智慧化企業整合

以 SSO-CNN 和 VGG16 辨識肺部 X 光照片

李旖庭 109034565

指導老師：邱銘傳教授

目錄

一、前言

1. 背景介紹----------p.3

2. 5W1H----------p.3

二、模型介紹

1. VGG16----------p.4

2. SSO-CNN----------p.5

三、資料前分析

1. 資料類型----------p.6

2. 資料降維----------p.6

四、模型分析

1. 模型一:VGG16----------p.10

2. 模型二:SSO-CNN----------p.13

五、結果與討論

1. 結論----------p.20

2. 未來展望----------p.20

一、 前言

1. 背景介紹

目前正被新型冠狀病毒(COVID-19)全球性疫情肆虐，感染人數大約 7990 萬人，且死亡人數已達到 175 萬，狀況日益嚴重，而新型冠狀病毒(COVID-19)的一大病徵即是肺炎，辨識肺炎成為新的一大議題，各大醫院與科研單位無不合作防堵疫情擴散。在防疫的過程中，篩檢過程被醫院重視，其中胸腔 X-ray 是一項重要的檢查指標，主要用來判斷是否出現肺炎等徵狀。然而當受檢驗的患者數量過多，或醫護人員人手不足，無法即時做出診斷或是疏忽一些病灶，就容易造成防疫上的缺口。

2. 5W1H

醫療科學相關的資料向來十分豐富。幾世紀以來，科學家透過各式各樣的方式收集資料，無論是來自實驗、縱貫（貫時性）研究、研究調查，或者問診病患的病歷紀錄等日常活動。然而，新型冠狀病毒（COVID-19）的疫情衝擊大幅改變了醫療體系。

即使目前規模最大的醫療研究計畫已致力於消化數十萬則病例，疫情仍難以避免地影響了全球數十億人。大量的資料從各種不同的管道雪片般飛來，而此時，正是利用機器學習 SSO-CNN 和 VGG16 辨識肺部 X 光照片登場對抗病毒的時刻，利用 5W1H 分析並了解問題和思考解決問題的方法:

- Who ： 醫療人員
- What ： 解決大量肺炎 X 光照片
- Why ： 更有效率辨識肺炎
- When ： 患者至醫院治療時
- Where ： 醫院、科研中心
- How： 資料分析、深度學習、機器學習(SSO-CNN 和 VGG16)

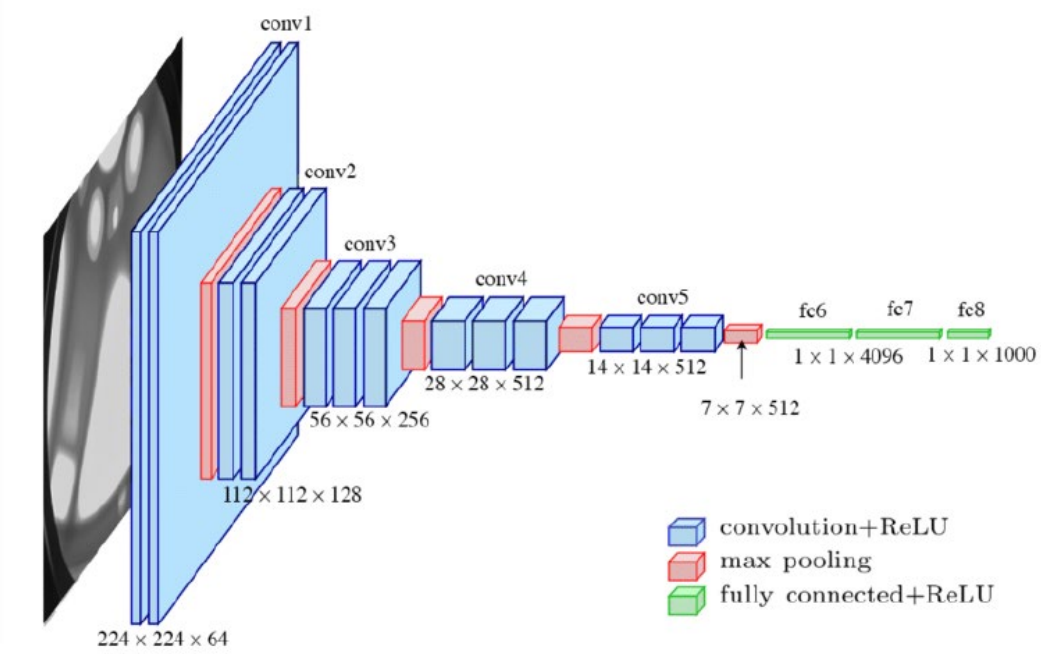
二、 模型介紹

1. VGG16

ImageNet 每年舉辦的競賽(ILSVRC)這幾年產生了不少的 CNN 冠軍，而 VGG16 就是 2014 年的亞軍，承襲了 AlexNet 的思路，建立更多層的模型，達到 16 個隱藏層。關於 VGG，共使用了 ImageNet 的 100 萬張圖片，共 1000 種類別，幾乎涵蓋日常生活看到的事物，例如動物、交通工具...等，訓練出來的模型，就變成一種『通用解決方案』(Generic Solution)，如果要辨識照片內事物屬於這 1000 類，例如貓、狗、大象等，就可以直接拿 VGG 模型來用了，反之，如果要辨識的內容不屬於 1000 類，也可以換掉 input 卷積層，只利用中間層萃取的特徵，這種能力稱為『Transfer Learning』。而之所以稱為 VGG16，則是因為裡面共有 13 個卷積層和 3 個全連階層。

而 CNN 的改善主要有兩種，一種是使用更小的卷積或是更小的 stride，另一種主要是利用不同的 Data Augmentation，例如：Multiple Scale training 等。而在 VGG 當中，將兩種概念合併在一起，提供一個更深且結果更穩定的網路。當中最重要的概念是，VGG 使用了大量的 3x3 卷積，我認為將較大的卷積抽換成較小的卷積可讓 Receptive field 提高，也就是指資訊量的增加。除此之外，使

用較多較小的卷積成可以提高網路的非線性結構，同時也可以減少需要訓練的參數量，下圖為 VGG16 的示意圖



2. SSO-CNN

SSO 全名為 Simplified Swarm Optimization，又稱為簡化群演算法，簡單概述 SSO 演算法可以用以下公式來表達

$$x_{t+1,i,j}=\begin{cases} p_{gBest,j} & \text{if } \rho_{[0,1]}\in[0,C_g) \\ p_{i,j} & \text{if } \rho_{[0,1]}\in[C_g,C_p) \\ x_{t,i,j} & \text{if } \rho_{[0,1]}\in[C_p,C_w) \\ x & \text{if } \rho_{[0,1]}\in[C_w,\ 1] \end{cases}$$

