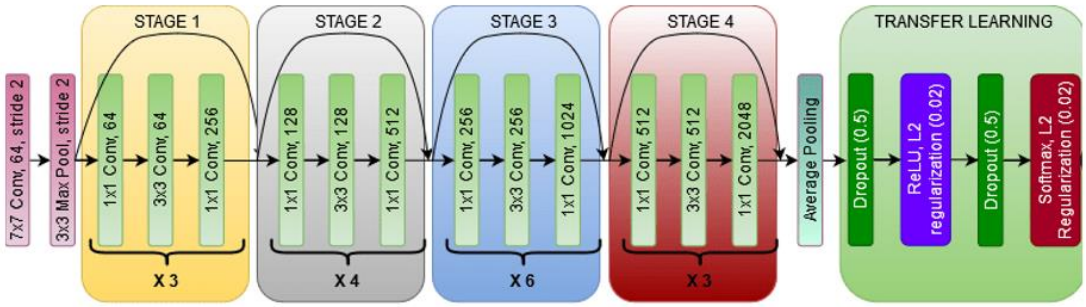
- VGG 耗費更多計算資源，並且使用了更多的參數，導致更多的內存佔用。

(二)、Resnet50

ResNet 全名為 Deep Residual Neural Network，中文名為「殘差網路」，是 2015 年的 ILSVRC 比賽冠軍。ResNet 的優點是它在 VGG 的基礎上，新增直連通路(skip connection)，不會新增任何參數，但是可以解決梯度消失的問題，但

其缺點需要較久的訓練時間。Resnet50 架構如下:

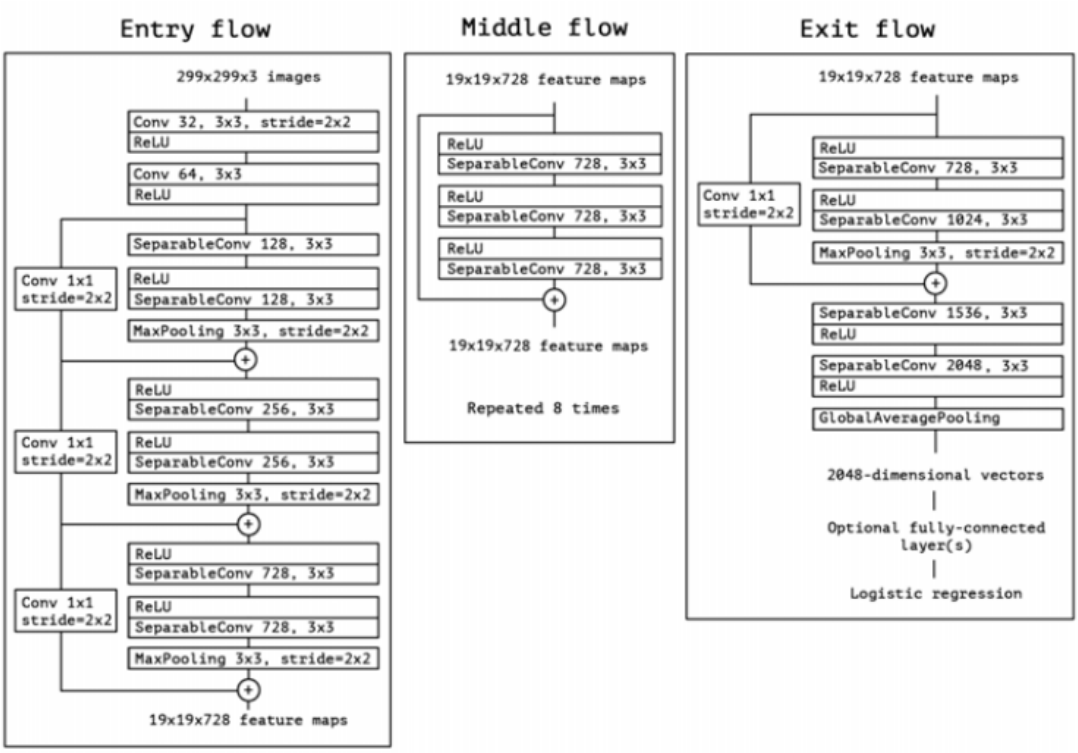
- 1.conv1：7x7 卷積核(共 1 層)
- 2.conv2_x：由三種卷積核(1x1,3x3,1x1)組成一個 block，疊 3 次。(共 9 層)
- 3.conv3_x：由三種卷積核(1x1,3x3,1x1)組成一個 block，疊 4 次。(共 12 層)
- 4.conv4_x：由三種卷積核(1x1,3x3,1x1)組成一個 block，疊 6 次。(共 18 層)
- 5.conv5_x：由三種卷積核(1x1,3x3,1x1)組成一個 block，疊 3 次。(共 9 層)
- 6.output：由 pooling+全連接層+softmax 組成。(共 1 層)
- 上方這六個部分加起來共 50 層。
- 模型加深可以提高準確率，但是加深會有許多困難要克服，例如：1.參數量變大、2.推論時間變久、3.梯度消失嚴重、4.硬體設備不足。這些困難導致模型無法訓練成功。而殘差網路解決了 1.和 3.，但它也只是模型加深的帶來的困難，而非完全解決。



