王老先生有塊地

— 糧食作物分類 —

110034560 王勇盛

目錄
contents

目錄
contents

PART. 01

背景介紹

背景介紹

全球 2050 年人口預估達 75-105 億人。

人口

成本

原物料成本及人力成本持續上升，獲利下降

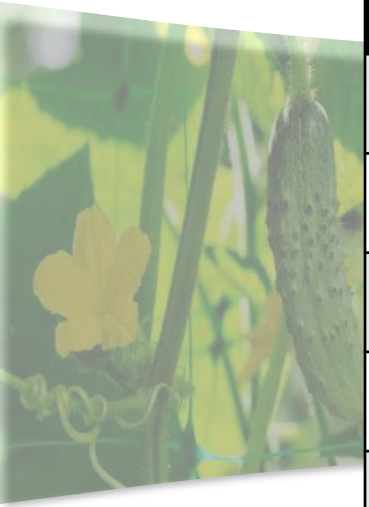
農村人口老化及少子化的影響，從事農業人力大幅短缺

老化
少子化

氣候

氣候變遷趨勢所致極端氣候日趨嚴重的困境下

背景介紹



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

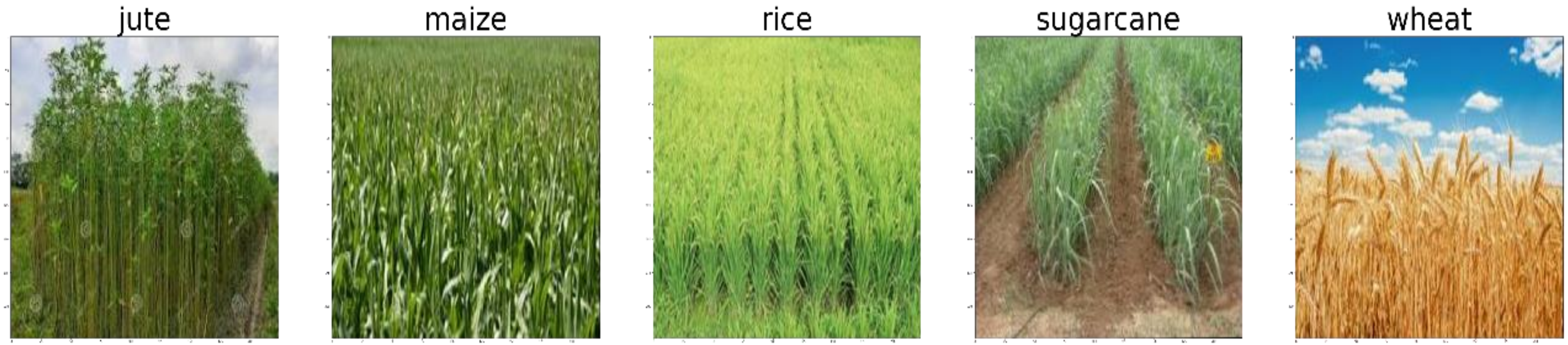
問題分析

目錄
contents

PART. 02

資料介紹、處理

資料介紹、處理



- **Agriculture crop images**

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

- **水平翻轉**

  對圖片進行水平翻轉並輸出儲存

- **資料的正規化**

  圖像在RGB通道都是0~255的整數，將這個值定為0~1之間的數

資料介紹、處理

	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

One Hot Encoding轉換

目錄
contents

PART. 03

模型介紹

模型介紹

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					

VGG19

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。

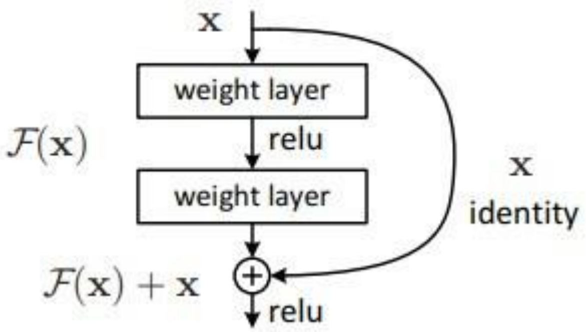
缺點：

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。

模型介紹

Resnet

優點:
1.ResNet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
2.直連通路與卷積層進行元素點(element-wise)相加，而非直接連接。這使某一層模型只需要學習輸入與輸出之間的殘差(residual)，大幅度降低了模型學習的難度。