也就是說，在每一次迭代過程中，都會隨機產生一個 0~1 的隨機數，並透過該數來決定下一次的演算策略。借公式來解釋，當隨機數介於 0 到C_g之間，則採用目前最佳解的參數；當C_g介於到C_P之間，使用當次迭代中的最佳解的參數；當介於C_P到C_w之間，則保留此次的參數，直接進到下一次迭代；最後，當介於C_w到 1 之間，則重新隨機產生一個參數作為取代。而以上 C 的參數都是實驗者必須事先給定的，這四個範圍通常會依序遞減，由寬到窄，而為什麼會有第四種情況，就是確保解可以跳脫區域最佳解，進行收斂到全域最佳解的一個機制。

而本實驗所採用的是改進式簡化群演算法，又可以稱為 Improved SSO，簡稱 ISSO。將演化過程縮為三類，第一類依舊是採用當前最佳解，第二類為保留原解，第三類則是隨機產生新解作為取代。而完整的演算法步驟由以下表格所呈現

步驟一	產生初始解 X_i^0，並計算適應度函數 $F(X_i^0), i=1,2,...Par$
步驟二	令 $t=1$，令 $F(G)$為 $F(X_i^0)$中最優良的解，$i=1,2,...Par$
步驟三	令 $i=1$.
步驟四	演化出 X_i^{t+1}，並計算適應度函數 $F(X_i^{t+1})$.
步驟五	若 $F(X_i^{t+1})$比 $F(X_i^t)$優良，則令 $X_i^t = X_i^{t+1}$；若無，則 X_i^t保持不變並跳至**步驟七**.
步驟六	若 $F(X_i^{t+1})$ 比 $F(G_i)$ 優良，令 $G_i = X_i^{t+1}$.
步驟七	若 $i<$PAR，令 $i=i+1$ 且跳回**步驟四**
步驟八	若 $t=MaxIter$，程式終止；否則 $t=i+1$ 且跳回**步驟三**

三、 資料前處理

1. 資料類型

資料為肺部 X 光照片，原始像素為 224x224，共有 5856 筆，分為以下三種:

- 訓練資料：共 5216 張，1341 張正常，3875 張肺炎
- 驗證資料：共 16 張，8 張正常，8 張肺炎
- 測試資料：共 624 張，234 張正常，390 張肺炎

由數據來看在不同資料集之間的分布還算均勻，不會有過度極端的情況發生，是可以接受的範圍

2. 資料降維

【MPCA】Algorithm

- **Step1:** 使用 tensorflow 讀入 data(training data, validation data, test data)
- **Step2:** 將 training data 依據 row 及 column，進行 unfold
- **Step3:** 分別進行 SVD 分解，取 column(row) mode eigenvalues 以及 eigenvectors
- **Step4:** 透過計算 validation data 的 reconstruction error，找出