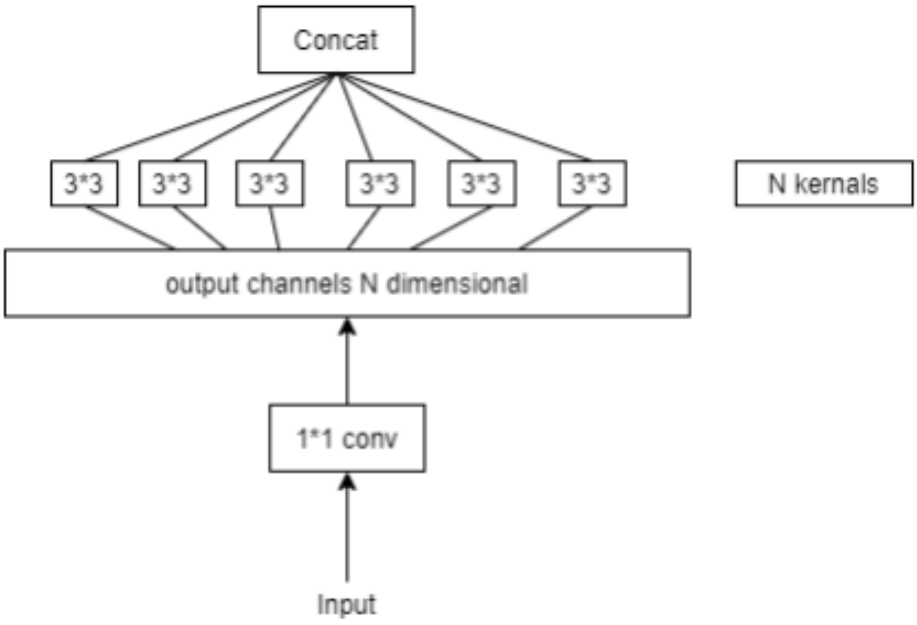
圖(四)、Resnet50 網路架構

(三)、Xception

以 InceptionV3 為基礎去做改進，採用 depthwise separable convolution(Extreme Inception) 來替代原本的 Inception module，並且加入了 ResNet 的 Residual Learning 方法。下圖就是 Xception 的整體架構，跟 Inception 一樣由多個不同的 module 組成，總共分為三個 flow — Entry/Middle/Exit，其中 Middle flow 會重複使用八次。



圖(五)、Xception 架構

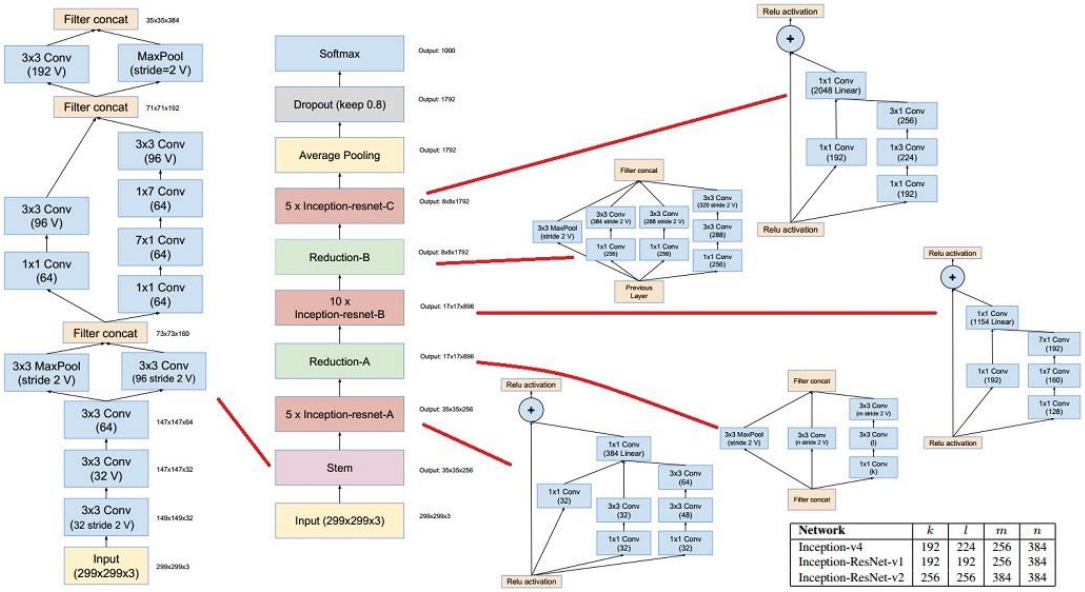


圖(六)、depthwise separable convolution

上圖為 Extreme Inception 的示意圖，其核心概念為個別考慮 channels 的資訊對其進行卷積之後再將其融合，將 channels 與空間分開是 ExtremeInception 的核心概念。

(四)、InceptionResNetV2

- ResNet 提出的殘差直連通路，可透過堆疊的方式加深網路，提升模型表現。
- 故 InceptionResNetV2 結合了 Inception Module 與殘差直連通路，設計新的 Inception-ResNet-A、B、C，期望透過加深網路，獲得更好的模型準確度。