3.殘差塊使用bottleneck形式，將一個3x3卷積核拆分成一組(1x1,3x3,1x1)卷積核，可減少模型參數使用量。
4.綜合前面3點，ResNet保障了深度神經網路的可訓練性，並且在深度模型上也可以避免使用過多參數量。

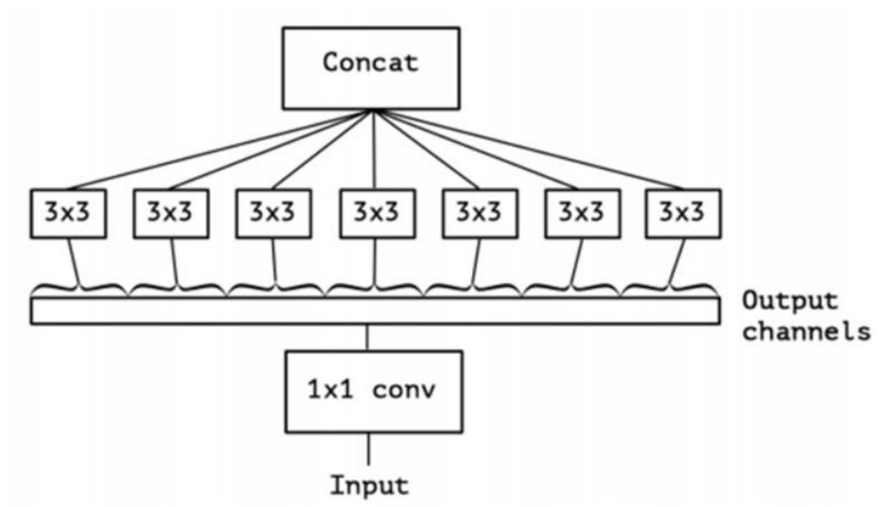
缺點:
1.模型推論速度慢：ResNet50比VGG16還慢。

	layers	parameter	top.5error
ResNet	20	0.27M	8.75
ResNet	32	0.46M	7.51
ResNet	44	0.66M	7.17
ResNet	56	0.85M	6.97
ResNet	110	1.7M	**6.43** (6.61±0.16)
ResNet	1202	19.4M	7.93

圖為ResNet作者在CIFAR-10上的實驗結果，ResNet-1202的表現甚至比ResNet-56還差。

模型介紹

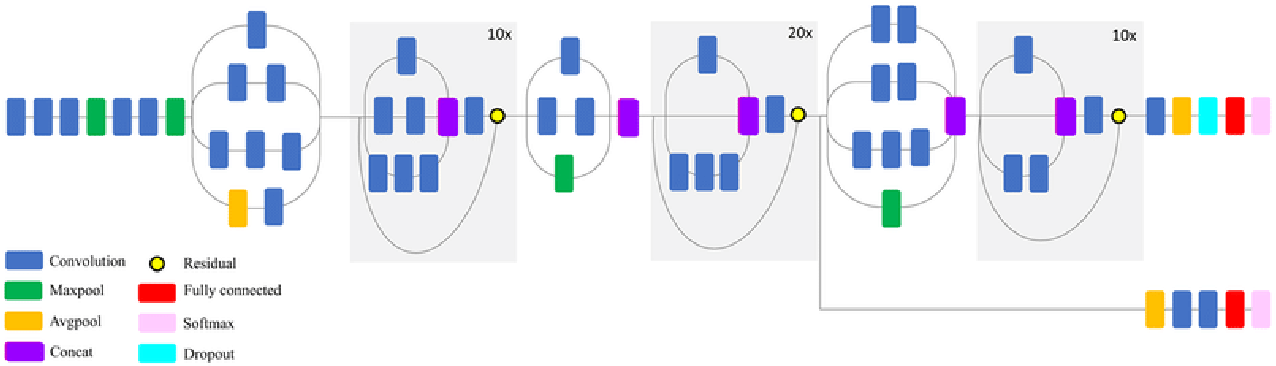
Xception



Xception net 中利用了 depthwise separable convolution (Extreme Inception)的架構，這個計算方式可以幫助該網路藉由較少的計算量得到卷積結果，加速模型的訓練並降低所需要的參數量。