適當的 rank

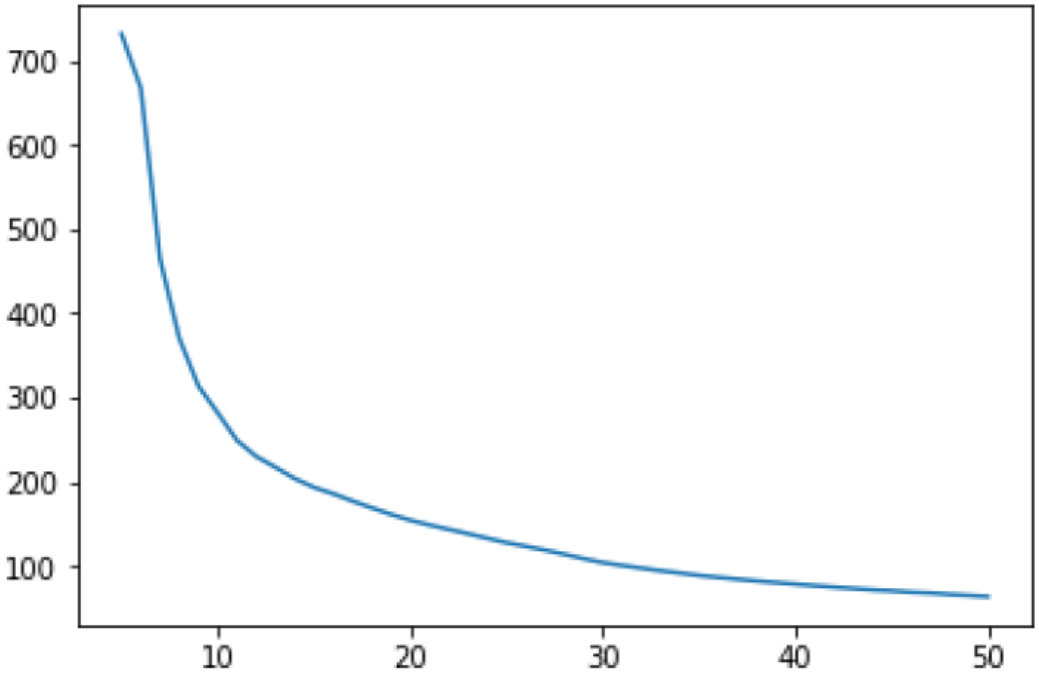
- **Step5:** 畫出 basis plot

- **Step6:** 隨機抽取 test data，進行 reconstruction

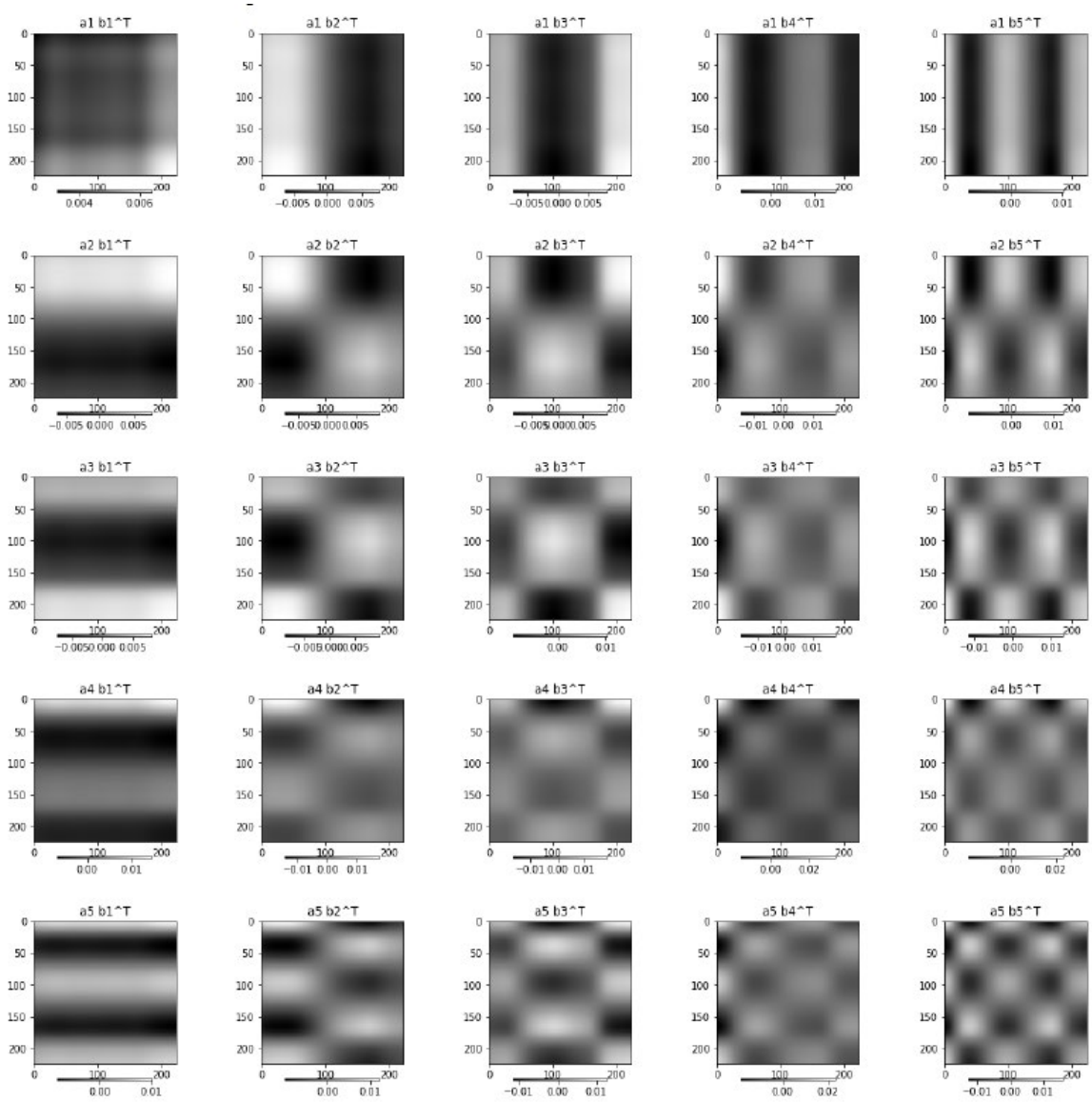
首先將 training data X <5216x224x224> 進行 unfold，變成 X1,X2 <224x1168384>。接著進行 SVD 分解，為了找出適當的 rank，本實驗將 rank 從 5 遍到 50 遍，計算不同 rank 情況下的 error，並對每一個 rank 進行 5 次的 iteration，使 error 趨於穩定。而在每一個 iteration 中，更新 A 及 B，計算 validation data 的 reconstruction error，並針對每一個 rank 所計算出來的 reconstruction error，計算 reduced rate。以下為 reduced rate 的公式：

reduced rate=(advance error−error) / advance error

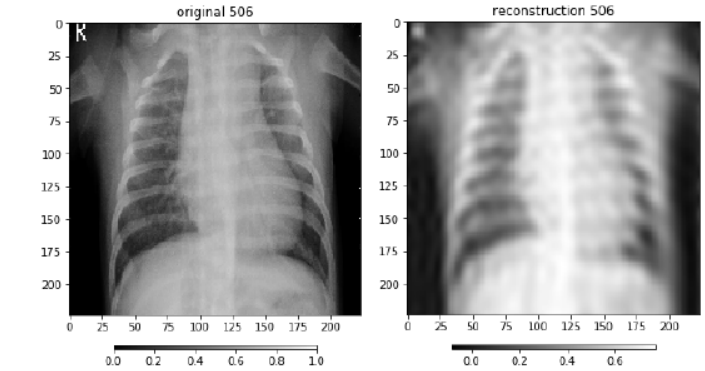
也就是說，該值代表在下一個 rank 所計算出來的 error，當中的減少量佔前一次 error 的比率。在此本實驗主觀預設，當 reduced rate 低於 3% 時，則挑選該 rank。透過結果，選擇的是 rank 為 30。下圖為 error 隨著 rank 的增加而降低的趨勢圖：