圖(七)、InceptionResNetV2 架構圖

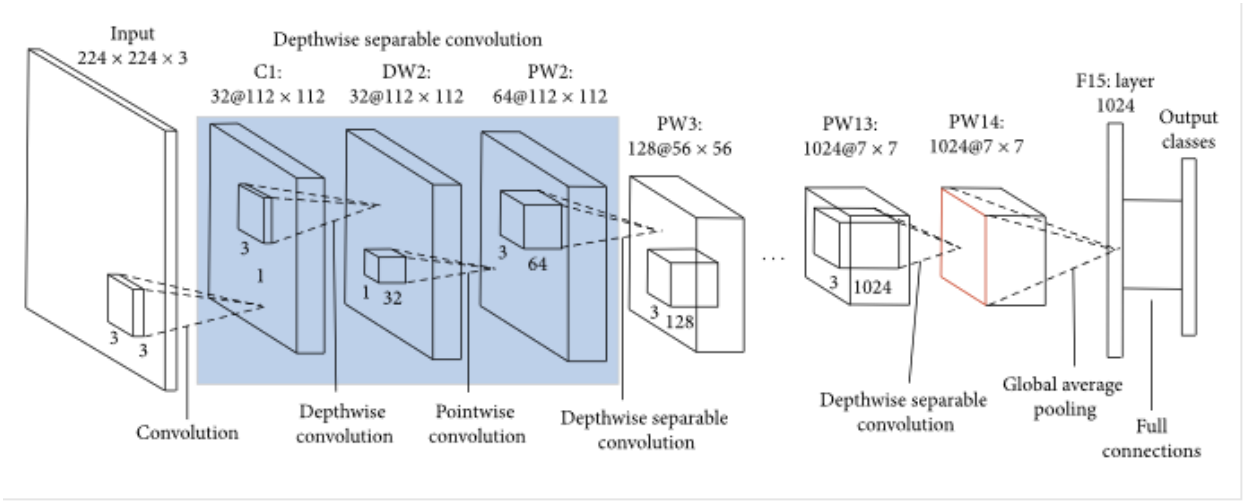
(五)、Mobilenet

MobileNetV1 的核心方法是透過兩個 Hyper parameters：Width multiplier 和 Resolution multiplier 控制模型的整體架構和參數量，並採用 Depth-wise Separable Convolution 做 Feature extraction。

假設 Convolution 的 Kernel size 為 5 × 5 ， 總共給 10 個 Kernel (filter)，輸入的圖片尺寸為 28 × 28 × 3，一般的 Convolutional layer 總共會有 750 (5 × 5 × 10 × 3)個參數，而 Mobilenet 則可以透過 Depth-wise Separable Convolution 及 Point-wise convolution 降低參數，在相同假設下，Mobilenet 的 Depth-wise convolution 參數為 75(5 × 5 × 3 = 75)個，加上 Point-wise convolution 參數為 30(1 × 1 × 10 × 3 = 30)個，總共 105 個參數，在維持相同的表現下，大幅減少運算資源。

$$\frac{D_K{}^2 \times M \times D_F{}^2 + M \times N \times D_F{}^2}{D_K{}^2 \times M \times N \times D_F{}^2} = \frac{1}{N} + \frac{1}{D_K{}^2}$$

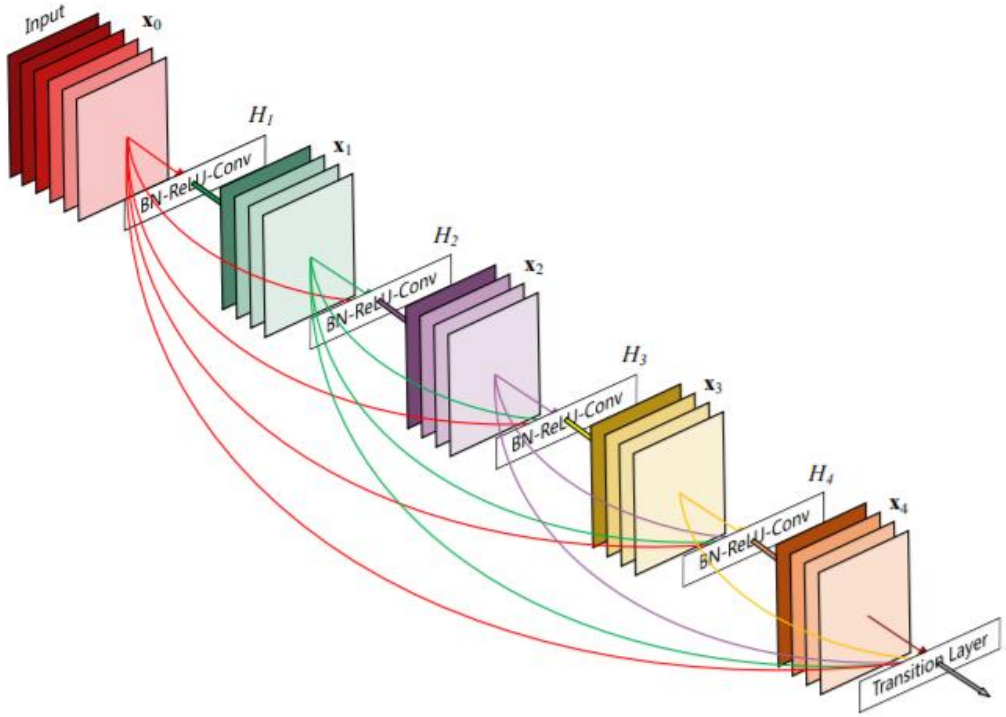
圖(八)、參數計算公式



圖(九)、Mobilenet 架構圖

(六)、Densenet

DenseNet 模型，它的基本思路與 ResNet 一致，但是它建立的是前面所有層與後面層的密集連接（dense connection），它的名稱也是由此而來。 DenseNet 的另一大特色是通過特徵在 channel 上的連接來實現特徵重用（feature reuse）。這些特點讓 DenseNet 在參數和計算成本更少的情形下實現比 ResNet 更優的性能，DenseNet 也因此斬獲 CVPR 2017 的最佳論文獎