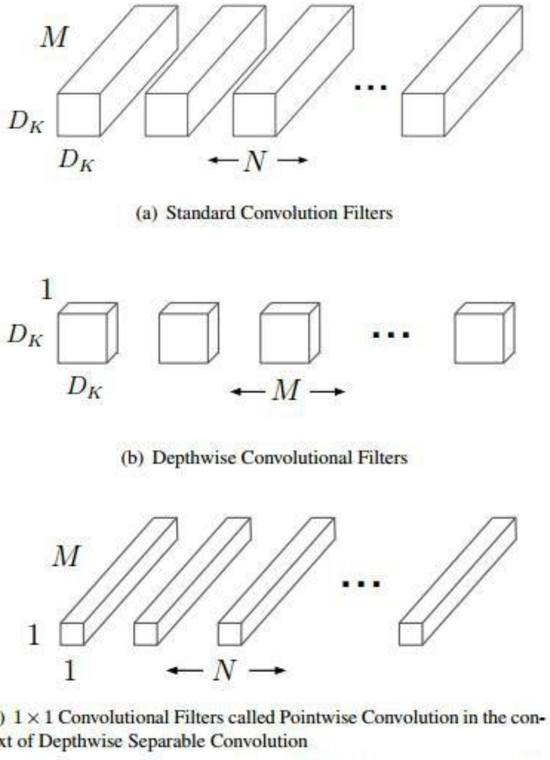
模型介紹

InceptionResNetV2



InceptionResNet V2 的架構，藉由增加 residual 的運算，
使得模型深度可以加深，讓模型參數可以增加

模型介紹



(a) Standard Convolution Filters

(b) Depthwise Convolutional Filters

(c) 1 × 1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

MobileNet

MobileNet網絡是由google團隊在2017年提出的，專注於移動端或者嵌入式設備中的輕量級CNN網絡。

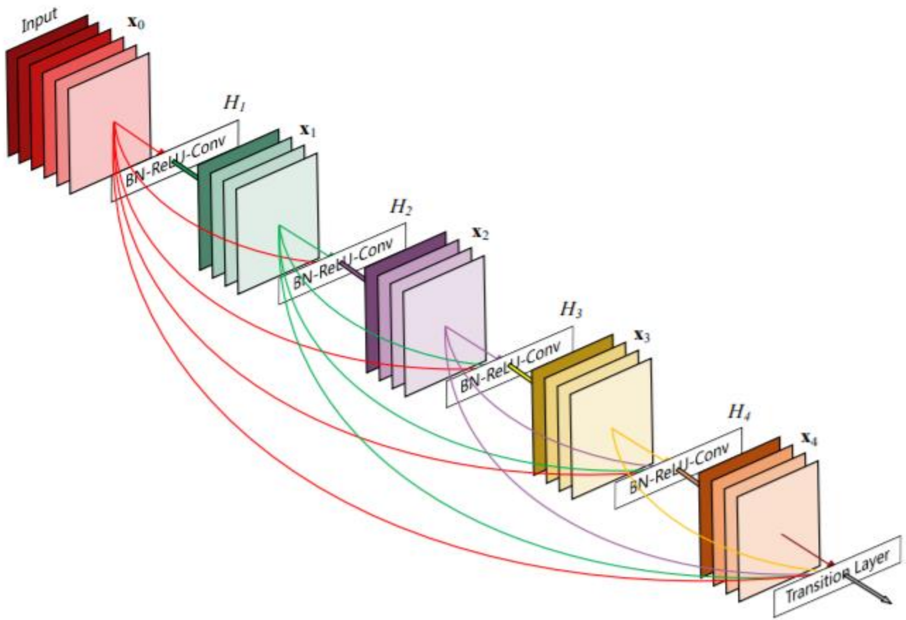
- 將模型輕量化，例如: 模型剪枝 (Model pruning)、模型量化 (Quantization)、權重共享 (Weight sharing) 、知識蒸餾 (Knowledge Distillation) 等
- 使用**輕量化的模型進行訓練**