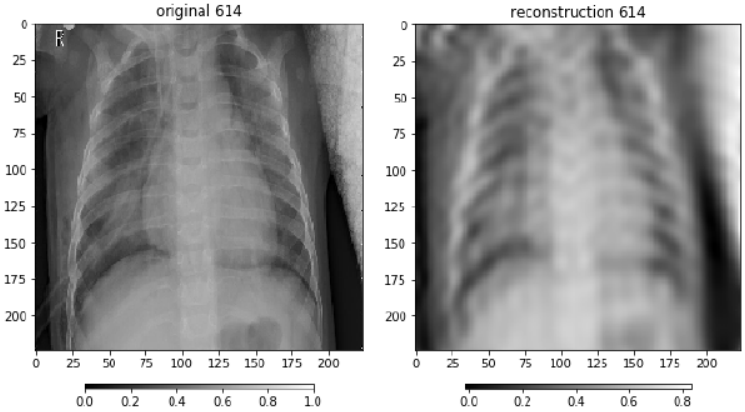


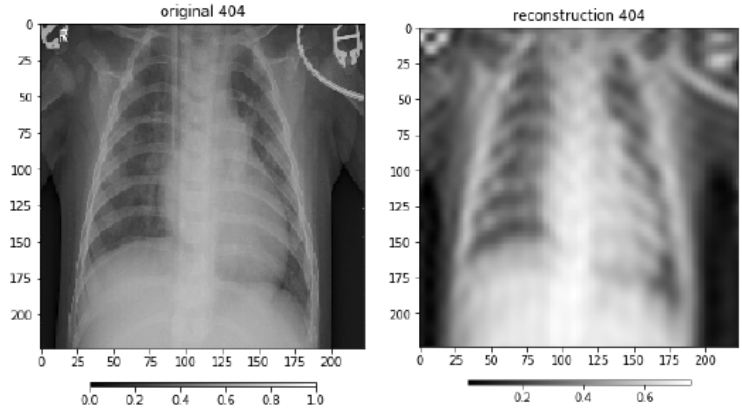
接著，畫出 basis plot

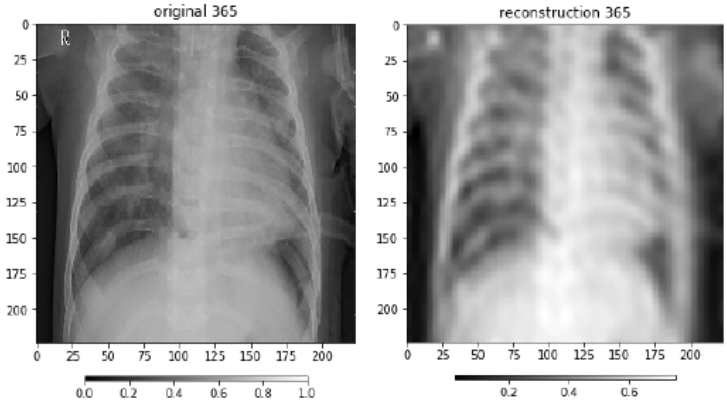


在最後，隨機挑取 4 筆 test data，進行 reconstruction。


original 506
reconstruction 506


original 614
reconstruction 614


original 404
reconstruction 404


original 365
reconstruction 365

四、 模型分析

1. VGG16

本實驗採用手動建立 VGG16-model，結構為下圖所示

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 64)        640
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 30, 30, 64)        36928
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 15, 15, 128)       73856
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 15, 15, 128)       147584
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 128)         0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 7, 7, 256)         295168
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 7, 7, 256)         590080
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 7, 7, 256)         590080
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 256)         0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 3, 3, 512)         1180160
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 3, 3, 512)         2359808
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 3, 3, 512)         2359808
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 1, 1, 512)         0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 1, 1, 512)         2359808
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 1, 1, 512)         2359808
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 1, 1, 512)         2359808
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 4096)              2101248
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              16781312
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 8194
=================================================================
Total params: 33,604,290
Trainable params: 33,604,290
Non-trainable params: 0
```

所使用的 loss function 為 categorical_crossentropy，optimizer 選擇 RMS，metrics 為 accuracy，epochs 次數為 3。training model 的過程以及最後預測 testing data 的結果如下

```
Epoch 1/3
5216/5216 [==============================] - 135s 26ms/sample - loss: 0.6033 - categorical_crossentropy: 0.6033 - acc: 0.7414 -
val_loss: 0.8171 - val_categorical_crossentropy: 0.8171 - val_acc: 0.5000
Epoch 2/3
5216/5216 [==============================] - 134s 26ms/sample - loss: 0.5752 - categorical_crossentropy: 0.5752 - acc: 0.7429 -
val_loss: 0.7638 - val_categorical_crossentropy: 0.7638 - val_acc: 0.5000
Epoch 3/3
5216/5216 [==============================] - 138s 26ms/sample - loss: 0.5729 - categorical_crossentropy: 0.5729 - acc: 0.7429 -
val_loss: 0.8707 - val_categorical_crossentropy: 0.8707 - val_acc: 0.5000
```

```
624/624 [==============================] - 4s 6ms/sample - loss: 0.7173 - categorical_crossentropy: 0.7173 - acc: 0.6250
```

Training accuracy 為 0.7429，Testing accuracy 為 0.6250。在經過幾次的試驗之後，發現 testing data 的準確率一直卡在 62.5%，推測可能過程當中出了以下問題:

(1) 不適合的 model：由於不同模型適用對象不同，推測預測結果不如預期，就是 model 對於此次的數據不適合。對此，我嘗試了以下幾種作法：

A. 更改 model 的 optimizer，將原來的 RMS 更改為 adadelta

B. 更改 Dense 層的 activation function，從 relu 改為 sigmoid

C. 更改 model 的 optimizer，參照網路上針對此問題所建議的 SGD(隨機梯度下降)，但以上辦法皆無法改善該問題

(2) Overfitting：參數過多、模型過於複雜都有可能導致 overfitting，假設樣本數不夠或者分布不均，皆有可能會導致預測能力不佳，而該模型的參數數量高達 3300 多萬，可以合理懷疑此為造成準確率不高的原因之一。對此，我首先試圖減少卷積層的數量(雖然這可能就不是 VGG16)，將後面九層卷積層取消，以及全連階層內的數量降低至 1024，進行訓練及測試，其模型結構以及訓練預測結果如下圖所示

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_13 (Conv2D)           (None, 30, 30, 64)        640
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 30, 30, 64)        36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 15, 15, 64)        0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 15, 15, 128)       73856
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 15, 15, 128)       147584
_________________________________________________________________
flatten_1 (Flatten)          (None, 28800)             0
_________________________________________________________________
dense_3 (Dense)              (None, 1024)              29492224
_________________________________________________________________
dense_4 (Dense)              (None, 1024)              1049600
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 2050
=================================================================
Total params: 30,802,882
Trainable params: 30,802,882
Non-trainable params: 0
_________________________________________________________________
```

```
Train on 5216 samples, validate on 16 samples
Epoch 1/3
5216/5216 [==============================] - 61s 12ms/sample - loss: 0.5960 - categorical_crossentropy: 0.5960 - acc: 0.8148 -
val_loss: 0.7935 - val_categorical_crossentropy: 0.7935 - val_acc: 0.6250
Epoch 2/3
5216/5216 [==============================] - 62s 12ms/sample - loss: 0.2305 - categorical_crossentropy: 0.2305 - acc: 0.9139 -
val_loss: 0.3301 - val_categorical_crossentropy: 0.3301 - val_acc: 0.8750
Epoch 3/3
5216/5216 [==============================] - 63s 12ms/sample - loss: 0.1707 - categorical_crossentropy: 0.1707 - acc: 0.9375 -
val_loss: 0.6321 - val_categorical_crossentropy: 0.6321 - val_acc: 0.8125
```

```
624/624 [==============================] - 1s 2ms/sample - loss: 0.5985 - categorical_crossentropy: 0.5985 - acc: 0.7853
```

Training accuracy 為 0.9375，Testing accuracy 為 0.7853，比起之前的模型有了顯著的提升，但我們仍然可以發現，需要預測的參數依舊高達 3000 萬個，其中產生最多參數的層為全連接層，因此我接著試圖將全連接層內的數量再降低成 256，看是否會繼續提升準確率

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_17 (Conv2D)           (None, 30, 30, 64)        640
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 30, 30, 64)        36928
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 15, 15, 64)        0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 15, 15, 128)       73856
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 15, 15, 128)       147584
_________________________________________________________________
flatten_2 (Flatten)          (None, 28800)             0
_________________________________________________________________
dense_6 (Dense)              (None, 256)               7373056
_________________________________________________________________
dense_7 (Dense)              (None, 256)               65792
_________________________________________________________________
dense_8 (Dense)              (None, 2)                 514
=================================================================
Total params: 7,698,370
Trainable params: 7,698,370
Non-trainable params: 0
_________________________________________________________________
```

```
Train on 5216 samples, validate on 16 samples
Epoch 1/3
5216/5216 [==============================] - 31s 6ms/sample - loss: 0.4447 - categorical_crossentropy: 0.4447 - acc: 0.8096 -
val_loss: 0.2901 - val_categorical_crossentropy: 0.2901 - val_acc: 0.8750
Epoch 2/3
5216/5216 [==============================] - 29s 6ms/sample - loss: 0.2096 - categorical_crossentropy: 0.2096 - acc: 0.9162 -
val_loss: 0.6605 - val_categorical_crossentropy: 0.6605 - val_acc: 0.7500
Epoch 3/3
5216/5216 [==============================] - 29s 6ms/sample - loss: 0.1619 - categorical_crossentropy: 0.1619 - acc: 0.9356 -
val_loss: 0.5073 - val_categorical_crossentropy: 0.5073 - val_acc: 0.7500
```