圖(十)、Densenet

四、 訓練過程

(一)、VGG19

這裡使用官方已經打包好的 VGG19 的模型來檢視是否能有所改善。

模型層級與層級說明：

- 模型初始化
- 層級說明 Keras model 使用 Keras 提供之 CNN 模型(VGG19)
- 輸出層 Dense(1000， activation=tanh)
- 輸出層 Dense(1000， activation=tanh)
- 輸出層 Dense(1000， activation=tanh)
- 輸出層 Dense(5， activation=softmax)
- model.summary()如下圖：

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg19 (Functional)           (None, 512)               20024384
_________________________________________________________________
dense (Dense)                (None, 1000)              513000
_________________________________________________________________
dense_1 (Dense)              (None, 1000)              1001000
_________________________________________________________________
dense_2 (Dense)              (None, 1000)              1001000
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 5005
=================================================================
Total params: 22,544,389
Trainable params: 22,544,389
Non-trainable params: 0
_________________________________________________________________
```

圖(十一)、VGG19 架構

(二)、Resnet50

這裡使用官方已經打包好的 Resnet50 的模型來檢視是否能有所改善。

模型層級與層級說明：

- 模型初始化
- 層級說明 Keras model 使用 Keras 提供之 CNN 模型(Resnet50)
- 輸出層 Dense(1000， activation=tanh)
- 輸出層 Dense(1000， activation=tanh)

- 輸出層 Dense(1000， activation=tanh)
- 輸出層 Dense(5， activation=softmax)
- model.summary()如下圖：

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Functional)        (None, 2048)              23587712
_________________________________________________________________
dense (Dense)                (None, 1000)              2049000
_________________________________________________________________
dense_1 (Dense)              (None, 1000)              1001000
_________________________________________________________________
dense_2 (Dense)              (None, 1000)              1001000
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 5005
=================================================================
Total params: 27,643,717
Trainable params: 27,590,597
Non-trainable params: 53,120
_________________________________________________________________
```

圖(十二)、Resnet50 架構

(三)、Xception

這裡使用官方已經打包好的 Xception 的模型來檢視是否能有所改善。

模型層級與層級說明：

- 模型初始化
- 層級說明 Keras model 使用 Keras 提供之 CNN 模型(Xception)
- 輸出層 Dense(1000， activation=tanh)
- 輸出層 Dense(1000， activation=tanh)
- 輸出層 Dense(1000， activation=tanh)
- 輸出層 Dense(5， activation=softmax)
- model.summary()如下圖：

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
xception (Functional)        (None, 2048)              20861480
_________________________________________________________________
dense (Dense)                (None, 1000)              2049000
_________________________________________________________________
dense_1 (Dense)              (None, 1000)              1001000
_________________________________________________________________
dense_2 (Dense)              (None, 1000)              1001000
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 5005
=================================================================
Total params: 24,917,485
Trainable params: 24,862,957
Non-trainable params: 54,528
_________________________________________________________________
```

圖(十三)、Xception 架構

(四)、InceptionResNetV2

這裡使用官方已經打包好的 InceptionResNetV2 的模型來檢視是否能有所改善。

模型層級與層級說明：

- 模型初始化
- 層級說明 Keras model 使用 Keras 提供之 CNN 模型 (InceptionResNetV2)
- 輸出層 Dense(1000， activation=tanh)
- 輸出層 Dense(1000， activation=tanh)
- 輸出層 Dense(1000， activation=tanh)
- 輸出層 Dense(5， activation=softmax)
- model.summary()如下圖：

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_resnet_v2 (Functio (None, 1536)              54336736
_________________________________________________________________
dense (Dense)                (None, 1000)              1537000
_________________________________________________________________
dense_1 (Dense)              (None, 1000)              1001000
_________________________________________________________________
dense_2 (Dense)              (None, 1000)              1001000
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 5005
=================================================================
Total params: 57,880,741
Trainable params: 57,820,197
Non-trainable params: 60,544
_________________________________________________________________
```

圖(十四)、InceptionResNetV2 架構

(五)、MobileNet

這裡使用官方已經打包好的 Mobilenet 的模型來檢視是否能有所改善。

模型層級與層級說明：

- 模型初始化
- 層級說明 Keras model 使用 Keras 提供之 CNN 模型(MobileNet)
- 輸出層 Dense(1000， activation=tanh)
- 輸出層 Dense(1000， activation=tanh)
- 輸出層 Dense(1000， activation=tanh)
- 輸出層 Dense(5， activation=softmax)
- model.summary()如下圖：

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenet_1.00_224 (Function (None, 1024)              3228864
_________________________________________________________________
dense (Dense)                (None, 1000)              1025000
_________________________________________________________________
dense_1 (Dense)              (None, 1000)              1001000
_________________________________________________________________
dense_2 (Dense)              (None, 1000)              1001000
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 5005
=================================================================
Total params: 6,260,869
Trainable params: 6,238,981
Non-trainable params: 21,888
_________________________________________________________________
```

圖(十五)、MobileNet 架構

(六)、Densenet

這裡使用官方已經打包好的 Densenet 的模型來檢視是否能有所改善。

模型層級與層級說明：

- 模型初始化
- 層級說明 Keras model 使用 Keras 提供之 CNN 模型(Densenet)
- 輸出層 Dense(1000， activation=tanh)
- 輸出層 Dense(1000， activation=tanh)
- 輸出層 Dense(1000， activation=tanh)
- 輸出層 Dense(5， activation=softmax)
- model.summary()如下圖：

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
densenet169 (Functional)     (None, 1664)              12642880
_________________________________________________________________
dense (Dense)                (None, 1000)              1665000
_________________________________________________________________
dense_1 (Dense)              (None, 1000)              1001000
_________________________________________________________________
dense_2 (Dense)              (None, 1000)              1001000
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 5005
=================================================================
Total params: 16,314,885
Trainable params: 16,156,485
Non-trainable params: 158,400
_________________________________________________________________
```