DK x DK x M x N x DF x DF

DK x M x DF x DF + M x N x DF x DF

模型介紹

DenseNet



DenseNet 全名為 Densely Connected Convolutional Network。
由許多個Dense Block組成，每個Block皆採用bottleneck結構。
而Dense Block用了許多「層與層的連結」來達到特徵重用性。

目錄
contents

PART. 04

訓練過程

訓練過程

```
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
_________________________________________________________________
```

- 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)

訓練過程

```
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: 4,056,005
Non-trainable params: 23,587,712
```

- 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)

訓練過程

```
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
_________________________________________________________________
```

- 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)

訓練過程

```
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: 3,544,005
Non-trainable params: 54,336,736
_________________________________________________________________
```

- 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)

訓練過程

```
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
_________________________________________________________________
```

- 模型初始化
- 層級說明Keras model使用Keras 提供之CNN 模型(moblienet)
- 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)

訓練過程

```
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
_________________________________________________________________
```

- 模型初始化
- 層級說明Keras model使用Keras 提供之CNN 模型(densenet169)
- 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)

訓練過程

Optimizer	Learning rate	Loss function
• Adam • SGD • Adagrad	• 0.001 • 0.05 • 0.01	• Cross entropy • MSE

3X3X2=
18種

訓練過程

VGG19

Optimizer:[Adam,SGD,Adagrad]

Learning rate:[0.001,0.05,0.01]

Loss function:[cross entropy,MSE]

	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%

訓練過程

Resnet50

Optimizer:[Adam,SGD,Adagrad]
Learning rate:[0.001,0.05,0.01]
Loss function:[cross entropy,MSE]

	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%

訓練過程

Xception

Optimizer:[Adam,SGD,Adagrad]
Learning rate:[0.001,0.05,0.01]
Loss function:[cross entropy,MSE]

	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%

訓練過程

InceptionResNetV2

Optimizer:[Adam,SGD,Adagrad]

Learning rate:[0.001,0.05,0.01]

Loss function:[cross entropy,MSE]

	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%

訓練過程

Moblienet

Optimizer:[Adam,SGD,Adagrad]

Learning rate:[0.001,0.05,0.01]

Loss function:[cross entropy,MSE]

	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%

訓練過程

Densenet

Optimizer:[Adam,SGD,Adagrad]

Learning rate:[0.001,0.05,0.01]

Loss function:[cross entropy,MSE]

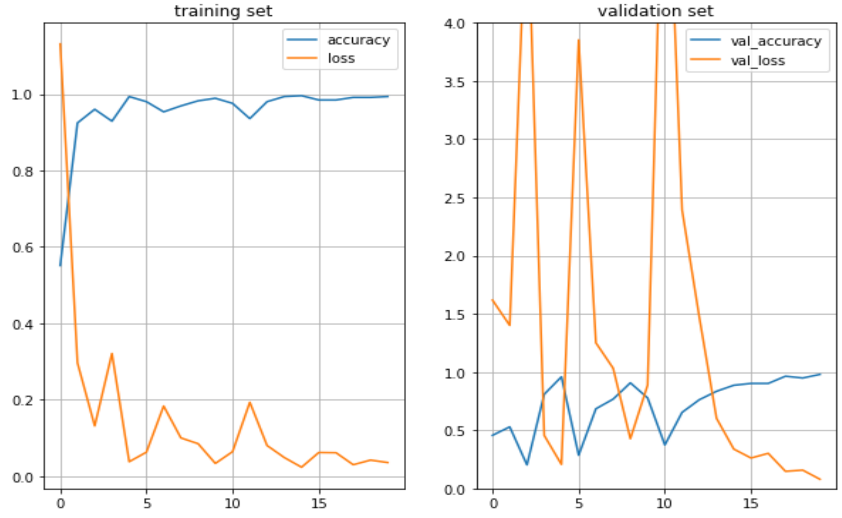
	Optimizer	Learning rate	Loss function	Accuracy
1.	Adam	Lr=0.001	cross entropy	99.35%
2.	Adam	Lr=0.001	MSE	19.88%
3.	Adam	Lr=0.05	cross entropy	19.88%
4.	Adam	Lr=0.05	MSE	16.77%
5.	Adam	Lr=0.01	cross entropy	24.22%
6.	Adam	Lr=0.01	MSE	17.39%
7.	SGD	Lr=0.001	cross entropy	85.09%
8.	SGD	Lr=0.001	MSE	57.14%
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%