```
624/624 [==============================] - 1s 2ms/sample - loss: 0.7691 - categorical_crossentropy: 0.7691 - acc: 0.7468
```

Training accuracy 為 0.9356，Testing accuracy 為 0.7468，並沒有顯著的升降。

對於上面所做的這些操作，讓我有一個想法，比起這樣手動、主觀的去調整模型架構，可能會造成一些人為的判斷錯誤。既然如此，我將結合一些尋求最佳解的演算法，替模型參數進行估計，也就是我接下來要展示的第二個方法 SSO-CNN。

2. SSO-CNN

在本實驗，該方法所使用的基底模型為下圖所示

```
Model: "sequential_100"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_200 (Conv2D)          (None, 30, 30, 256)       4352
_________________________________________________________________
conv2d_201 (Conv2D)          (None, 30, 30, 16)        36880
_________________________________________________________________
max_pooling2d_100 (MaxPoolin (None, 7, 7, 16)          0
_________________________________________________________________
dropout_200 (Dropout)        (None, 7, 7, 16)          0
_________________________________________________________________
flatten_100 (Flatten)        (None, 784)               0
_________________________________________________________________
dense_200 (Dense)            (None, 256)               200960
_________________________________________________________________
dropout_201 (Dropout)        (None, 256)               0
_________________________________________________________________
dense_201 (Dense)            (None, 2)                 514
=================================================================
Total params: 242,706
Trainable params: 242,706
Non-trainable params: 0
_________________________________________________________________
```

(參數僅參考，在此僅表達 model 架構，參數是接下來要估計的)

使用該模型的動機是，運算機器效能有限，無法替大型的 model 進行參數最佳化(例如 VGG16)，因此用一個相較起來較為簡單的 model，不影響該實驗想要表達的內容，目的為如何在不更動 model 架構下，透過調整參數使準確率最大化。

接著將 model 中所需設定的參數，透過改進式的 SSO（以下簡稱 ISSO）進行最佳化，參數包含

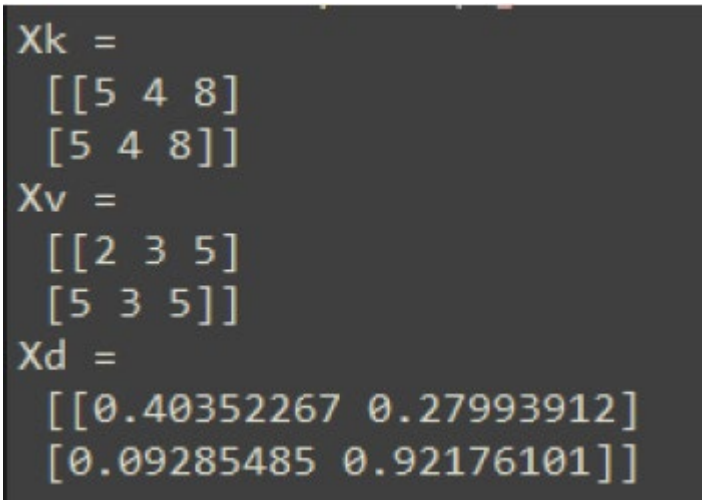
- Convolution Layer：filter 以及 kernel size，因有兩卷積層，所以有 4 個參數。
- MaxPooling Layer：pool_size，一個參數。
- Dropout Layer：probability，兩個參數。
- Dense Layer：units，一個參數。

共有八個參數需要透過 ISSO 進行最佳化求解，在本實驗中，共跑 50 個 generation，每一個 generation 進行兩次的求解。每一次的求解只會進行一次 CNN model fit。為使本實驗更加明瞭，將 50 次的過程中，每五次便擷取一次結果，方便比較。在進行第一次 generation

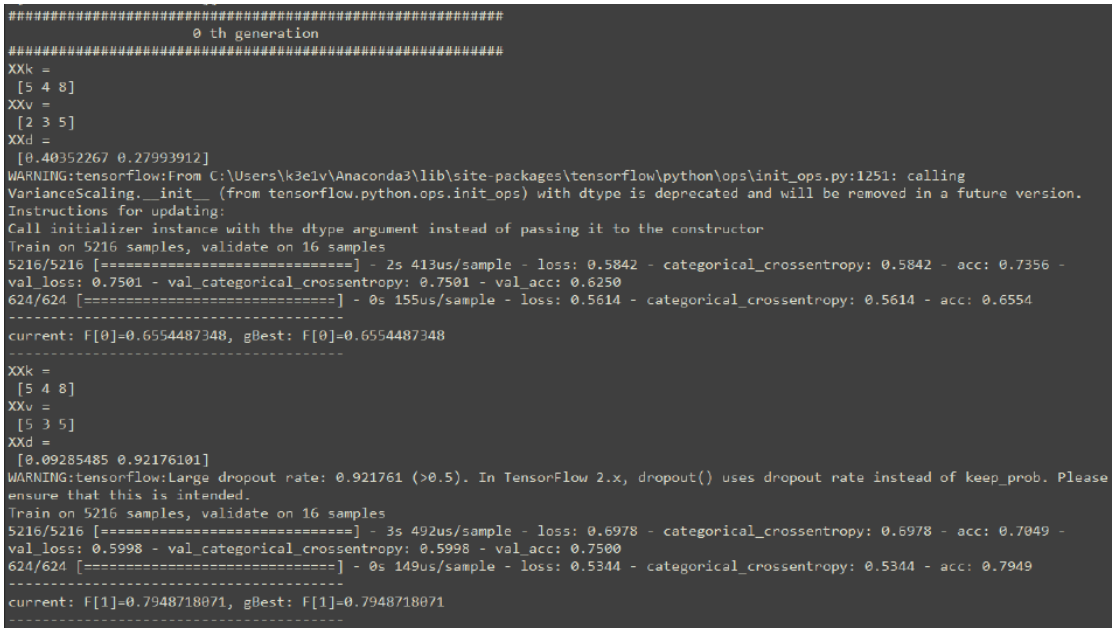
時，初始隨機產生指定範圍內的參數

- Filter & units：2^3 ~ 2^9 之間 (必為 2 的次方)
- Kernel size & Pool size：2 ~ 6 之間 (必為整數)
- Dropout Probability：0 ~ 1 之間 (任意值)

以下為初次隨機產生的參數



以及每五次所記錄的結果，XX 開頭為該次進入 CNN model 的參數，XXk 三個數字表 Filter 以及 Dense 的 units，XXv 的三個數字表 kernel size 以及 pool size，XXd 表兩次 dropout 各自的機率。Cuurent 後之值為當次的準確率，gBest 後之值為當前最佳的準確率，以下展示每五次 generation 所得到的結果