圖(十六)、Densenet 架構

(七)、訓練結果分析

透過調整 optimizer、learning rate 和 loss function，後所得出的結果如下：

	Optimizer	Learning rate	Loss function	Accuracy
1.	Adam	Lr=0.001	cross entropy	91.93%
2.	Adam	Lr=0.001	MSE	17.39%
3.	Adam	Lr=0.05	cross entropy	24.22%
4.	Adam	Lr=0.05	MSE	16.77%
5.	Adam	Lr=0.01	cross entropy	24.22%
6.	Adam	Lr=0.01	MSE	16.77%
7.	SGD	Lr=0.001	cross entropy	34.16%
8.	SGD	Lr=0.001	MSE	34.16%
9.	SGD	Lr=0.05	cross entropy	24.22%
10.	SGD	Lr=0.05	MSE	45.96%
11.	SGD	Lr=0.01	cross entropy	49.07%
12.	SGD	Lr=0.01	MSE	26.09%

13.	Adagrad	Lr=0.001	cross entropy	27.33%
14.	Adagrad	Lr=0.001	MSE	39.13%
15.	Adagrad	Lr=0.05	cross entropy	32.92%
16.	Adagrad	Lr=0.05	MSE	40.37%
17.	Adagrad	Lr=0.01	cross entropy	39.75%
18.	Adagrad	Lr=0.01	MSE	21.12%

表(二)、VGG 訓練結果分析

	Optimizer	Learning rate	Loss function	Accuracy
1.	Adam	Lr=0.001	cross entropy	21.74%
2.	Adam	Lr=0.001	MSE	24.22%
3.	Adam	Lr=0.05	cross entropy	21.74%
4.	Adam	Lr=0.05	MSE	24.22%
5.	Adam	Lr=0.01	cross entropy	24.22%
6.	Adam	Lr=0.01	MSE	24.22%
7.	SGD	Lr=0.001	cross entropy	19.88%
8.	SGD	Lr=0.001	MSE	26.09%
9.	SGD	Lr=0.05	cross entropy	24.22%
10.	SGD	Lr=0.05	MSE	19.88%
11.	SGD	Lr=0.01	cross entropy	24.84%
12.	SGD	Lr=0.01	MSE	18.63%
13.	Adagrad	Lr=0.001	cross entropy	26.71%
14.	Adagrad	Lr=0.001	MSE	23.60%
15.	Adagrad	Lr=0.05	cross entropy	16.77%
16.	Adagrad	Lr=0.05	MSE	17.39%
17.	Adagrad	Lr=0.01	cross entropy	25.47%
18.	Adagrad	Lr=0.01	MSE	24.22%

表(三)、Resnet50 訓練結果分析

	Optimizer	Learning rate	Loss function	Accuracy
1.	Adam	Lr=0.001	cross entropy	96.89%
2.	Adam	Lr=0.001	MSE	24.22%
3.	Adam	Lr=0.05	cross entropy	24.22%
4.	Adam	Lr=0.05	MSE	24.22%
5.	Adam	Lr=0.01	cross entropy	21.74%
6.	Adam	Lr=0.01	MSE	24.22%
7.	SGD	Lr=0.001	cross entropy	81.99%
8.	SGD	Lr=0.001	MSE	46.58%
9.	SGD	Lr=0.05	cross entropy	99.38%
10.	SGD	Lr=0.05	MSE	96.89%

11.	SGD	Lr=0.01	cross entropy	98.14%
12.	SGD	Lr=0.01	MSE	85.09%
13.	Adagrad	Lr=0.001	cross entropy	91.93%
14.	Adagrad	Lr=0.001	MSE	67.70%
15.	Adagrad	Lr=0.05	cross entropy	98.14%
16.	Adagrad	Lr=0.05	MSE	98.76%
17.	Adagrad	Lr=0.01	cross entropy	98.76%
18.	Adagrad	Lr=0.01	MSE	93.17%

表(四)、Xception 訓練結果分析

	Optimizer	Learning rate	Loss function	Accuracy
1.	Adam	Lr=0.001	cross entropy	95.65%
2.	Adam	Lr=0.001	MSE	19.88%
3.	Adam	Lr=0.05	cross entropy	16.77%
4.	Adam	Lr=0.05	MSE	17.39%
5.	Adam	Lr=0.01	cross entropy	24.22%
6.	Adam	Lr=0.01	MSE	24.22%
7.	SGD	Lr=0.001	cross entropy	80.75%
8.	SGD	Lr=0.001	MSE	52.80%
9.	SGD	Lr=0.05	cross entropy	98.14%
10.	SGD	Lr=0.05	MSE	93.17%
11.	SGD	Lr=0.01	cross entropy	97.52%
12.	SGD	Lr=0.01	MSE	84.47%
13.	Adagrad	Lr=0.001	cross entropy	99.38%
14.	Adagrad	Lr=0.001	MSE	96.89%
15.	Adagrad	Lr=0.05	cross entropy	99.36%
16.	Adagrad	Lr=0.05	MSE	95.65%
17.	Adagrad	Lr=0.01	cross entropy	99.37%
18.	Adagrad	Lr=0.01	MSE	97.52%