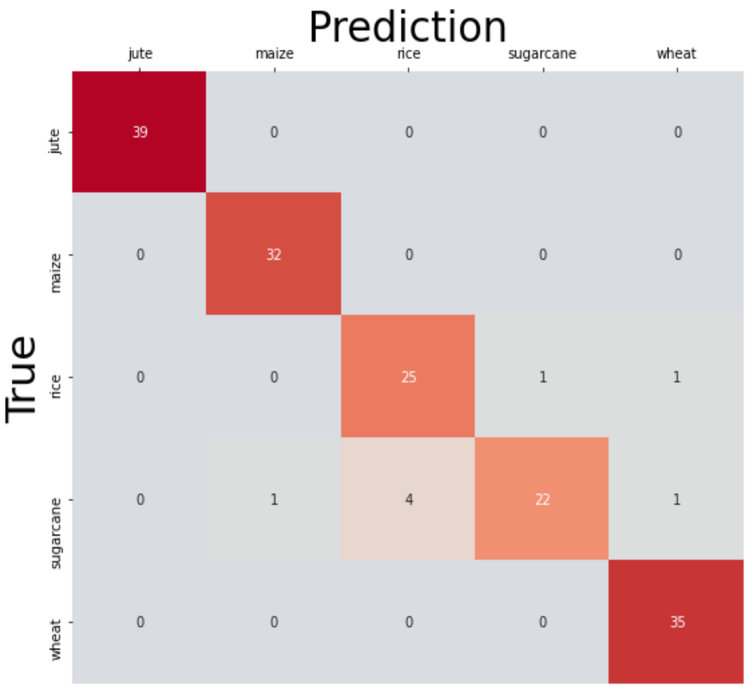
訓練過程

model	Parameter(M)	accuracy
VGG19	22	**91.93%**
Resnet50	27	**26.71%**
Xception	24	**99.38%**
InceptionResNetV2	57	**99.38%**
Moblienet	6	**98.76%**
Densenet	16	**99.35%**

訓練過程

Xception: optimizer=SGD,Learning rate=0.05,Loss function=cross entropy

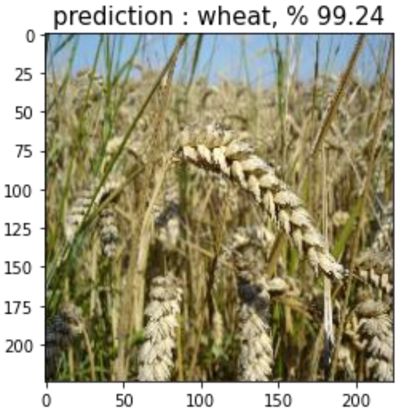


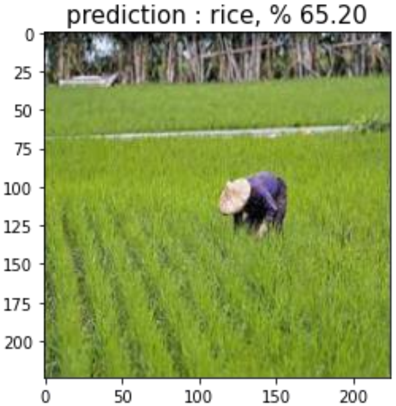


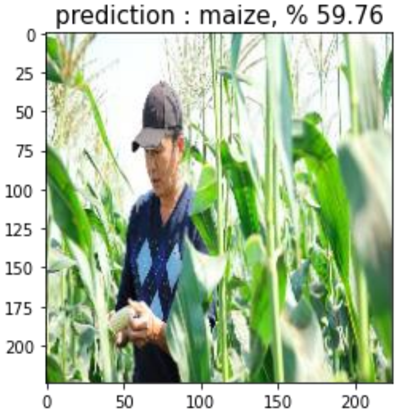
	jute	maize	rice	sugarcane	wheat
jute	39	0	0	0	0
maize	0	32	0	0	0
rice	0	0	25	1	1
sugarcane	0	1	4	22	1
wheat	0	0	0	0	35