```
############################################################
                      5  th generation
############################################################
<sol=%d> SSO_updated parameters:
XXk =
 [5 6 8]
XXv =
 [4 2 4]
XXd =
 [0.09285485 0.27993912]
Train on 5216 samples, validate on 16 samples
5216/5216 [==============================] - 8s 1ms/sample - loss: 0.6701 - categorical_crossentropy: 0.6701 - acc: 0.7590 -
val_loss: 0.5272 - val_categorical_crossentropy: 0.5272 - val_acc: 0.6875
624/624 [==============================] - 0s 265us/sample - loss: 0.5203 - categorical_crossentropy: 0.5203 - acc: 0.7516
Xk =
 [[5 4 8]
 [5 4 8]]
Xv =
 [[2 3 5]
 [4 3 4]]
Xd =
 [[0.40352267 0.27993912]
 [0.09285485 0.80284623]]
----------------------------------------
current: F[0]=0.6554487348, gBest: F[1]=0.8285256624
----------------------------------------
<sol=%d> SSO_updated parameters:
XXk =
 [5 4 6]
XXv =
 [4 3 4]
XXd =
 [0.09285485 0.80284623]
Train on 5216 samples, validate on 16 samples
5216/5216 [==============================] - 3s 570us/sample - loss: 0.6030 - categorical_crossentropy: 0.6030 - acc: 0.7253 -
val_loss: 0.6769 - val_categorical_crossentropy: 0.6769 - val_acc: 0.6250
624/624 [==============================] - 0s 160us/sample - loss: 0.5434 - categorical_crossentropy: 0.5434 - acc: 0.6603
Xk =
 [[5 4 8]
 [5 4 8]]
Xv =
 [[2 3 5]
 [4 3 4]]
Xd =
 [[0.40352267 0.27993912]
 [0.09285485 0.80284623]]
----------------------------------------
current: F[1]=0.8285256624, gBest: F[1]=0.8285256624
----------------------------------------
```

```
############################################################
                      10  th generation
############################################################
<sol=%d> SSO_updated parameters:
XXk =
 [5 4 8]
XXv =
 [4 3 4]
XXd =
 [0.09285485 0.27993912]
Train on 5216 samples, validate on 16 samples
5216/5216 [==============================] - 3s 576us/sample - loss: 0.5720 - categorical_crossentropy: 0.5720 - acc: 0.7602 -
val_loss: 1.0555 - val_categorical_crossentropy: 1.0555 - val_acc: 0.5625
624/624 [==============================] - 0s 171us/sample - loss: 0.8384 - categorical_crossentropy: 0.8384 - acc: 0.6250
Xk =
 [[5 4 8]
 [5 4 8]]
Xv =
 [[2 3 5]
 [4 3 4]]
Xd =
 [[0.40352267 0.27993912]
 [0.09285485 0.80284623]]
----------------------------------------
current: F[0]=0.6554487348, gBest: F[1]=0.8285256624
----------------------------------------
<sol=%d> SSO_updated parameters:
XXk =
 [5 4 8]
XXv =
 [4 3 4]
XXd =
 [0.09285485 0.80284623]
Train on 5216 samples, validate on 16 samples
5216/5216 [==============================] - 3s 639us/sample - loss: 0.5697 - categorical_crossentropy: 0.5697 - acc: 0.7464 -
val_loss: 0.5067 - val_categorical_crossentropy: 0.5067 - val_acc: 0.8125
624/624 [==============================] - 0s 152us/sample - loss: 0.4582 - categorical_crossentropy: 0.4582 - acc: 0.8125
Xk =
 [[5 4 8]
 [5 4 8]]
Xv =
 [[2 3 5]
 [4 3 4]]
Xd =
 [[0.40352267 0.27993912]
 [0.09285485 0.80284623]]
----------------------------------------
current: F[1]=0.8285256624, gBest: F[1]=0.8285256624
----------------------------------------
```

```
############################################################
                    15  th generation
############################################################
<sol=%d> SSO_updated parameters:
XXk =
 [5 4 7]
XXv =
 [4 3 4]
XXd =
 [0.40352267 0.05113537]
Train on 5216 samples, validate on 16 samples
5216/5216 [==============================] - 3s 571us/sample - loss: 0.5555 - categorical_crossentropy: 0.5555 - acc: 0.7519 -
val_loss: 0.4728 - val_categorical_crossentropy: 0.4728 - val_acc: 0.8125
624/624 [==============================] - 0s 179us/sample - loss: 0.4488 - categorical_crossentropy: 0.4488 - acc: 0.8109
Xk =
 [[5 4 8]
 [5 4 8]]
Xv =
 [[2 3 5]
 [4 3 4]]
Xd =
 [[0.40352267 0.27993912]
 [0.09285485 0.80284623]]
----------------------------------------
current: F[0]=0.6554487348, gBest: F[1]=0.8285256624
----------------------------------------
<sol=%d> SSO_updated parameters:
XXk =
 [8 4 8]
XXv =
 [4 3 4]
XXd =
 [0.09285485 0.80284623]
Train on 5216 samples, validate on 16 samples
5216/5216 [==============================] - 15s 3ms/sample - loss: 0.4936 - categorical_crossentropy: 0.4936 - acc: 0.7776 -
val_loss: 0.4221 - val_categorical_crossentropy: 0.4221 - val_acc: 0.8750
624/624 [==============================] - 1s 1ms/sample - loss: 0.3929 - categorical_crossentropy: 0.3929 - acc: 0.8317
Xk =
 [[5 4 8]
 [8 4 8]]
Xv =
 [[2 3 5]
 [4 3 4]]
Xd =
 [[0.40352267 0.27993912]
 [0.09285485 0.80284623]]
----------------------------------------
current: F[1]=0.8317307830, gBest: F[1]=0.8317307830
----------------------------------------
```

```
############################################################
                    20  th generation
############################################################
<sol=%d> SSO_updated parameters:
XXk =
 [7 4 8]
XXv =
 [4 3 4]
XXd =
 [0.09285485 0.27993912]
Train on 5216 samples, validate on 16 samples
5216/5216 [==============================] - 8s 2ms/sample - loss: 0.5051 - categorical_crossentropy: 0.5051 - acc: 0.7749 -
val_loss: 0.4301 - val_categorical_crossentropy: 0.4301 - val_acc: 0.8750
624/624 [==============================] - 0s 516us/sample - loss: 0.3915 - categorical_crossentropy: 0.3915 - acc: 0.8285
Xk =
 [[8 4 8]
 [8 4 8]]
Xv =
 [[4 3 4]
 [4 3 4]]
Xd =
 [[0.09285485 0.27993912]
 [0.09285485 0.80284623]]
----------------------------------------
current: F[0]=0.8381410241, gBest: F[0]=0.8381410241
----------------------------------------
<sol=%d> SSO_updated parameters:
XXk =
 [8 6 8]
XXv =
 [4 3 4]
XXd =
 [0.09285485 0.27993912]
Train on 5216 samples, validate on 16 samples
5216/5216 [==============================] - 32s 6ms/sample - loss: 0.5892 - categorical_crossentropy: 0.5892 - acc: 0.7753 -
val_loss: 0.4710 - val_categorical_crossentropy: 0.4710 - val_acc: 0.9375
624/624 [==============================] - 2s 3ms/sample - loss: 0.4188 - categorical_crossentropy: 0.4188 - acc: 0.8365
Xk =
 [[8 4 8]
 [8 4 8]]
Xv =
 [[4 3 4]
 [4 3 4]]
Xd =
 [[0.09285485 0.27993912]
 [0.09285485 0.80284623]]
----------------------------------------
current: F[1]=0.8317307830, gBest: F[0]=0.8381410241
----------------------------------------
```