表(五)、InceptionResNetV2 訓練結果分析

	Optimizer	Learning rate	Loss function	Accuracy
1.	Adam	Lr=0.001	cross entropy	97.52%
2.	Adam	Lr=0.001	MSE	24.22%
3.	Adam	Lr=0.05	cross entropy	21.74%
4.	Adam	Lr=0.05	MSE	16.77%
5.	Adam	Lr=0.01	cross entropy	19.88%
6.	Adam	Lr=0.01	MSE	16.77%
7.	SGD	Lr=0.001	cross entropy	91.30%
8.	SGD	Lr=0.001	MSE	44.72%

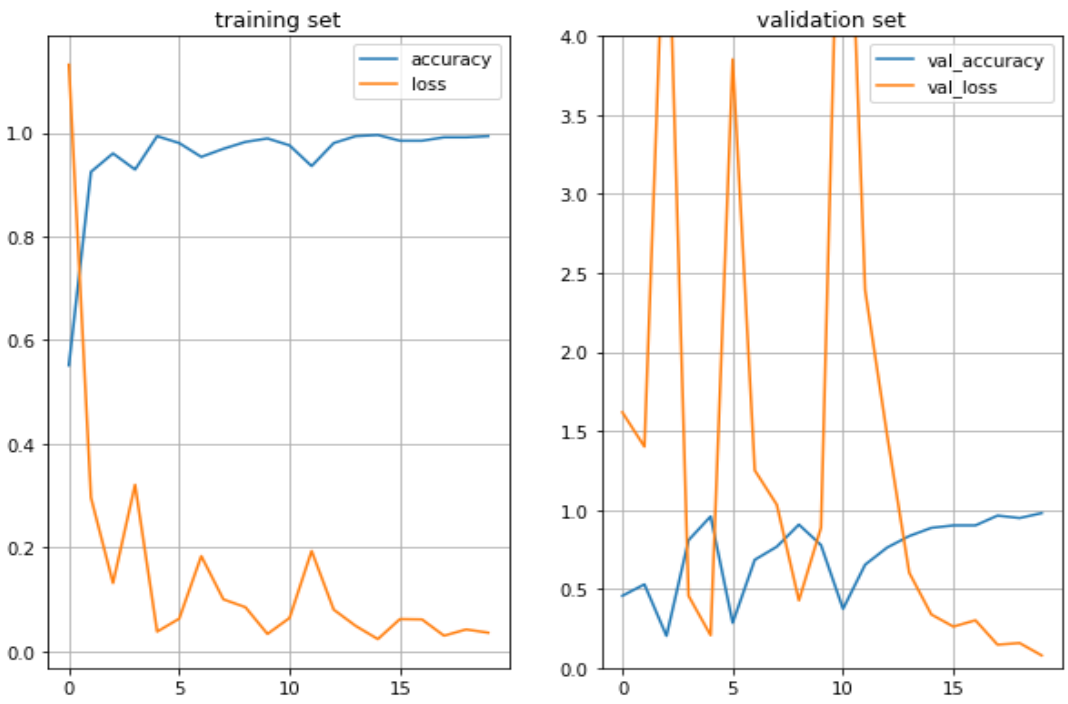
9.	SGD	Lr=0.05	cross entropy	98.14%
10.	SGD	Lr=0.05	MSE	98.14%
11.	SGD	Lr=0.01	cross entropy	98.14%
12.	SGD	Lr=0.01	MSE	90.68%
13.	Adagrad	Lr=0.001	cross entropy	98.74%
14.	Adagrad	Lr=0.001	MSE	98.14%
15.	Adagrad	Lr=0.05	cross entropy	98.76%
16.	Adagrad	Lr=0.05	MSE	96.27%
17.	Adagrad	Lr=0.01	cross entropy	98.14%
18.	Adagrad	Lr=0.01	MSE	97.52%

表(六)、Moblienet 訓練結果分析

	Optimizer	Learning rate	Loss function	Accuracy
1.	Adam	Lr=0.001	cross entropy	99.35%
2.	Adam	Lr=0.001	MSE	19.88%
3.	Adam	Lr=0.05	cross entropy	19.88%
4.	Adam	Lr=0.05	MSE	16.77%
5.	Adam	Lr=0.01	cross entropy	24.22%
6.	Adam	Lr=0.01	MSE	17.39%
7.	SGD	Lr=0.001	cross entropy	85.09%
8.	SGD	Lr=0.001	MSE	57.14%
9.	SGD	Lr=0.05	cross entropy	98.76%
10.	SGD	Lr=0.05	MSE	97.52%
11.	SGD	Lr=0.01	cross entropy	99.38%
12.	SGD	Lr=0.01	MSE	88.20%
13.	Adagrad	Lr=0.001	cross entropy	98.76%
14.	Adagrad	Lr=0.001	MSE	98.14%
15.	Adagrad	Lr=0.05	cross entropy	98.74%
16.	Adagrad	Lr=0.05	MSE	96.27%
17.	Adagrad	Lr=0.01	cross entropy	98.14%
18.	Adagrad	Lr=0.01	MSE	97.52%