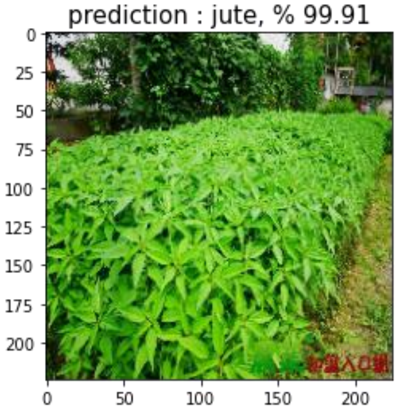
泛化性測試(Generalization)

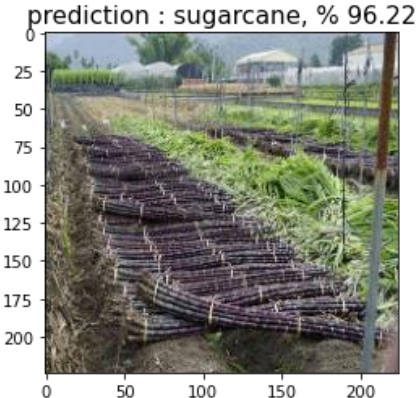
```
test_pic=ImageScale("https://fll.cc/wp-content/uploads/2013/11/Wheat_close-up.jpg",size=244)
deepmodelpipeline("img_resize.jpg")
```











目錄
contents

PART. 05

結論與展望

結論與展望

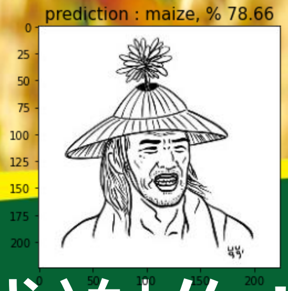
各個圖片預測機率及類別

輸出各個圖片預測結果

/kaggle/input/kag2/wheat/wheat024a.jpeg	4
/kaggle/input/kag2/wheat/wheat003.jpeg	4
/kaggle/input/kag2/wheat/wheat032ahs.jpeg	4
/kaggle/input/kag2/wheat/wheat011rot.jpeg	4
/kaggle/input/kag2/wheat/wheat021ahs.jpeg	4
/kaggle/input/kag2/wheat/wheat014ahs.jpeg	4
/kaggle/input/kag2/wheat/wheat0002a.jpeg	4
/kaggle/input/kag2/wheat/wheat037arot.jpeg	4
/kaggle/input/kag2/wheat/wheat004arot.jpeg	4
/kaggle/input/kag2/wheat/wheat020ahs.jpeg	4
/kaggle/input/kag2/wheat/wheat0001ahf.jpeg	4
/kaggle/input/kag2/wheat/wheat006arot.jpeg	4
/kaggle/input/kag2/wheat/wheat0002ahf.jpeg	4
/kaggle/input/kag2/jute/jute025ahs.jpeg	0
/kaggle/input/kag2/jute/jute017arot.jpeg	0
/kaggle/input/kag2/jute/jute001ahs.jpeg	0
/kaggle/input/kag2/jute/jute032arot.jpeg	0
/kaggle/input/kag2/jute/jute003a.jpeg	0
/kaggle/input/kag2/jute/jute002ahf.jpeg	0
/kaggle/input/kag2/jute/jute031arot.jpeg	0

結論與展望

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，目前完成主要基本建置階段，並持續投入可操作模式建立與實際應用，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


prediction : maize, % 78.66
0
25
50
75
100
125
150
175
200
0
50
100
150
200

王老先生感謝你!