```
############################################################
                      25  th generation
############################################################
<sol=%d> SSO_updated parameters:
XXk =
 [7 4 8]
XXv =
 [4 3 2]
XXd =
 [0.12099458 0.27993912]
Train on 5216 samples, validate on 16 samples
5216/5216 [==============================] - 9s 2ms/sample - loss: 0.5556 - categorical_crossentropy: 0.5556 - acc: 0.7659 -
val_loss: 0.3968 - val_categorical_crossentropy: 0.3968 - val_acc: 0.9375
624/624 [==============================] - 0s 585us/sample - loss: 0.3932 - categorical_crossentropy: 0.3932 - acc: 0.8285
Xk =
 [[8 4 8]
 [8 4 8]]
Xv =
 [[4 3 4]
 [4 3 4]]
Xd =
 [[0.09285485 0.27993912]
 [0.09285485 0.27993912]]
----------------------------------------
current: F[0]=0.8381410241, gBest: F[1]=0.8525640965
----------------------------------------
<sol=%d> SSO_updated parameters:
XXk =
 [8 4 8]
XXv =
 [4 3 4]
XXd =
 [0.09285485 0.27993912]
Train on 5216 samples, validate on 16 samples
5216/5216 [==============================] - 15s 3ms/sample - loss: 0.5068 - categorical_crossentropy: 0.5068 - acc: 0.7860 -
val_loss: 0.5614 - val_categorical_crossentropy: 0.5614 - val_acc: 0.6250
624/624 [==============================] - 1s 997us/sample - loss: 0.4215 - categorical_crossentropy: 0.4215 - acc: 0.7901
Xk =
 [[8 4 8]
 [8 4 8]]
Xv =
 [[4 3 4]
 [4 3 4]]
Xd =
 [[0.09285485 0.27993912]
 [0.09285485 0.27993912]]
----------------------------------------
current: F[1]=0.8525640965, gBest: F[1]=0.8525640965
----------------------------------------
```

```
############################################################
                      30  th generation
############################################################
<sol=%d> SSO_updated parameters:
XXk =
 [8 4 6]
XXv =
 [4 3 4]
XXd =
 [0.09285485 0.27993912]
Train on 5216 samples, validate on 16 samples
5216/5216 [==============================] - 15s 3ms/sample - loss: 0.4795 - categorical_crossentropy: 0.4795 - acc: 0.7851 -
val_loss: 0.3574 - val_categorical_crossentropy: 0.3574 - val_acc: 0.9375
624/624 [==============================] - 1s 1ms/sample - loss: 0.3808 - categorical_crossentropy: 0.3808 - acc: 0.8221
Xk =
 [[8 4 8]
 [8 4 8]]
Xv =
 [[4 3 4]
 [4 3 4]]
Xd =
 [[0.09285485 0.27993912]
 [0.09285485 0.27993912]]
----------------------------------------
current: F[0]=0.8381410241, gBest: F[1]=0.8525640965
----------------------------------------
<sol=%d> SSO_updated parameters:
XXk =
 [8 4 8]
XXv =
 [4 3 4]
XXd =
 [0.09285485 0.27993912]
Train on 5216 samples, validate on 16 samples
5216/5216 [==============================] - 14s 3ms/sample - loss: 0.5009 - categorical_crossentropy: 0.5009 - acc: 0.7937 -
val_loss: 0.4500 - val_categorical_crossentropy: 0.4500 - val_acc: 0.8750
624/624 [==============================] - 1s 989us/sample - loss: 0.3858 - categorical_crossentropy: 0.3858 - acc: 0.8253
Xk =
 [[8 4 8]
 [8 4 8]]
Xv =
 [[4 3 4]
 [4 3 4]]
Xd =
 [[0.09285485 0.27993912]
 [0.09285485 0.27993912]]
----------------------------------------
current: F[1]=0.8525640965, gBest: F[1]=0.8525640965
```

```
############################################################
                      35  th generation
############################################################
<sol=%d> SSO_updated parameters:
XXk =
 [8 4 8]
XXv =
 [4 3 4]
XXd =
 [0.09285485 0.27993912]
Train on 5216 samples, validate on 16 samples
5216/5216 [==============================] - 15s 3ms/sample - loss: 0.5412 - categorical_crossentropy: 0.5412 - acc: 0.7669 -
val_loss: 0.6326 - val_categorical_crossentropy: 0.6326 - val_acc: 0.6250
624/624 [==============================] - 1s 1ms/sample - loss: 0.4782 - categorical_crossentropy: 0.4782 - acc: 0.7660
Xk =
 [[8 4 8]
 [8 4 8]]
Xv =
 [[4 3 4]
 [4 3 4]]
Xd =
 [[0.09285485 0.27993912]
 [0.09285485 0.27993912]]
----------------------------------------
current: F[0]=0.8381410241, gBest: F[1]=0.8525640965
----------------------------------------
<sol=%d> SSO_updated parameters:
XXk =
 [8 4 8]
XXv =
 [4 3 2]
XXd =
 [0.09285485 0.27993912]
Train on 5216 samples, validate on 16 samples
5216/5216 [==============================] - 16s 3ms/sample - loss: 0.5702 - categorical_crossentropy: 0.5702 - acc: 0.7749 -
val_loss: 0.5889 - val_categorical_crossentropy: 0.5889 - val_acc: 0.6250
624/624 [==============================] - 1s 1ms/sample - loss: 0.4646 - categorical_crossentropy: 0.4646 - acc: 0.7500
Xk =
 [[8 4 8]
 [8 4 8]]
Xv =
 [[4 3 4]
 [4 3 4]]
Xd =
 [[0.09285485 0.27993912]
 [0.09285485 0.27993912]]
----------------------------------------
current: F[1]=0.8525640965, gBest: F[1]=0.8525640965
----------------------------------------
```