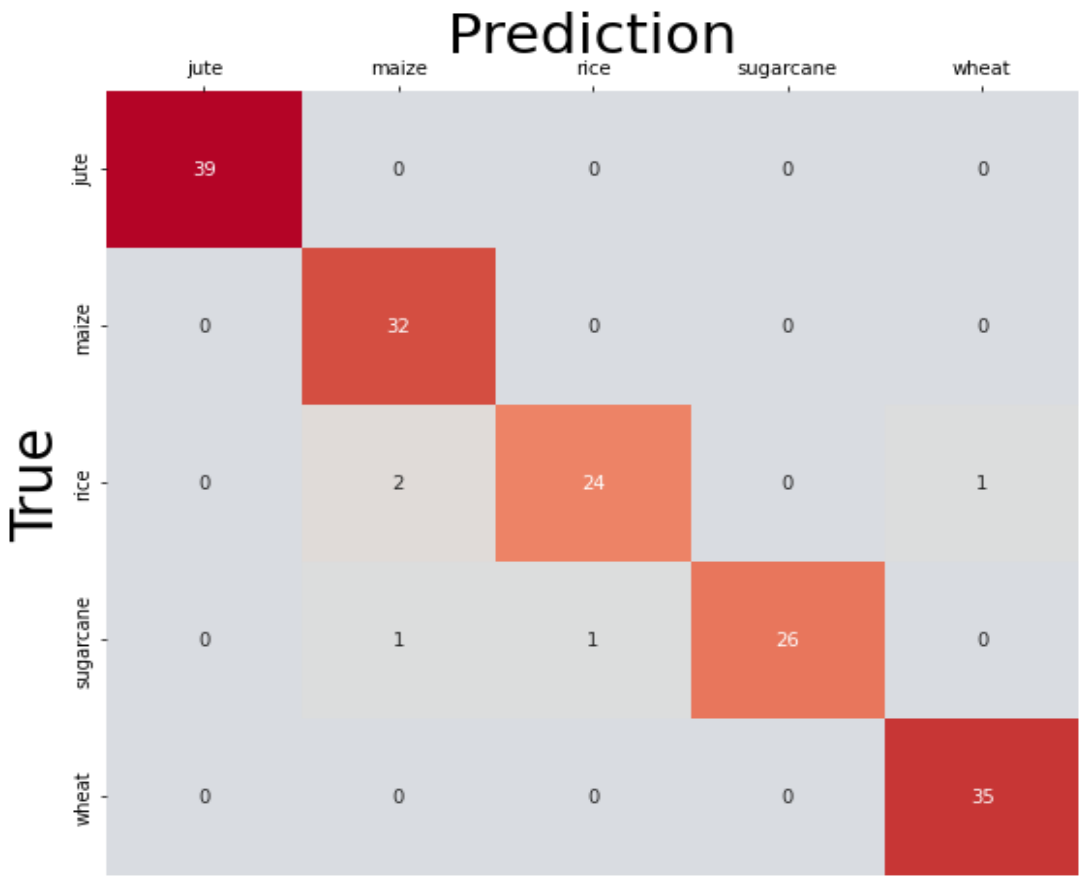
表(七)、Densenet 訓練結果分析

最後是以 Xception(optimizer=SGD,learning rate=0.05,loss function=cross entropy)