```
############################################################
                      40  th generation
############################################################
<sol=%d> SSO_updated parameters:
XXk =
 [8 4 8]
XXv =
 [3 3 4]
XXd =
 [0.09285485 0.27993912]
Train on 5216 samples, validate on 16 samples
5216/5216 [==============================] - 16s 3ms/sample - loss: 0.5910 - categorical_crossentropy: 0.5910 - acc: 0.7542 -
val_loss: 0.6422 - val_categorical_crossentropy: 0.6422 - val_acc: 0.6250
624/624 [==============================] - 1s 1ms/sample - loss: 0.5112 - categorical_crossentropy: 0.5112 - acc: 0.7067
Xk =
 [[8 4 8]
 [8 4 8]]
Xv =
 [[4 3 4]
 [4 3 4]]
Xd =
 [[0.09285485 0.27993912]
 [0.09285485 0.27993912]]
----------------------------------------
current: F[0]=0.8381410241, gBest: F[1]=0.8525640965
----------------------------------------
<sol=%d> SSO_updated parameters:
XXk =
 [8 4 8]
XXv =
 [4 3 4]
XXd =
 [0.09285485 0.27993912]
Train on 5216 samples, validate on 16 samples
5216/5216 [==============================] - 15s 3ms/sample - loss: 0.5025 - categorical_crossentropy: 0.5025 - acc: 0.7713 -
val_loss: 0.3810 - val_categorical_crossentropy: 0.3810 - val_acc: 0.8750
624/624 [==============================] - 1s 1ms/sample - loss: 0.4179 - categorical_crossentropy: 0.4179 - acc: 0.8109
Xk =
 [[8 4 8]
 [8 4 8]]
Xv =
 [[4 3 4]
 [4 3 4]]
Xd =
 [[0.09285485 0.27993912]
 [0.09285485 0.27993912]]
----------------------------------------
current: F[1]=0.8525640965, gBest: F[1]=0.8525640965
----------------------------------------
```

```
############################################################
                      45  th generation
############################################################
<sol=%d> SSO_updated parameters:
XXk =
 [5 4 5]
XXv =
 [4 4 4]
XXd =
 [0.09285485 0.00783526]
Train on 5216 samples, validate on 16 samples
5216/5216 [==============================] - 11s 2ms/sample - loss: 0.5115 - categorical_crossentropy: 0.5115 - acc: 0.7701 -
val_loss: 0.4601 - val_categorical_crossentropy: 0.4601 - val_acc: 0.8750
624/624 [==============================] - 0s 275us/sample - loss: 0.4396 - categorical_crossentropy: 0.4396 - acc: 0.8077
Xk =
 [[8 4 8]
 [8 4 8]]
Xv =
 [[4 3 4]
 [4 3 4]]
Xd =
 [[0.09285485 0.27993912]
 [0.09285485 0.27993912]]
----------------------------------------
current: F[0]=0.8381410241, gBest: F[1]=0.8701922894
----------------------------------------
<sol=%d> SSO_updated parameters:
XXk =
 [7 4 8]
XXv =
 [4 3 4]
XXd =
 [0.09285485 0.59293023]
Train on 5216 samples, validate on 16 samples
5216/5216 [==============================] - 9s 2ms/sample - loss: 0.5438 - categorical_crossentropy: 0.5438 - acc: 0.7632 -
val_loss: 0.6245 - val_categorical_crossentropy: 0.6245 - val_acc: 0.6250
624/624 [==============================] - 0s 555us/sample - loss: 0.4410 - categorical_crossentropy: 0.4410 - acc: 0.7788
Xk =
 [[8 4 8]
 [8 4 8]]
Xv =
 [[4 3 4]
 [4 3 4]]
Xd =
 [[0.09285485 0.27993912]
 [0.09285485 0.27993912]]
----------------------------------------
current: F[1]=0.8701922894, gBest: F[1]=0.8701922894
----------------------------------------
```

```
############################################################
                      49  th generation
############################################################
<sol=%d> SSO_updated parameters:
XXk =
 [8 4 8]
XXv =
 [4 3 4]
XXd =
 [0.09285485 0.27993912]
Train on 5216 samples, validate on 16 samples
5216/5216 [==============================] - 16s 3ms/sample - loss: 0.5503 - categorical_crossentropy: 0.5503 - acc: 0.7742 -
val_loss: 0.4745 - val_categorical_crossentropy: 0.4745 - val_acc: 0.8125
624/624 [==============================] - 1s 1ms/sample - loss: 0.4093 - categorical_crossentropy: 0.4093 - acc: 0.8205
Xk =
 [[8 4 8]
 [8 4 8]]
Xv =
 [[4 3 4]
 [4 3 4]]
Xd =
 [[0.09285485 0.27993912]
 [0.09285485 0.27993912]]
----------------------------------------
current: F[0]=0.8381410241, gBest: F[1]=0.8701922894
----------------------------------------
<sol=%d> SSO_updated parameters:
XXk =
 [8 4 8]
XXv =
 [4 3 4]
XXd =
 [0.09285485 0.27993912]
Train on 5216 samples, validate on 16 samples
5216/5216 [==============================] - 16s 3ms/sample - loss: 0.5528 - categorical_crossentropy: 0.5528 - acc: 0.7765 -
val_loss: 0.3468 - val_categorical_crossentropy: 0.3468 - val_acc: 0.8750
624/624 [==============================] - 1s 1ms/sample - loss: 0.4019 - categorical_crossentropy: 0.4019 - acc: 0.8157
Xk =
 [[8 4 8]
 [8 4 8]]
Xv =
 [[4 3 4]
 [4 3 4]]
Xd =
 [[0.09285485 0.27993912]
 [0.09285485 0.27993912]]
----------------------------------------
current: F[1]=0.8701922894, gBest: F[1]=0.8701922894
----------------------------------------
```

可以發現，準確率從最初的 65.5%，提升到 87.0%，是具有顯著的提升，對於這麼樣一個簡單的 model，只透過調整裡面的參數，便可以達到將近九成的準確率，代表這個 SSO-CNN 是具有效力的

五、 結果與討論

1. 結論

透過以上兩種方法，分別的模型準確率為 74.7% 以及 87.0%，可以了解不只是 model 的選擇，當中參數的設定是相當重要的，如果用了過於複雜的模型未必能使準確率提高，還有可能會導致 overfitting，使模型太過擬合 training data。而簡單的模型未必不能達到好的預測效果，如果能夠適當的調整模型內參數，使該 model 更加吻合所需預測的資料，也能夠達到高準確率，使肺炎的判斷更為準確。

2. 未來展望

本實驗在 ISSO 所使用的C_g、C_p 為主觀訂定，為 0.7 以及 0.9，代表有 0.7 的機率下一次的迭代會使用當前最佳解，0.2 的機率會保留原解以及 0.1 的機率會隨機產生新解並取代。而此參數的不同可能會導致演算法的表現，若能夠適當的調整，或許能使整體績效再更提升。