根據訓練結果分析進一步觀察模型在學習過程的 loss 和 accuracy，及模型在預測各個類別的 confusion matrix。

圖(十七)、最佳模型分析



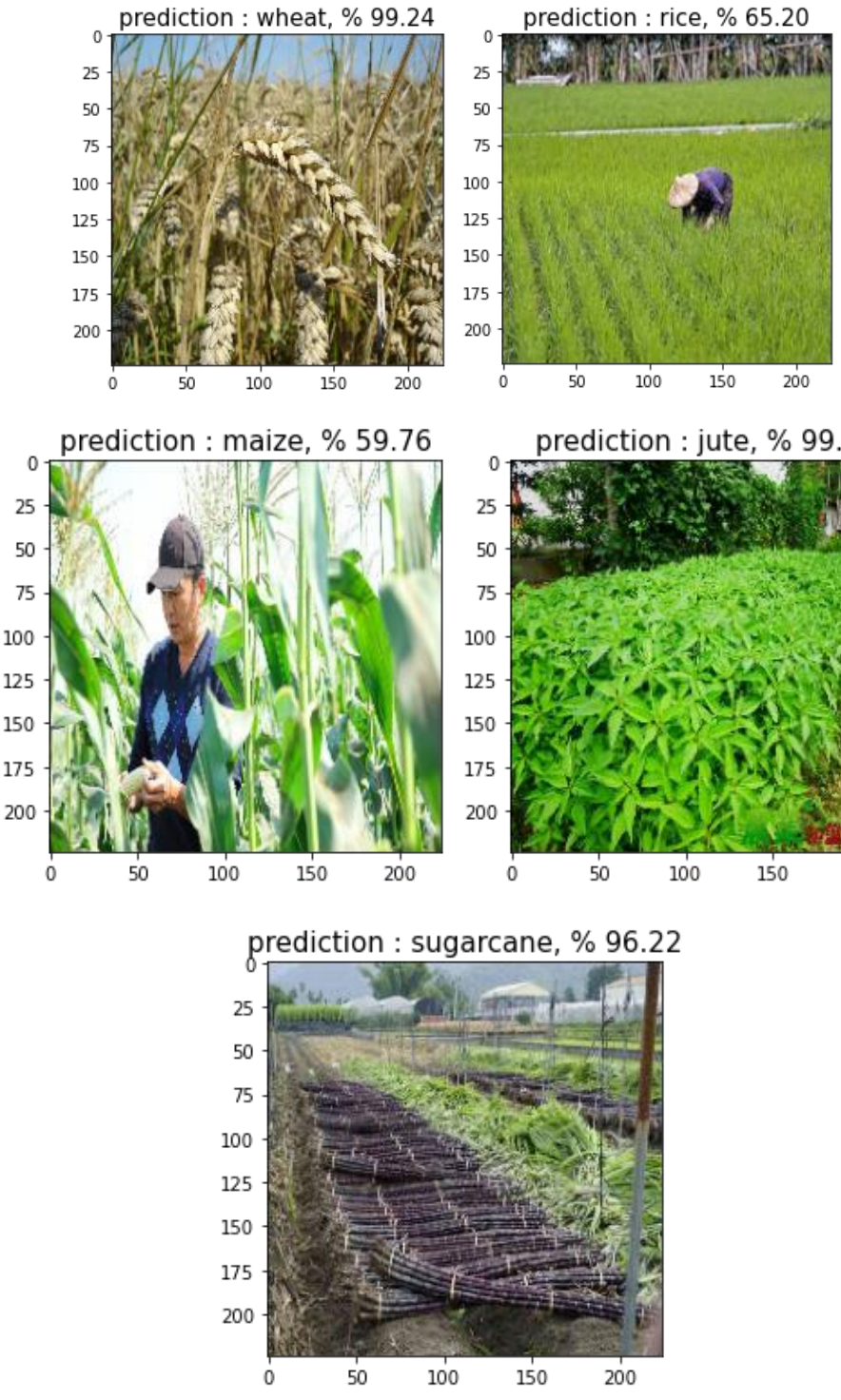
圖(十八)、confusion matrix

(八)、泛化性測試

```
test_pic=ImageScale("https://fll.cc/wp-content/uploads/2013/11/Wheat_close-up.jpg",size=244)
deepmodelpipeline("img_resize.jpg")

```

圖(十九)、網址匯入



圖(二十)、預測結果

五、 結論與展望

(一)、結論

在台灣現在的土地越來越珍貴，而且隨著人口不斷地老化，願意從事農業的年輕人越來越少，如何利用現有的科技力量結合農業的專業知識，必將是未來糧食作物發展的一大方向，而透過此次的報告，將過去需要人力巡田耗時耗力的工作，透過無人機配合後，將省去大量的人力成本及時間，以達到初步的智慧農業化。

圖(二十一)、各個圖片預測機率及類別

而除了能夠大量的減少人力成本及時間外，也將預測結果輸出成 CSV 檔後，能夠再透過分析後粗估目前農業的產品種植數量，能夠有效的控制市場供需，避免市場價格崩跌的狀況產生。

323	/kaggle/input/kag2/jute/jute001ahs.jpeg	0
324	/kaggle/input/kag2/jute/jute032arot.jpeg	0
325	/kaggle/input/kag2/jute/jute003a.jpeg	0
326	/kaggle/input/kag2/jute/jute002ahf.jpeg	0
327	/kaggle/input/kag2/jute/jute031arot.jpeg	0
328	/kaggle/input/kag2/jute/jute033a.jpeg	0
329	/kaggle/input/kag2/jute/jute028a.jpeg	0
330	/kaggle/input/kag2/jute/jute026ahs.jpeg	0
331	/kaggle/input/kag2/jute/jute040arot.jpeg	0
332	/kaggle/input/kag2/jute/jute029ahf.jpeg	0
333	/kaggle/input/kag2/jute/jute024ahf.jpeg	0
334	/kaggle/input/kag2/jute/jute028arot.jpeg	0
335	/kaggle/input/kag2/jute/jute016ahs.jpeg	0
336	/kaggle/input/kag2/jute/jute018a.jpeg	0
337	/kaggle/input/kag2/jute/jute005ahs.jpeg	0
338	/kaggle/input/kag2/jute/jute007ahf.jpeg	0
339	/kaggle/input/kag2/jute/jute034a.jpeg	0
340	/kaggle/input/kag2/jute/jute023a.jpeg	0
341	/kaggle/input/kag2/jute/jute020arot.jpeg	0
342	/kaggle/input/kag2/jute/jute019arot.jpeg	0
343	/kaggle/input/kag2/jute/jute005arot.jpeg	0
344	/kaggle/input/kag2/jute/jute040ahf.jpeg	0
345	/kaggle/input/kag2/jute/jute004ahf.jpeg	0
346	/kaggle/input/kag2/jute/jute029arot.jpeg	0
347	/kaggle/input/kag2/jute/jute033arot.jpeg	0
348	/kaggle/input/kag2/jute/jute039ahf.jpeg	0
349	/kaggle/input/kag2/jute/jute022a.jpeg	0
350	/kaggle/input/kag2/jute/jute011ahs.jpeg	0

圖(二十二)、分類結果

(二)、展望

臺灣以農立國，農民卻是要靠天吃飯。農業過去是支持經濟發展的重要產業，然而隨著人口結構改變，從農人口流失與高齡化，加上貿易自由化與氣候變遷加劇等，大幅影響了農事生產。政府透過科技計畫的資源投入，建立農業生產管理的新模式，目前完成主要基盤建置階段，並持續投入可操作模式建立與落地應用，期望能利用臺灣資通訊技術的產業優勢，結合物聯網、區塊鏈等技術，以跨領域技術協助農業的轉型與升級，解決農業發展困境。

六、 參考資料

- https://www.kaggle.com/aman2000jaiswal/agriculture-crop-images
- https://www.youtube.com/watch?v=xjrykYpaBBM&t=583s&ab_channel=GrandmaCan-%E6%88%91%E9%98%BF%E5%AC%A4%E9%83%BD%E6%9C%83
- https://mymkc.com/article/content/24493
- https://www.intelligentagri.com.tw/xmdoc/cont?xsmsid=0K303430418791365191&sid=0K304352448213621447
- https://keras.io/zh/applications/#_1
- http://ielab.ie.nthu.edu.tw/109_IIE_project/2/109IIE_proj2_5_word.pdf